Distributed Caching Algorithms in the Realm of Layered Video Streaming

Konstantinos Poularakis, *Member, IEEE*, George Iosifidis, *Member, IEEE*, Antonios Argyriou, *Senior Member, IEEE*, Iordanis Koutsopoulos, *Senior Member, IEEE*, and Leandros Tassiulas, *Fellow, IEEE*

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, which determines to an extent the user Quality of Experience (QoE). This problem obtains an interesting new twist with the advent of advanced layered-video encoding techniques such as Scalable Video Coding. In this paper, we show that the problem of finding the caching configuration of video encoding layers that minimizes delivery delay for a network operator is NP-Hard, and we establish a pseudopolynomial-time optimal solution by using a connection with the multiple-choice knapsack problem. Next, we design caching algorithms for multiple network 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 fetching content from distant servers. We derive an approximate solution to this cooperative caching problem by using a technique that partitions the cache capacity into amounts dedicated to own and other operators' caching needs. Trace-driven evaluations demonstrate up to 25 percent reduction in delay over existing caching schemes. As a side benefit, our algorithms achieve smoother playback for video streaming applications, with fewer playback stalls and higher decoded quality.

Index Terms—Distributed caching, cooperation, layered-video encoding

1 Introduction

1.1 Motivation

On-demand video is the driving force of the data tsunami that we are witnessing nowadays [2], 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, since delay constitutes a prime factor that determines the user quality of experience (QoE). 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) [3] and recently also for wireless mobile networks [4].

A key challenge in these architectures is to design the *optimal caching policy*: for a given anticipated video content demand, determine which content should be placed in each cache, so as to reduce the average video delivery delay[1] over all requests. If these requests are not satisfied by the locally available cache, content needs to be fetched from distant back-end servers, which induces significantly larger delay. Optimal caching is a well known NP-hard problem, and many heuristic or approximation algorithms have been proposed to address it [3], [4], [6], [7].

Nevertheless, a specific aspect has been hitherto overlooked. Today more often than not, networks deliver video files that are encoded at different *qualities* to their customers. Users may implicitly or explicitly ask for certain video quality (e.g., certain resolution for YouTube videos [8]), while in other cases the delivered video quality is determined by the user equipment (e.g., based on the mobile device model and screen size) or by the operator (e.g., based on agreements with content providers [9]). User mobility and the wireless channel further increase the need to have different qualities for streaming video to users. Depending on the wireless channel conditions, it makes sense to dynamically adapt the quality of the video stream, for example, high quality for good channel conditions, lower quality as channel conditions deteriorate.

These developments together with stringent requirements for higher user QoE and advances in video-encoding technology have led to the 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) [10], which allows for multiple spatial resolutions (frame sizes), different frame

- *K. Poularakis and L. Tassiulas are with the Department of Electrical Engineering, Yale Institute for Network Science, Yale University, New Haven, CT 06520. E-mail: kpoularakis@gmail.com, leandros.tassiulas@yale.edu.*
- *G. Iosifidis is with the School of Computer Science and Statistics, Trinity College Dublin, Dublin 2, Ireland. E-mail: giosifid@gmail.com.*
- *A. Argyriou is with the Department of Electrical and Computer Engineering, University of Thessaly, Filellinon, Volos 382 21, Greece. E-mail: anargyr@gmail.com.*
- *I. Koutsopoulos is with the Department of Informatics, Athens University of Economics and Business, Athina 104 34, Greece. E-mail: jordan@aueb.gr.*

Manuscript received 6 Apr. 2017; revised 1 Apr. 2018; accepted 10 June 2018. Date of publication 26 June 2018; date of current version 4 Mar. 2019. (Corresponding author: Konstantinos Poularakis.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2018.2850818

1. Video delivery delay refers to the time it takes from the moment the first packet of a video has been transmitted by the source until all the packets are delivered to the destination [5].

rates, or different signal-to-noise ratio (SNR) qualities. With SVC, each video file is encoded as 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 enhancement layer to achieve that quality. Moreover, with SVC, the user device has the option to adapt the playback quality of video stream by dynamically adding or dropping layers (e.g., always streaming the basic layer and optionally the enhancement layers).

SVC is considered today one of the emerging video technologies [11], and it is already used for video streaming [12], [13], web services [14], and video storage [15], among other applications. For completeness, we stress that an alternative to SVC technique is to perform transcoding of a video to lower bitrate versions in order to satisfy user requests [16], [17]. Although transcoding is often preferred in industry, it requires the real-time processing of the videos in the network before delivered to users. On the other hand, SVC alleviates the need for in-network processing, requiring only from user devices to combine the different layers together.

With SVC, it is possible to store different layers of a certain video at different caches. For a user that requests a video at a given quality level, the different required 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 video delivery delay metric is determined by the *largest* delay needed to deliver a layer among all layers required from a cache or a back-end server.

Due to SVC, the repertoire of caching policies increases significantly, as the caching decisions must be taken per layer and not per video file, while the video delivery delay experienced depends jointly on retrieval delays of all layers of the video for the required quality. Hence all previous theoretical results (e.g., approximation ratios [3], [4], [6], [7]) 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 the SVC caching problem has already been studied for various network architectures [18], [19], [20], [21], [22], [23], [24], [25], [26] these pioneering works do not provide optimal solutions and/or approximation ratio guarantees against optimal caching policies. *In this work, we fill this gap by addressing precisely the problem of minimizing user perceived video delivery delay* for a network operator through optimized caching of layered video content.

Moreover, going one step further, we study the delay performance benefits that may arise when different network operators cooperate in caching. Today there exist many network operators (e.g., Wireless Service Providers) that often deploy their own caches in the same locations so that each of them serves its users-clients. These caches may be amenable to joint coordination. Thus, it is meaningful to explore the possibility for a local cache that belongs to a certain network operator to retrieve a video layer from the co-located cache of another operator, instead of fetching it from a distant server of its own, which would cause larger delay. In fact, such cooperation scenarios between the caches attract increasing interest, especially in the context of wireless mobile networks [17], [27], [28]. However, the diverse user demands that different network operators must serve render this cooperative caching problem particularly challenging. *The second problem we tackle is to derive a joint caching policy of a set of involved network operators that minimizes the total delivery delay for all operators, considering the global content demand.*

1.2 Methodology and Contributions

We consider a general (abstract) distributed caching architecture comprising several *local nodes* with caching capability such as mobile switching centers [27], cellular base stations [28] or mobile edge computing servers [17], in the proximity of end-users. Requests for SVC-encoded video files at 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 aggregate delay for delivering requested 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* [29].

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 video delivery delay for all operators. We assume that users of different operators request the same set of video files (or, a common subset) with possibly different rates and quality requirements. Therefore, the cooperative policy may on average reduce the video delivery delay for users of some networks, and increase it for some others. Using a technique that *partitions the space of a cache owned by an operator into two parts, dedicated to own and other operators' caching needs respectively*, we present a solution algorithm with established approximation ratio.

The contribution of this work can be summarized as follows:

- *Layered Video Caching*. We model the problem that designs per-video-layer caching policies, aimed at optimizing the aggregate video delivery delay of users in a distributed caching network. This is a problem of increasing importance due to the momentum of layered-video encoding, especially in mobile networks where users often need to receive videos in different qualities depending on the wireless channel conditions. We reduce this to the MCK problem and provide a pseudopolynomial-time optimal and a FPTA algorithm [29].
- *Operator Cooperation*. We propose cooperation policies among different network operators (e.g., Wireless Service Providers [17], [27], [28]) and formulate the respective optimization problem for designing the globally optimal caching policy. Using a novel cache-partition technique, we establish an approximation algorithm that achieves at least half of the

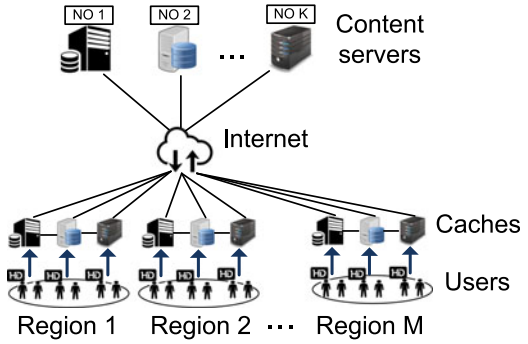


Fig. 1. A distributed caching architecture with K network operators and M geographical regions. Each cache is connected with a back-end content server and possibly with other caches in the same region.

optimal performance for a symmetric case with equal transmission rates of the links between cache-nodes.

- *Benefits in Average Delivery Delay*. We evaluate numerically the proposed schemes using system parameters driven from real traces. We show that our approach reduces average video delivery delay up to 25 percent over existing schemes for typical cache sizes and video popularity distributions.
- *Benefits in Video Streaming Performance*. Although the proposed algorithms are not designed to directly optimize performance metrics related to video streaming, we show that in practical scenarios they can indeed smoothen video playback by achieving fewer playback stalls and higher decoded quality. The benefits are more pronounced when the bandwidth capacity is relatively low.

The rest of the paper is organized as follows. Section 2 describes the system model and formalizes the layered video caching problem. Sections 3 and 4 describe our solution algorithms when network operators serve their requests independently from each other and when they cooperate respectively. Section 5 presents the evaluation results, while Section 6 reviews our contribution compared to related works. We conclude our work in Section 7.

2 System Model and Problem Statement

We consider a general (abstract) network architecture wherein a set $\mathcal{K}$ of K Network Operators (NOs), e.g., Wireless Service Providers, provide internet access to their subscribers, or users, distributed in a set $\mathcal{M}$ of M geographical regions. For each region, each NO has installed a cache at a certain location along the path from its subscribers to the back-end content server (e.g., at a mobile switching center [27], a base station [28] or a mobile edge computing server [17]). The NOs may act independently or in cooperation as we will explain in the sequel. An example caching network is depicted in Fig. 1 and the key notation is summarized in Table 1.

2.1 Independent Caching by Network Operators

We first consider the case when NOs act independently from each other and focus on a single NO $k \in \mathcal{K}$ and its subscribers. We denote by $\mathcal{N}_k$ the set of caches, or cache-nodes, of NO k, each one located at a different region. The capacity of cache $n \in \mathcal{N}_k$ is denoted by $C_n \geq 0$ (bytes). The average user

TABLE 1
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

demand for each pre-recorded video in a set $\mathcal{V} = \{1, 2, \ldots, V\}$ 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 [3], [4]. For example, the demand can be learned by analyzing previous time statistics of user request patterns to infer future demand or by using machine-learning techniques [30]. We consider that each video is available at some specific quality levels, indexed in a list $\mathcal{Q} = \{1, 2, \ldots, Q\}$. Each quality level may represent a different combination of temporal, spatial and SNR qualities. With SVC, there is a set $\mathcal{L}$ of layers, 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 lth layer of video v is denoted with $o_{vl} > 0$ (bytes), which typically decreases with l, i.e., $o_{v1} \geq o_{v2} \geq \cdots \geq o_{vQ}$ [11], [31].

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 choose to request videos at low-definition quality ($q = 1$), while the other half ask for high-definition (HD) quality ($q = 2$), as in Fig. 1. The user requests for different quality levels can also account for the wireless channel conditions. For example, users in certain regions may experience on average worse channel conditions than in the other regions, and hence the average requested video quality is lower in the former. To capture these factors, we denote by $\lambda_{nvq} \geq 0$ the average user demand associated with node n for the qth quality level of video v. In other words, the λ_{nvq} values can capture both the expected preferences of the users and the wireless channel conditions [22]. We define the request vector for each cache node n, and the total demand vector for NO k, respectively

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

In order to deliver to a user the qth 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 video streaming 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 that is delivered last, and hence the delay for delivering the entire video at the given quality will be equal to the *maximum* delay needed for each of these layers to be delivered.[2]

Ideally, the user would like to receive all the required layers from the locally available cache-node n of NO k 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 back-end content server that contains all videos and layers. Similarly to the works in [3], [4], [6], [7], [21] 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 [32], one connection for each layer, with fixed average bandwidth allocated per connection.

Let the binary decision variable x_{nvl} indicate whether the lth 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

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

Clearly, each node $n \in \mathcal{N}_k$ cannot cache more data than its capacity, i.e., it should hold that

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

Our goal is to design the caching policy that minimizes the aggregate video delivery delay[3] for all users of NO k, denoted by $J_k(\boldsymbol{x}_k)$

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

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.

2.2 Cooperative Caching Among Network Operators

Let us now consider the case that the NOs have decided to jointly coordinate their caches in the *same* region.[4] Therefore, one cache can send video layers to the other to satisfy the other's demand through wireline or wireless backhaul links as it is described in [17], [27], [28]. Assume that each NO in $\mathcal{K}$ serves requests for the same set $\mathcal{V}$ of videos.[5] Nevertheless, each NO has its own subscribers 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

$$\mathcal{N} = \bigcup_{k\in\mathcal{K}} \mathcal{N}_k, \tag{5}$$

and the total expected demand

$$\Lambda = \bigcup_{k\in\mathcal{K}} \boldsymbol{\lambda}_k. \tag{6}$$

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 per unit data 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 video delivery delay will be equal to the *maximum* of the respective delays.

The objective of the cooperating NOs is to minimize the total 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

$$J_T^c(\boldsymbol{x}) = \sum_{k\in\mathcal{K}} J_k^c(\boldsymbol{x}), \tag{7}$$

where

$$\begin{aligned} &J_k^c(\boldsymbol{x}) \\ &= \sum_{n\in\mathcal{N}_k} \sum_{v\in\mathcal{V}} \sum_{q\in\mathcal{Q}} \lambda_{nvq} \max_{l\in\{1,\ldots,q\}} \Bigg\{ \prod_{\substack{n'\in\mathcal{N}:\\ \mathcal{M}_{n'}=\mathcal{M}_n}} (1 - x_{n'vl}) o_{vl} d_n \\ &\quad + \left(1 - \prod_{\substack{n'\in\mathcal{N}:\\ \mathcal{M}_{n'}=\mathcal{M}_n}} (1 - x_{n'vl})\right) o_{vl} \min_{\substack{n'\in\mathcal{N}:\\ \mathcal{M}_{n'}=\mathcal{M}_n,\, x_{n'vl}=1}} \{d_{nn'}\} \Bigg\}. \end{aligned} \tag{8}$$

In the above expression, $\mathcal{M}_n \in \mathcal{M}$ indicates the region where node n is located. Every required layer $l \in \{1, \ldots, q\}$ 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.

2.3 Motivating Example

The benefits that such cooperation policies may yield can be easily understood through the simple example in Fig. 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

2. We note that in a non-streaming system, the required layers could be received serially (rather than in parallel) by the user. In this case, the delay would be the sum (rather than the max) of the respective layer delays.

3. We focus on the video delivery delay to study the impact of the network on user service. The delay introduced by other applications like decoding and buffering processes is neglected.

4. We note that still cache-nodes at different regions act independently each other. This is because the regions are in general far away each other and hence the delay required for exchanging content can be very large in practice.

5. Our model captures also the case that the network operators provide different, yet overlapping sets of videos, in which case $\mathcal{V}$ stands for the overlapping video set.

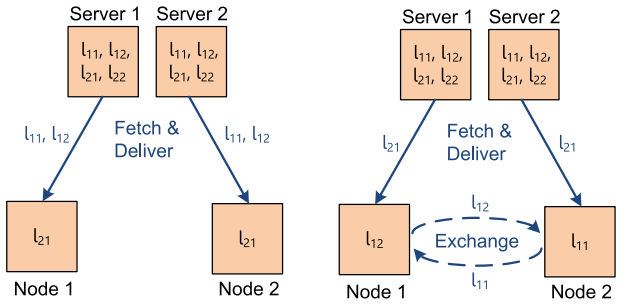


Fig. 2. An example illustrating the benefits of cooperative caching for two network operators.

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 layer caching solutions, we remark that our model considers the delay required for delivering the entire video at certain quality asked by the user. We choose this metric (delay) for mathematical tractability in an optimization framework in a time-average sense, with the understanding that this metric and its optimization will also have positive repercussions on QoE performance metrics related to video streaming. We explore this issue numerically in Section 5 and further discuss it in the online Appendix [33].

3 Independent Caching by Network Operators

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

$$\min_{\boldsymbol{x_k}} \quad J_k(\boldsymbol{x}_k) \tag{9}$$

$$s.t. \quad \sum_{v\in\mathcal{V}}\sum_{l\in\mathcal{L}} o_{vl} x_{nvl} \leq C_n, \forall n \in \mathcal{N}_k, \tag{10}$$

$$x_{nvl} \in \{0,1\},\ \forall n \in \mathcal{N}_k,\ v \in \mathcal{V},\ l \in \mathcal{L}. \tag{11}$$

3.1 Problem Decomposition

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 $\sum_{v\in\mathcal{V}}\sum_{q\in\mathcal{Q}} \lambda_{nvq} o_{vl} d_n$ where $l = 1$. 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 $\sum_{q\in\mathcal{Q}} \lambda_{nvq} \cdot d_n \cdot (o_{v1} - o_{v2})$, 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 $\sum_{q\in\mathcal{Q}} \lambda_{nvq} \cdot d_n \cdot (o_{v2} - o_{v3})$ more, and so on. Hence, the equivalent problem of *maximizing the delay savings for node* n (named P_n) can be expressed as follows:

$$P_n: \max_{\boldsymbol{x_n}} \sum_{v\in\mathcal{V}}\sum_{q\in\mathcal{Q}} \lambda_{nvq} d_n \sum_{l=1}^{q} (o_{vl} - o_{v,l+1}) \prod_{i=1}^{l} x_{nvi} \tag{12}$$

$$s.t.\ \text{constraint: } (3), \quad x_{nvl} \in \{0,1\},\ \forall v \in \mathcal{V},\ l \in \mathcal{L}, \tag{13}$$

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 $o_{v,l+1}$ to be equal to zero for $l = q$ in the above summation.

Subsequently, we characterize the complexity of problem P_n, and present efficient solutions.

3.2 Complexity and Solution to Problem P_n

We first prove the intractability of the problem P_n in Theorem 1.

Theorem 1. *Problem P_n is NP-Hard.*

Proof. We prove the NP-Hardness of the problem P_n by reduction from the *Knapsack problem*, which is NP-Hard [29]. The latter is defined as follows: Given a knapsack of capacity W, and a set of T items with nonnegative weights w_1 to w_T and values p_1 to p_T, the objective is to place in the knapsack the subset of items of total weight no more than W with the largest total value. Every instance of the knapsack problem can be written as a special case of the problem P_n, where there is one video for each item $(V = T)$, each video has one quality level $(Q = 1)$, each layer is of size equal to the weight of the mapped item $(o_{v1} = w_v, \forall v \in \mathcal{V})$ and the demand for each video is equal to the value of the mapped item $(\lambda_{nv1} = p_v, \forall v \in \mathcal{V})$. Given a solution to the problem P_n one can find a solution to the knapsack problem of the same value by placing in the knapsack the items corresponding to the layers placed in the cache of node n. □

The following lemma provides information about the structure of the optimal solution.

Lemma 1. *There is an optimal solution to P_n such that, if a layer l is cached, then all the previous layers $l' < l$ of the same video are also cached.*

Proof. Let us assume that the optimal solution to problem P_n 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 the problem P_n to the following variant of the knapsack problem [34]:

Definition 1 (Multiple-Choice Knapsack). *Given R classes $E_1, E_2, \ldots, E_R$ of items to pack in a knapsack of capacity W, where the ith 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 P_n and MCK problems in the following lemma.

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

Proof. Given an instance of the problem P_n, we construct the equivalent instance of the problem MCK as follows: There is a knapsack of size equal to C_n and V item classes $E_1, E_2, \ldots, E_V$, one class for each video. Each class contains Q items, one item for each quality. The ith item in class E_v has a weight

$$w_{vi} = \sum_{l=1}^{i} o_{vl}, \tag{14}$$

and a value

$$p_{vi} = \sum_{q \in \mathcal{Q}} \lambda_{nvq} d_n \sum_{l=1}^{q} (o_{vl} - o_{v,l+1}) \prod_{j=1}^{l} (\mathbf{1}_{\{j \in \{1,2,\ldots,i\}\}}), \tag{15}$$

where $\mathbf{1}_{\{.\}}$ is the indicator function, i.e., it is equal to 1 if the condition in the subscript is true; otherwise it is zero. We also set $o_{v,l+1} = 0$ for $l = q$.

Each maximum-value solution to the MCK instance can be mapped to a solution to the P_n instance of the same value as follows: For each item i in class E_v packed in the knapsack, place the i first layers of video v to the cache-node n. Clearly, the obtained solution stores no more data than the cache capacity and satisfies the property in Lemma 1. By Eq. (15), the values of the items placed in the knapsack are equal to the delay savings obtained by the cached layers. Hence, the value of the solution to the problem P_n is equal to the solution value of the problem MCK.

Conversely, for every feasible solution to the problem P_n there is a feasible solution to the MCK instance of the same value. That is, for each sequence of i layers of video v placed in the cache-node n, we pack the item i of class v in the knapsack. Clearly, the obtained solution packs no more item weight than the knapsack capacity, and at most one item from each class is packed in the knapsack. □

Lemma 2 provides a valuable result, since it paves the way for exploiting a wide range of efficient algorithms that have been proposed for problem MCK in order to solve problem P_n. Specifically, although MCK is NP-hard, there exists a *pseudopolynomial-time optimal* algorithm and a *fully-polynomial-time approximation* algorithm to solve it [34]. 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.*

4 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 network operators. The problem of determining the caching policy that *minimizes the total user delay of all NOs* can be expressed as follows:

$$\min_{\mathbf{x}} \quad J_T^c(\boldsymbol{x}) \tag{16}$$

$$s.t. \quad \sum_{v \in \mathcal{V}} \sum_{l \in \mathcal{L}} o_{vl} x_{nvl} \leq C_n, \forall n \in \mathcal{N}, \tag{17}$$

$$x_{nvl} \in \{0,1\}, \ \forall n \in \mathcal{N}, v \in \mathcal{V}, l \in \mathcal{L}, \tag{18}$$

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

4.1 Problem 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

$$D_{wc}^m = \sum_{n \in \mathcal{N}_m} \sum_{v \in \mathcal{V}} \sum_{q \in \mathcal{Q}} \lambda_{nvq} o_{vl} d_n, \text{ where } l = 1, \tag{19}$$

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:

$$R_m : \max_{\boldsymbol{x_m}} D_{wc}^m - \sum_{n \in \mathcal{N}_m} \sum_{v \in \mathcal{V}} \sum_{q \in \mathcal{Q}} \lambda_{nvq} \max_{l \in \{1,\ldots,q\}} \left\{ \prod_{n' \in \mathcal{N}_m} (1 - x_{n'vl}) o_{vl} d_n + (1 - \prod_{n' \in \mathcal{N}_m} (1 - x_{n'vl})) o_{vl} d_{nn^*} \right\} \tag{20}$$

$$s.t. \sum_{v \in \mathcal{V}} \sum_{l \in \mathcal{L}} o_{vl} x_{nvl} \leq C_n, \forall n \in \mathcal{N}_m \tag{21}$$

$$x_{nvl} \in \{0,1\}, \ \forall n \in \mathcal{N}_m, \ v \in \mathcal{V}, \ l \in \mathcal{L}, \tag{22}$$

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 $n^* = \arg\min_{n' \in \mathcal{N}_m : x_{n'vl}=1}\{d_{nn'}\}$.

4.2 Solution to Problem R_m

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*, i.e., the sum of local demands across nodes). 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* $F \in [0, 1]$. At a high level, F represents the portion of each cache that is filled in with *globally popular video content* (i.e., layers of videos that are popular with respect to the global demand), while the rest $1 - F$ portion is filled in with *locally popular video content* (i.e., layers of videos that are popular with respect to the local demand). 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 solutions to the following two problems:

1. $MCK(m)$: The instance of the problem MCK 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 ith item of the vth class has weight

$$w'_{vi} = \sum_{l=1}^{i} o_{vl}, \tag{23}$$

and value

$$p'_{vi} = \sum_{n \in \mathcal{N}_m} \sum_{q \in \mathcal{Q}} \lambda_{nvq} d_n \sum_{l=1}^{q} (o_{vl} - o_{v,l+1}) \prod_{j=1}^{l} (\mathbf{1}_{\{j \in \{1,2,\ldots,i\}\}}). \tag{24}$$

In the above expression $\mathbf{1}_{\{.\}}$ is the indicator function, i.e., $\mathbf{1}_{\{c\}} = 1$ if condition c is true; otherwise it is zero, and $o_{v,l+1} = 0$ for $l = q$. Here, the ith item of the vth class corresponds to the first i layers of video v.

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

Problem $MCK(m)$ represents the placement of globally popular video layers in the F portion of the caches. To this end, a knapsack is formed of size equal to the aggregate size of these portions. The item values w'_{vi} and p'_{vi} represent the cache space needed and the possible delay savings of delivering the first i layers of video v from a cache instead of the remote servers. Similarly, problem $P_n(\mathcal{A}_n)$ represents the placement of locally popular videos in the remaining portion $(1 - F)$ of the caches. To this end, it has to take into account the placement of layers of globally popular videos (set $\mathcal{A}_n$), to avoid wasting resources in placing again the same layers in the same cache.

We now present the proposed *Layer-aware Cooperative Caching (LCC)* algorithm, which operates in two stages:

- *Stage 1.* Solve the problem $MCK(m)$. 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 problem $P_n(\mathcal{A}_n)$, 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 problem* R_m*, where*

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

The proof of Theorem 3 is deferred to the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2018.2850818. 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.

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

$$\max_{0 \le F \le 1} \min\{\rho\mu,\ \rho'\mu'\}. \tag{25}$$

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 [35].

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 $MCK(m)$ and P_n, 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 problem MCK.*

Proof. It is easy to show that the property in Lemma 1 holds for problem $P_n(\mathcal{A}_n)$, since it has the same objective with

the problem P_n. Therefore, given an instance of the problem $P_n(\mathcal{A}_n)$, we can construct the equivalent instance of the problem MCK as follows: There is a knapsack of size equal to $C_n - |\mathcal{A}_n|$, where $|\mathcal{A}_n|$ denotes the total size of the layers in $\mathcal{A}_n$, and the item classes $E_1, E_2, \ldots, E_V$, each with Q items. The ith item in class E_v has a weight

$$w''_{vi} = \sum_{l \in \{1,2,\ldots,i\}, l \notin \mathcal{A}_n} o_{vl}, \tag{26}$$

and a value

$$p''_{vi} = \sum_{q \in \mathcal{Q}} \lambda_{nvq} d_n \sum_{l=1}^{q} (o_{vl} - o_{v,l+1}) \prod_{j=1}^{l} (\mathbf{1}_{\{j \in \{1,2,\ldots,i\}\}}) \cdot (\mathbf{1}_{\{\{1,2,\ldots,j\} \notin \mathcal{A}_n\}}), \tag{27}$$

where $1_{\{.\}}$ is the indicator function, i.e., it is equal to 1 if the condition in the subscript is true; otherwise it is zero, and $o_{v,l+1} = 0$ for $l = q$. 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 further discuss this issue in the Appendix, available in the online supplemental material.

5 Trace-Driven Evaluation

In this section, we present the evaluation results of the experiments that we have conducted to show the superiority of the proposed algorithms over state of the art methods. Specifically, we implement the following three caching algorithms:

- *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 problem P_n that is defined in Section 3.
- *Layer-aware Cooperative Caching:* The proposed cooperative algorithm in Section 4, 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 [4]*: 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 emphasize that the Femtocaching algorithm has been extensively used as a benchmark by previous works. It is well-known that this algorithm achieves near-optimal delay for the traditional (layer-agnostic) video caching problem. Therefore, a natural question is whether the efficiency of Femtocaching is maintained or novel algorithms are needed when the delivery of layered video is considered. Our evaluation study targets to answer this question.

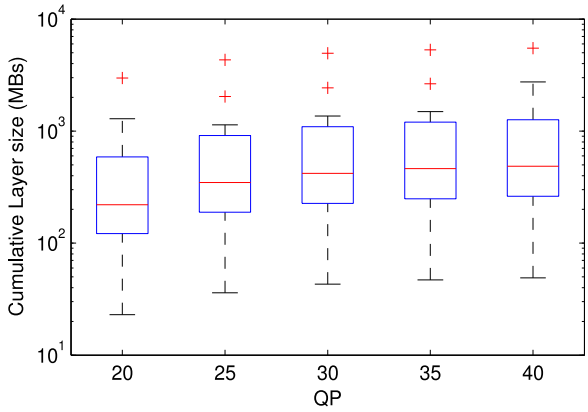


Fig. 3. The cumulative size of the layers required at each quality level for the videos in the library [32]. Each video is encoded into five quality levels corresponding to different quantization parameters; $QP \in \{20, 25, 30, 35, 40\}$.

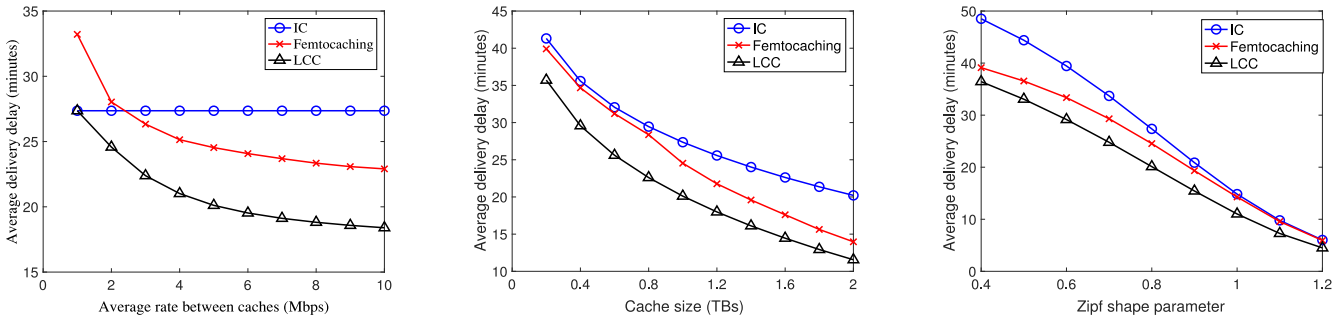
Before we proceed with the evaluation results, we remark that, in order to solve the problem MCK in IC and LCC schemes, we used the Mosek Optimization Toolbox. The execution time is in the scale of minutes. Our code is written in C language in the Visual Studio 2010 environment and it is publicly available online in [36]. We expect that the reproducibility of the results will encourage future experimentation with video caching algorithms for the benefit of the research community. In the sequel, we describe the evaluation setup used in the later evaluations.

5.1 Evaluation Setup

The evaluation is carried out for $K = 3$ NOs and a single geographical region ($M = 1$). Each NO has installed a cache of capacity equal to C (bytes). The rate with which a layer is delivered over the link between a content server and a cache 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: $1/d_{nn'} \in \{1, 2, \ldots, 10\}$ Mbps. We later explore the impact of dynamic rate, where the rate varies with the link load.

Requests for $V = 10{,}000$ 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 50,000 respective layers randomly using the real-world trace in [31]. This dataset contains detailed information about 19 SVC-encoded popular movies spanning 5 SNR quality levels (boxplot in Fig. 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 time duration of each video is between 5.5 minutes and 111 minutes. The total size of the 50,000 layers is slightly lower than 10 TBs.

Following empirical studies in VoD systems, we spread the user requests across the videos using a Zipf distribution, i.e., the request rate for the ith most popular video is proportional



(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 delivery rate of the links between caches, (b) the cache sizes, and (c) the shape parameter of the Zipf distribution.

to i^{-z}, for some shape parameter $z > 0$ [37]. We further spread the requests across the $Q = 5$ quality levels uniformly at random. Unless otherwise specified, we set: $C = 1$ TB [3] and $z = 0.8$ [37], while we run the LCC algorithm for each value of F at 0.1 granularity, and pick the value that results the lowest total delay.

5.2 Benefits in Average Delivery Delay

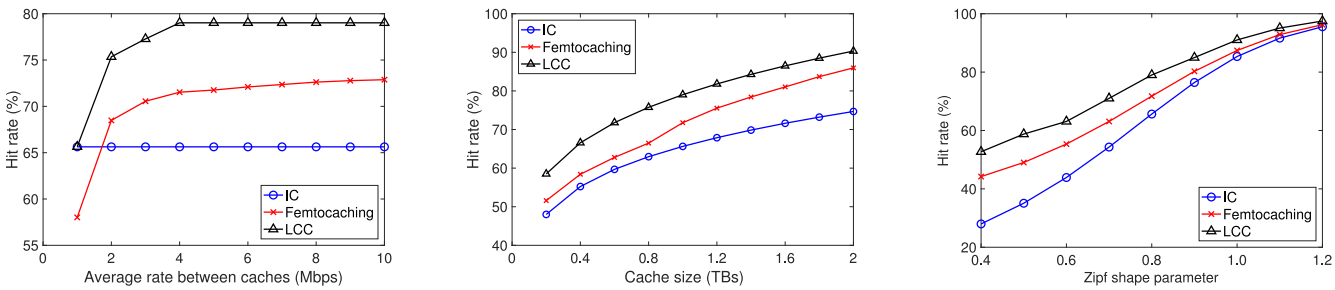
We first explore the impact of varying the bandwidth rate between the caches on the average (over all user requests) video delivery delay. In the experiment in Fig. 4a, the rate spans a wide range of values, starting from 1 to 10 Mbps, reflecting different operating conditions. We note that the performance of the IC algorithm is unaffected by this variation, since the caches are excluded from transmitting content one another. On the 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 33 and 20 percent when compared to IC and Femtocaching algorithm respectively.

We analyze the impact of cache sizes on performance in Fig. 4b. As expected, increasing cache sizes reduces delay for all the algorithms as more requests are satisfied without the involvement of the content server. The proposed LCC algorithm performs better than its counterparts for all the cache sizes. The gains over IC algorithm increase with cache sizes starting at 14 percent for 0.2 TBs and reaching 43 percent for 2 TBs. On the other hand, the gains over Femtocaching initially increase with cache sizes (from 10.5 percent for 0.2 TBs up to 21 percent for 0.8 TBs), but then they slightly reduce (down to 17.5 percent for 2 TBs). In other words, *LCC initially reduces delay at a higher pace than Femtocaching, but its performance starts to saturate first as the cache sizes increase.*

We show the impact of the Zipf shape parameter z on algorithms' performance in Fig. 4c. As the z value increases the video demand distribution becomes steeper and a few videos attract most of the demand. On the other hand, a small z value corresponds to an almost uniform video demand distribution. The delay decreases with z for all the algorithms, reflecting that *caching effectiveness improves with the steepness of video demand distribution*. LCC performs significantly better than IC and Femtocaching for all the values of z. The gains over IC are relatively stable over z (25-26 percent), while *the gains over Femtocaching steadily increase with z (from 7 percent for $z = 0.4$ up to 25 percent for $z = 1.2$)*.

In addition to delay, another metric of caching algorithm performance is the *cache hit rate*, i.e., the percentage of video data delivered by the caches instead of the remote servers. To explore this, we repeat the evaluations and depict the achieved cache hit rate in Figs. 5a, 5b, and 5c. We find that *the proposed algorithm (LCC) achieves higher cache hit rate than its counterparts for all the scenarios*. This, to some extent, indicates that *the optimization of the considered delay metric has positive implications on other metrics as well*.

Another issue is the timescale that the caching algorithms are applied. As the time passes, the demand for video content changes with new videos becoming popular and taking the place of older videos in the library $\mathcal{F}$. This content aging process impacts the efficiency of the caching decisions. Fig. 6 depicts the delay gains achieved by the three caching algorithms when 0-10 percent of the most popular videos are replaced with new (uncached) videos. Here, the new



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

Fig. 5. Cache hit rate achieved by IC, Femtocaching, and LCC algorithms as a function of (a) the delivery rate of the links between caches, (b) the cache sizes, and (c) the shape parameter of the Zipf distribution.

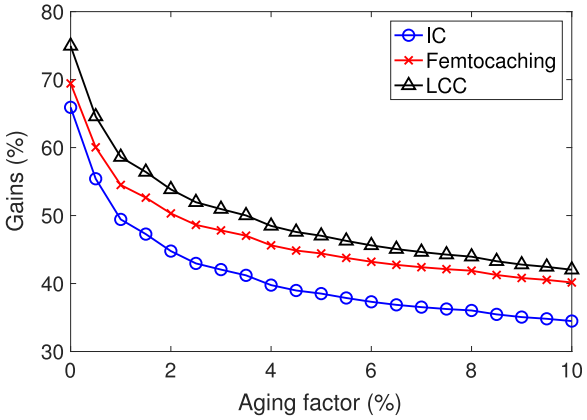


Fig. 6. Impact of content aging.

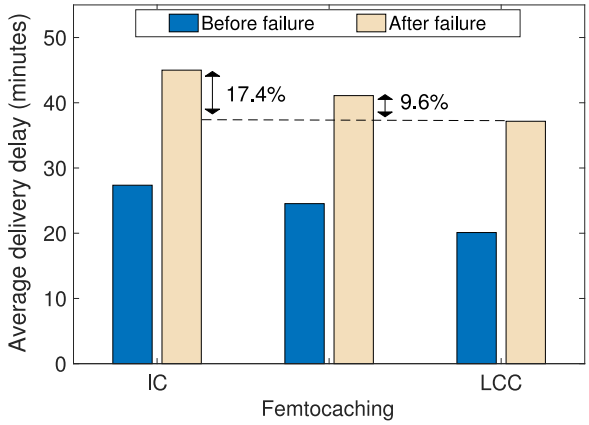


Fig. 7. Impact of node failure.

videos are positioned in the end of the demand vector λ_n, $\forall n$, i.e., they become the least popular videos. We find that, *although the gains decrease, the presented algorithms still achieve significant gains*. Depending on the scenario, injection of new videos can take hours or days. Hence, proactive caching can be a practical approach.

It would be also interesting to quantify the impact of failure of a cache-node on the efficiency of the proposed caching algorithms. Fig. 7 demonstrates that although the average delay increases after a node failure, still significant benefits are achieved. Importantly, the proposed caching algorithm (LCC) continues to perform better than the other two algorithms. Hence, our approach can be valuable even in these cases.

5.3 Benefits in Video Streaming Performance

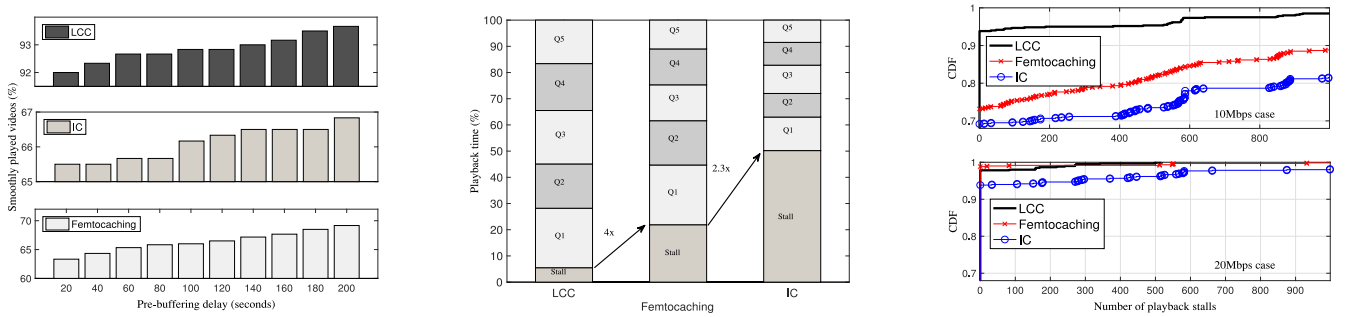
The evaluation results presented so far focused on the delay and cache hit rate associated to the delivery of videos at certain qualities asked by users. For video streaming applications, though, the user satisfaction depends on other metrics as well. In such cases, the user watches the video at the same time that it is downloaded. In order for playback to start there is a need to buffer a certain number of video frames that can be translated to either a portion of the file in bytes or seconds. We call this the *pre-buffering delay* measured in seconds. Due to network bandwidth fluctuations, the decoder might experience a buffer underrun which means that it requires data for decoding and playback but they have not yet been received. This is typically addressed with the undesired *playback video stalls*. Using advanced video streaming mechanisms, like dynamic adaptive streaming over HTTP (DASH) [38], video quality can be dynamically selected to ensure continuous playback. For example, if packets of a required layer are missing from the buffer, DASH can avoid a playback stall by decoding the video at a lower quality. In the extreme case that the base layer (quality level 1) is missing, the video playback will be inevitably stalled.

We implement such an adaptive video streaming mechanism and compare the performance achieved by the presented caching algorithms. Specifically, we consider a dynamic scenario where one video request is generated every minute for an overall period of 10 hours. The requests are distributed across the videos, quality levels and local nodes as in the previous experiments. The bandwidth capacities of the links between the caches and to the server are *shared* across the requested layers. In other words, at each time the delivery rate of a layer is not constant but it is given by the capacity of the respective link over the number of pending requests. This allows us to study the impact of bandwidth fluctuations. Each requested layer will be delivered either by the server or by a cache that has stored that layer. The decision is taken in a way that ensures the highest bandwidth for the request. A requested layer will not be set to play for the next second if the layer's portion that is already buffered is below the portion of the video that is already played. More advanced scheduling algorithms that adaptively determine the duration of the time that the player should wait for more data are out of the scope of this paper. Unless otherwise specified, we set the overall bandwidth capacity of the server links to 10 Mbps each, and 100 Mbps for the links between the caches.

Fig. 8a shows the percentage of video streams that are played smoothly, i.e., without any playback video stalls or quality degradations, for each caching algorithm. Here, different pre-buffering delays are evaluated (from 20 to 200 seconds). As expected, the number of smoothly played videos increases with the pre-buffering delay for all the caching algorithms. With LCC, more than 90 percent of the requested videos are played smoothly, while the respective values are below 70 percent for the rest caching algorithms. Overall, *LCC achieves up to 45 percent more smoothly played videos than its counterparts*.

Moving one step further, we explore how the playback time distributes across the different decoded video qualities and video stalls. Fig. 8b shows the results for pre-buffering delay equal to 100 seconds. With LCC, the video stalls take 5.5 percent of the playback time, which is 4 times lower than Femtocaching, and 9.2 times lower than IC. Moreover, with LCC, videos are played more often at the high quality level (Q5) than with the rest algorithms. Particularly, the playback time at Q5 is 16.6, 11.0 and 8.5 percent for LCC, Femtocaching and IC respectively. Overall, *with LCC video playback is stalled for shorter time and videos are played at higher quality than with the rest algorithms*.

Finally, we quantify the exact number of times that videos experience stalls during their playback. Fig. 8c shows the respective Cumulative Distribution Function (CDF) for pre-buffering delay equal to 100 seconds. Here, two different cases are studied; when the rate between a cache and a server is 10 Mbps (top subplot) and 20 Mbps (bottom subplot). In both cases, *LCC achieves fewer playback stalls for more streams compared to the rest algorithms*. Certain videos experience hundreds



(a) Impact of pre-buffering delay. (b) Playback time distribution. (c) CDF of playback stalls.

Fig. 8. (a) Impact of pre-buffering delay on number of smoothly played videos. (b) Playback time distribution across video stalls and qualities (quality 1 (Q1) to quality 5 (Q5)). (c) CDF of number of playback stalls for 10 Mbps (top subplot) and 20 Mbps (bottom subplot) rate of server links.

of playback stalls, which means that these videos are stalled during almost all their playback time (so they are not practically available for streaming). In the 10 Mbps case, about 5, 25 and 30 percent of the streams experience over a hundred playback stalls for the LCC, Femtocaching and IC algorithm respectively. These numbers are drastically reduced in the 20 Mbps case; 2 percent for LCC and Femtocaching, and 6 percent for IC. Overall, *LCC achieves significantly fewer playback stalls than the rest algorithms. The gains are more pronounced when the bandwidth capacities are relatively low.*

6 Related Work

6.1 Online and Offline Caching

The schemes for caching content can be classified into *online* (or reactive) and *offline* (or proactive). Online caching is a popular technique that stores content in caches on-demand. Examples include simple cache replacement algorithms such as the Least Frequently Used (LFU) and Least Recently Used (LRU), and other variants [39]. On the other hand, offline caching requires a priori knowledge of the popularity distribution of content, and based on that it optimizes caching decisions. This work focuses on offline caching.

Offline caching is in general an NP-Hard problem. Optimal solutions are limited to special cases with: (i) a few content files [40], (ii) ultra-metric costs between cache-nodes [41], (iii) single-hop groups of cache-nodes [42], and (iv) line caching networks [43]. 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 [3], [4], [6], [7]. The approximation ratios are derived by applying linear relaxation and rounding techniques, by expressing the objective function as a submodular set function or by dynamic programming techniques. Nevertheless, all the above results are not applicable for the case of layered 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.*

6.2 Caching in Wireless Mobile Networks

Caching in wireless mobile networks is a relatively new trend. The caches can be installed at several locations such as mobile switching centers [27], cellular base stations [28] or mobile edge computing servers [17]. Wireless mobile networks posse several unique features that can affect the efficiency of the caching policies. These include the (i) *broadcast/multicast* nature of the wireless medium, (ii) *interference* between the wireless links, and (iii) *mobility* of the end-users. Schemes that design caching policies jointly with the multicast schedule have been proposed in [44], [45]. A joint caching, channel assignment and routing algorithm has been proposed in [28]. Here, the caching solution takes into consideration the interference graph that dictates which transmissions of cached content are blocked by other transmissions. A scheme that combines caching with coordinated multipoint (CoMP) transmission, an interference mitigation technique, was proposed in [46]. In this case, the nodes can engage in CoMP and increase throughput if the same file is cached at many nearby nodes. Besides, a caching scheme that exploits predictions about the future mobility patterns of the users has been proposed in [47]. This scheme disperses parts of popular files to many cache-nodes that are likely to be encountered sequentially by the users as they move. Although the above works revealed some fundamental differences between wireline and wireless caching, they did not consider encoding of videos into different qualities.

6.3 Caching of Encoded Video

The video caching problem attracts increasing interest. The work of [16] and [17] proposed to serve the requests for different qualities of a video by caching a high bitrate version of that video and do rate-down conversion (transrating) for each request requiring a lower rate version. However, this method requires to use a processing resource to do the conversion at the cache side. If such a resource is not available, all the possible bitrate versions of the video need to be available in the cache, which consumes significant amount of cache space. An alternative method is to encode the video into multiple SVC layers which when combined achieve the requested video quality. SVC has been shown to improve video streaming performance by always downloading the base layer and optionally downloading the enhancement layers when there is enough available throughput [48]. Various schemes have been proposed for optimizing SVC video streaming including dynamic quality control based on network bandwidth conditions, user preferences and buffering capacities of client devices [49], [50]. Nevertheless, these works do not consider caching.

Exploiting SVC in video caching has been recently proposed in several contexts including CDN [18], IPTV [19], helper-assisted VoD [20], Software-defined RAN [21], Cloud-RAN [22] and small-cell wireless networks [23], [24], [25],

[26]. These works either compared SVC with other video encoding technologies, or they proposed heuristic-based or numerically evaluated layer caching schemes. For example, the work of [24] considered the same delay objective as ours and proposed a convex programming relaxation based heuristic algorithm for a set of non-collaborating cache-nodes. The work of [21] regarded the same delay metric as a constraint (instead of objective), and proposed a two-stage rounding heuristic algorithm that maximizes a reward function for a set of collaborating cache-nodes. The work of [22] considered an hierarchical setup where a parent node collaborates with child nodes in caching video layers. For this special scenario, an approximation algorithm was presented to maximize the overall cache hit rate. The work of [25] studied the channel diversity gains brought by caching the same layers in neighbour base stations and proposed another heuristic solution. Finally, the implications on security of layered video caching were investigated in [26]. Deviating from the above, in this work, 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*.

7 Conclusion

We studied distributed caching policies for layered encoded videos aiming to reduce the 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 percent delay gains over conventional (encoding layer-agnostic) caching schemes, as well as side benefits in QoE performance metrics related to video streaming (e.g., fewer playback stalls and higher decoded quality).

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 [3], [4], [6], [7], [21]) or change the objective to directly optimize QoE performance metrics related to video streaming and study how the results are affected.

Acknowledgments

Part of this work appeared in the *Proceedings of IEEE International Conference on Computer Communications (Infocom)*, April 2016 [1]. K. Poularakis acknowledges the Bodossaki Foundation, Greece, for a postdoctoral fellowship. G. Iosifidis acknowledges support by a research grant from Science Foundation Ireland (SFI) under Grant Number 17/CDA/4760. The research of I. Koutsopoulos was supported by AUEB-RC under the internal project "Original Scientific Publications". The work of L. Tassiulas was supported by the US Office of Naval Research (ONR) under award N00014-14-1-2190.

References

[1] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas, "Caching and operator cooperation policies for layered video content delivery," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.

[2] Ericsson, "Mobility report," Feb. 2018. [Online]. Available online: https://www.ericsson.com/assets/local/mobility-report/documents/2018/emr-interim-feb-2018.pdf

[3] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[4] 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, pp. 1107–1115.

[5] G. Almes, S. Kalidindi, and M. Zekauskas, "A one-way delay metric for IPPM," 2016. [Online]. Available: https://www.rfc-editor.org/rfc/pdfrfc/rfc7679.txt.pdf

[6] I. D. Baev and R. Rajaraman, "Approximation algorithms for data placement problems," *SIAM J. Comput.*, vol. 38, pp. 1411–1429, 2008.

[7] A. Khreishah and J. Chakareski, "Collaborative caching for multi-cell coordinated systems," in *Proc. IEEE INFOCOM Workshops*, 2015, pp. 257–262.

[8] YouTube, "Live encoder settings, bitrates and resolutions." [Online]. Available: https://support.google.com/youtube/answer/2853702?hl=en, accessed on 2018.

[9] E. Wyatt and N. Cohen, NY Times, "Comcast and Netflix reach deal on service," Feb. 2014. [Online]. Available online: https://www.nytimes.com/2014/02/24/business/media/comcast-and-netflix-reach-a-streaming-agreement.html

[10] H. Schwartz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[11] Cisco Webcasts, "Emerging video technologies: H.265, SVC, and WebRTC," 2014. [Online]. Available: https://www.ciscolive.com

[12] Vidyo. [Online]. Available: http://www.vidyo.com, accessed on 2018.

[13] RADVISION. [Online]. Available: http://www.radvision.com, accessed on 2018.

[14] Nojitter, "Google, Skype, and WebRTC," 2013. [Online]. Available: http://www.nojitter.com/post/240160776/google-skype-and-webrtc

[15] Strech Inc. [Online]. Available: http://www.stretchinc.com, accessed on 2018.

[16] H. A. Pedersen and S. Dey, "Enhancing mobile video capacity and quality using rate adaptation, RAN caching and processing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 996–1010, Apr. 2016.

[17] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in mobile-edge computing networks," in *Proc. IEEE 13th Annu. Conf. Wireless On-Demand Netw. Syst. Services*, 2017, pp. 165–172.

[18] F. Hartanto, J. Kangasharju, M. Reisslein, and K. Ross, "Caching video objects: Layers versus versions?," *Multimedia Tools Appl.*, vol. 2, pp. 45–48, 2006.

[19] Y. Sanchez, T. Schierl, C. Hellge, D. Hong, D. D. Vleeschauwer, W. V. Leekwijck, Y. Lelouedec, and T. Wiegand, "Improved caching for HTTP-based video on demand using scalable video coding," in *Proc. IEEE Consum. Commun. Netw. Conf.*, 2011, pp. 595–599.

[20] P. Ostovari, A. Khreishah, and J. Wu, "Multi-layer video streaming with helper nodes using network coding," in *Proc. IEEE 10th Int. Conf. Mobile Ad-Hoc Sensor Syst.*, 2013, pp. 524–532.

[21] S. Qin, M. Bennis, X. Chen, G. Feng, Z. Han, and G. Xue, "Enhancing software-defined RAN with collaborative caching and scalable video coding," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–6.

[22] Z. Zhang, D. Liu, and Y. Yuan, "Layered hierarchical caching for SVC-based HTTP adaptive streaming over C-RAN," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2017, pp. 1–6.

[23] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in *Proc. IEEE INFOCOM*, 2014, pp. 1078–1086.

[24] C. Zhan and Z. Wen, "Content cache placement for scalable video in heterogeneous wireless network," *IEEE Commun. Lett.*, vol. 21, no. 12, pp. 2714–2717, Dec. 2017.

[25] T. Zhen, Y. Xu, T. Yang, and B. Hu, "QoE-aware proactive caching of scalable videos over small cell networks," arXiv:1604.07572, 2016, Available online: https://arxiv.org/abs/1604.07572

[26] L. Xiang, D. W. K. Ng, R. Schober, and V. W. S. Wong, "Secure video streaming in heterogeneous small cell networks with untrusted cache helpers" in *Proc. IEEE Global Commun. Conf.*, 2017, pp. 1–7.

[27] J. Dai, F. Liu, B. Li, B. Li, and J. Liu, "Collaborative caching in wireless video streaming through resource auctions," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 2, pp. 458–466, Feb. 2012.

[28] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2275–2284, Aug. 2016.
[29] Y. Lien, "Some properties of 0-1 knapsack problems," in *Proc. Conf. Combinatorics Complexity*, 1987, pp. 1–25.
[30] E. Bastug, M. Bennis, and M. Debbah, "Anticipatory caching in small cell networks: A transfer learning approach," in *Proc. 1st KuVS Workshop Anticipatory Netw.*, 2014, pp. 1–3.
[31] Video Trace Library. [Online]. Available: http://trace.eas.asu.edu, accessed on 2018.
[32] N. Bouten, S. Latre, J. Famaey, F. De Turck, and W. Van Leekwijck, "Minimizing the impact of delay on live SVC-based HTTP adaptive streaming services," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, 2013, 1399–1404.
[33] Appendix. [Online]. Available: https://www.dropbox.com/s/43f7m31u35nqzfy/Appendix.pdf?dl=0, accessed on 2018.
[34] M. S. Bansal and V. C. Venkaiah, "Improved fully polynomial time approximation scheme for the 0-1 multiple-choice knapsack problem," in *Proc. SIAM Conf. Discrete Math.*, 2004, pp. 1–10.
[35] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[36] Publicly available code. [Online]. Available: https://www.dropbox.com/s/s9yequ71ytlkylz/infocom16code.rar?dl=0, accessed on 2018.
[37] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE/ACM Trans. Netw.*, vol. 16, no. 6, pp. 1447–1460, Dec. 2008.
[38] T. Stockhammer, "Dynamic adaptive streaming over HTTP –: Standards and design principles," in *Proc. Annu. ACM Conf. Multimedia Syst.*, 2011, pp. 133–144.
[39] S. Li, J. Xu, M. Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Trans. Multimedia*, vol. 18, no. 12, pp. 2503–2516, Dec. 2016.
[40] 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, pp. 936–944.
[41] M. Korupolu, C. G. Plaxton, and R. Rajaraman, "Placement algorithms for hierarchical cooperative caching," in *Proc. 10th Annu. ACM/SIAM Symp. Discrete Algorithms*, 1999, pp. 586–595.
[42] M. Taghizadeh, K. Micinski, C. Ofria, E. Torng, and S. Biswas, "Distributed cooperative caching in social wireless networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 6, pp. 1037–1053, Jun. 2013.
[43] K. Poularakis and L. Tassiulas, "On the complexity of content placement in hierarchical caching networks," *IEEE Trans. Commun.*, vol. 64, no. 5, pp. 2092–2103, May 2016.
[44] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[45] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5G wireless networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 4, pp. 2995–3007, Apr. 2016.
[46] A. Liu and V. K. N. Lau, "Mixed-timescale precoding and cache control in cached MIMO interference network," *IEEE Trans. Signal Process.*, vol. 61, no. 24, pp. 6320–6332, Dec. 2013.
[47] K. Poularakis and L. Tassiulas, "Code, cache and deliver on the move: A novel caching paradigm in hyper-dense small-cell networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 3, pp. 675–687, Mar. 2017.
[48] J. Famaey, S. Latre, N. Bouten, and F. D. Turck, "On the merits of SVC-based HTTP adaptive streaming," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, 2013, pp. 419–426.
[49] X. Wang, J. Chen, A. Dutta, and M. Chiang, "Adaptive video streaming over whitespace: SVC for 3-tiered spectrum sharing," in *Proc. IEEE INFOCOM*, 2015, pp. 28–36.
[50] X. Li and B. Veeravalli, "A differentiated quality adaptation approach for scalable streaming services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 8, pp. 2089–2099, Aug. 2015.
[51] J. F. Nash, "The bargaining problem," *Econometrica: J. Econometric Soc.*, vol. 18, no. 2, pp. 155–162, 1950.

Konstantinos Poularakis received the diploma, MS, and PhD degrees in electrical engineering from the University of Thessaly, Greece, in 2011, 2013, and 2015, respectively. Currently, he is a post-doc researcher and a member of the Institute for Network Science, Yale University. His research interests lie in the broad area of network optimization. He has been honored with several awards and scholarships during his studies, from sources including the Greek State Scholarships Foundation (IKY), the Center for Research and Technology Hellas (CERTH), and the "Alexander S. Onassis Public Benefit Foundation". He received the Best Paper Award in IEEE Infocom 2017. He is a member of the IEEE.

George Iosifidis received the diploma degree in electronics and telecommunications engineering from the Greek Air Force Academy, in 2000, and the MS and PhD degrees in electrical engineering from the University of Thessaly, Greece, in 2007 and 2012, respectively. He worked as a post-doctoral researcher with CERTH, Greece, and Yale University. He is currently the ussher assistant professor in future networks with Trinity College Dublin, and also a funded investigator with the national research centre CONNECT in Ireland. His research interests lie in the broad area of wireless network optimization and network economics. He is a member of the IEEE.

Antonios Argyriou (S'99-M'06-SM'15) received the diploma degree in electrical and computer engineering from the Democritus University of Thrace, Greece, in 2001, and the MS and PhD degrees in electrical and computer engineering as a Fulbright scholar from the Georgia Institute of Technology, Atlanta, Georgia, in 2003 and 2005, respectively. Currently, he is an assistant professor with the Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece. From 2007 to 2010, he was a senior research scientist at Philips Research, Eindhoven, The Netherlands. From 2004 to 2005, he was a senior engineer with Soft.Networks, Atlanta, Georgia. He currently serves on the editorial board of the *Journal of Communications*. He has also served as guest editor of the *IEEE Transactions on Multimedia* special issue on quality-driven cross-layer design, and he was also a lead guest editor for the *Journal of Communications*, special issue on network coding and applications. He serves on the TPC of several international conferences and workshops in the area of communications, networking, and signal processing. His current research interests include the areas of wireless communication systems and networks, and signal processing. He is a senior member of the IEEE.

Iordanis Koutsopoulos (S'99-M'03-SM'13) received the diploma degree in electrical and computer engineering from the National Technical University of Athens (NTUA), Athens, Greece, in 1997, and the MS and PhD degrees in electrical and computer engineering from the University of Maryland, College Park, College Park, Maryland, in 1999 and 2002, respectively. He is now an associate professor with the Department of Informatics, Athens University of Economics and Business (AUEB), Athens, Greece. He was an assistant professor (2013-2015) with AUEB. Before that, he was an assistant professor (2010-2013) and a lecturer (2005-2010) with the Department of Computer Engineering and Communications, University of Thessaly, Volos, Greece. His research interests include network control and optimization, with applications on wireless networks, social and community networks, crowd-sensing systems, smart-grid, and cloud computing. He was the recipient of the single-investigator European Research Council (ERC) Competition Runner-Up Award for the project RECITAL: Resource Management for Self-coordinated Autonomic Wireless Networks (2012-2015). He is a senior member of the IEEE.

Leandros Tassiulas (S'89-M'91-SM'05-F'07) received the PhD degree in electrical engineering from the University of Maryland, College Park, in 1991. He is the John C. Malone Professor of Electrical Engineering and a member of the Institute for Network Science, Yale University. His research interests include the field of computer and communication networks with emphasis on fundamental mathematical models and algorithms of complex networks, architectures and protocols of wireless systems, sensor networks, novel internet architectures, and experimental platforms for network research. His most notable contributions include the max-weight scheduling algorithm and the back-pressure network control policy, opportunistic scheduling in wireless, the maximum lifetime approach for wireless network energy management, and the consideration of joint access control and antenna transmission management in multiple antenna wireless systems. His research has been recognized by several awards including the IEEE Koji Kobayashi Computer and Communications Award (2016), the inaugural INFOCOM 2007 Achievement Award for Fundamental Contributions to Resource Allocation in Communication Networks, the INFOCOM 1994 Best Paper Award, a National Science Foundation (NSF) Research Initiation Award (1992), an NSF CAREER Award (1995), an Office of Naval Research Young Investigator Award (1997), and a Bodossaki Foundation Award (1999). He has held faculty positions with Polytechnic University, New York, University of Maryland, College Park, and University of Thessaly, Greece. He is a fellow of the IEEE.