Enhancing Software-Defined RAN with Collaborative Caching and Scalable Video Coding

Ruozhou Yu, Shuang Qin, Mehdi Bennis, Xianfu Chen, Gang Feng, Zhu Han, Guoliang Xue

***Abstract*—The ever increasing video demands from mobile users have posed great challenges to cellular networks. To address this issue, video caching in radio access networks (RANs) has been recognized as one of the enabling technologies in future 5G mobile networks, which brings contents near the end-users, reducing the transmission cost of duplicate contents, meanwhile increasing the Quality-of-Experience (QoE) of users. Inspired by the emerging software-defined networking technology, recent proposals have employed centralized collaborative caching among cells to further increase the caching capacity of the RAN. In this paper, we explore a new dimension in video caching in software-defined RANs to expand its capacity. We enable the controller with the capability to adaptively select the bitrates of videos received by users, in order to maximize the number and quality of video requests that can be served, meanwhile minimizing the transmission cost. To achieve this, we further incorporate Scalable Video Coding (SVC), which enables caching and serving sliced video layers that can serve different bitrates. We formulate the problem of joint video caching and scheduling as a reward maximization (cost minimization) problem. Based on the formulation, we further propose a 2-stage rounding-based algorithm to address the problem efficiently. Simulation results show that using SVC with collaborative caching greatly improves the cache capacity and the QoE of users.**

***Keywords*—Software-defined radio access network, collaborative video caching, Scalable Video Coding, 5G mobile networks**

I. Introduction

The mobile traffic in modern cellular networks has undergone a drastic growth in the past decade. Mobile video traffic accounts for more than 50% of the overall mobile data traffic [1]. Such a high volume of video traffic poses a stringent challenge on the capacity of mobile networks. To handle this challenge, researchers have proposed to enhance the radio access networks (RANs) with caching at the network edge [5], [19]. Each base station (BS) is equipped with a cache, which stores video contents based on their popularity. Caching at BSs can enhance users' Quality-of-Experience (QoE) in terms of latency and quality, meanwhile reducing the backhaul usage and energy usage for duplicate transmissions.

One limitation of the above schemes is the limited cache storage at each BS. Motivated by the software-defined RANs (SDRANs) [9], [10], researchers have proposed several collaborative caching schemes [8], [11], [12]. In collaborative caching, a video request can be served using not only the local base station's cache, but also the cached copy at a remote base station via the wireless backhaul. While this requires real-time cache placement and flow scheduling in the network, SDRAN offers such flexibility via dynamic control over the data paths in the RAN.

In this paper, we explore a new dimension in SDRAN video caching to increase capacity. Modern video content providers like YouTube and Netflix commonly offer multiple bitrate versions per video. Lower bitrate version corresponds to lower video quality, but has lower storage and bandwidth requirements. When requesting a video content, a user also specifies the desired bitrate (quality) of the video. However, when the network is congested, it may adaptively lower the bitrate offered to the user, in order to serve as more users.

Traditionally, each bitrate version is offered as a disjoint stream (data file) to the end-user, and the data of one version cannot be reused to serve other versions. Therefore, the RAN needs to cache each bitrate version of the video to serve different bitrates, which leads to high usage of the RAN cache. To this end, we propose to incorporate Scalable Video Coding (SVC) [13], [16], which enables data reuse among different video versions. Using SVC, each video can be sliced into multiple *layers* (a base layer and several enhancement layers), and each layer can be streamed independently.

Using SVC, the RAN can cache videos in better granularity: instead of caching the entire file of each bitrate version, each BS can now opt to cache only a subset of layers of the video, while fetching other layers from other BSs. On the other hand, users requesting different bitrates of a video can reuse the same cached layer of the video, thus reducing the overhead for caching each bitrate version in full. One drawback is the additional overhead for coding. However, it has been shown that the overhead can be bounded below 10% [16].

In this paper, we focus on the joint video caching and scheduling problem in SDRAN, where we seek to maximize the number and quality of videos served, meanwhile minimizing average delay. We first formulate the problem as an integer linear program (ILP) that incorporates both resource (cache and backhaul) constraints and specific layering constraints imposed by SVC. Since this problem is $\mathcal{NP}$-hard (See § III), we further propose a 2-stage polynomial-time algorithm. In the first stage, the algorithm decides the caching of all video layers, then in the second stage it schedules the dissemination of all requests based on the caching decisions. Through simulation experiments, we show that using SVC achieves significant improvement in both RAN capacity and user QoE. To summarize, our contributions are as follows:

- To the best of our knowledge, we are the first to introduce SVC for video caching in SDRAN. More specifically, we

Yu and Xue ({ruozhouy, xue}@asu.edu) are with Arizona State University, Tempe, AZ 85287, USA. Qin and Feng ({blueqs, fenggang}@uestc.edu.cn) are with UESTC, Chengdu 611731, China. Bennis (bennis@ee.oulu.fi) is with University of Oulu, Oulu 90014, Finland. Chen (xianfu.chen@vtt.fi) is with VTT Technical Research Centre of Finland. Han (zhan2@mail.uh.edu) is with University of Houston, Houston, TX 77004, USA. This research was supported in part by NSF grant 1457262 (Yu and Xue), NSFC-61401076 and FRFCU-ZYGX2014J010 (Qin and Feng), TEKES grant 2364/31/2014 (Bennis), TEKES grant 2368/31/2014 (Chen), and NSF grant 1456921 and NSFC-61428101 (Han). The information reported here does not reflect the position or the policy of the federal governments.

combine SVC with collaborative caching to enhance the video serving capacity of RAN and users' QoE.
- The joint video caching and scheduling problem is formulated as an ILP that involves both resource constraints and specific SVC layering constraints.
- We further propose a 2-stage rounding-based polynomial-time algorithm to determine both video caching and scheduling solutions.

The rest of this paper is organized as follows. § II introduces background and related work of RAN caching and SVC. § III describes the network service model and the problem formulation. § IV presents our proposed algorithm in two stages. § V shows the performance evaluation of the proposed method. § VI concludes this paper.

II. Background and Related Work

A. Caching in Mobile Networks

There are three basic schemes for mobile content caching: core network (CN)-based caching [19], [20], RAN-based caching [3], [4], [8], [11], [12], [15], [21], and client-based caching [6], [14], [18]. CN-based caching is currently widely deployed in mobile networks [19]. Woo *et al.* [20] compared several common caching strategies in CN. However, CN is basically one hop away from the users than the RAN, and thus cannot reduce the network traffic at the RAN backhaul. On the other hand, client-based caching is either subject to the very limited cache and transmission capacities on user equipment [14], [18], or beneficial to only the user itself and does not reduce network traffic [6].

Hence most researches focus on content caching in the RAN. To name a few, Ahlehagh *et al.* [3], [4] first proposed two caching policies based on the video popularities within the cell. Ahlehagh *et al.* [15] further added processing capability to each BS, and utilized adaptive bit rate streaming to enhance the caching capacity. The above works focus on local content caching in each cell. Xu *et al.* [21] built a collaborative caching model for RANs to minimize energy consumption. Khreishah *et al.* [11] and Li *et al.* [12] used similar programming models to describe the collaborative caching problem, but solved them using different algorithms. Gharaibeh *et al.* [8] further studied the online problem based on [11] and [12], and proposed an algorithm with a provable competitive ratio.

This paper is most related to [11], [12] and [15]. However, [11] and [12] do not consider the different bitrate versions of videos and regard them as different video files, while [15] does not consider collaboration among BSs. None of them have considered utilizing SVC to increase the RAN capacity and enhance users' QoE, which we study in this paper.

B. Scalable Video Caching

SVC [16] is an extension of the H.264/MPEG-4 AVC video compression standard. A video file encoded using SVC consists of several operation points (OPs), each corresponding to a certain bitrate level of the video. In this paper, we assume that all OPs are linearly dependent, *i.e.*, each higher bitrate OP is dependent on all the lower bitrate OPs. Each OP is obtained by dropping some packets from the higher OP's bitstream. Hence, a video content consists of multiple layers: a base layer and several enhancement layers. The base layer encodes the minimum bitstream required to support the lowest bitrate version. Each enhancement layer includes all missing packets needed to offer a higher bitrate, in other words, all packets dropped from the higher OP's bitstream to obtain the lower OP's bitstream. We leave caching with multi-dimensional SVC (temporal, spatial and fidelity) as our future work.

Specifically, assume a video contains L OPs, and a user requests a bitrate corresponding to OP $l \in \{1, \cdots, L\}$. To fulfill the request, all layers $l' \in \{1, \cdots, l\}$ are needed to form the bitstream of OP l. We call this the *layering constraint* of SVC.

In this paper, we assume that the number of OPs is equal to the number of bitrates that can be requested by the users. We will not further elaborate on SVC in this paper, and refer the reader to [16] for details.

III. Network Model and Problem Statement

In this paper, we study the joint video caching and scheduling problem in an SDRAN. In the SDRAN network, a central controller periodically estimates the maximum concurrent demand of each video in each cell, and runs the video caching and scheduling algorithm to decide the caching of videos and scheduling of bitstreams to serve user requests. The estimation method is not within the scope of this paper. We assume that the estimation accurately reflects the actual request distribution in the RAN. In this section, we describe the network service model, and propose the program formulation of this problem.

A. Network Service Model

An SDRAN consists of BSs $\mathcal{B} = \{B_1, B_2, \cdots, B_M\}$. Each BS B_i has a cache storage of size c_i, and is connected to the CN through a backhaul link, with upstream bandwidth b_i^u and downstream bandwidth b_i^d. To simplify the problem formulation, we assume B_0 represents the Internet CDN, which has unlimited storage and backhaul bandwidth. It thus stores the cached copy of every video that may be requested.

For each pair of BSs B_i (including the Internet) and B_ι, function $d_{i,\iota}$ represents their distance, measured by the transmission delay from B_i to B_ι. We assume that the delay between the Internet and any BS is greater than the delay between any pair of BSs. Therefore, fetching a video from any BS's cache would incur lower delay than from the Internet.

There are in total N videos that may be requested, denoted by $\mathcal{V} = \{V_1, V_2, \cdots, V_N\}$. Using SVC, each video V_j can be sliced into L_j layers $\mathcal{L}_j = \{1, \cdots, L_j\}$, each corresponding to a specific bitrate. Two attributes are associated with each layer $l \in \mathcal{L}_j$: s_j^l denotes the size of layer l, and β_j^l denotes the minimum bandwidth required to transmit the layer without incurring stalling of the playback. Note that due to the layering constraint of SVC, offering bitrate l to user requires the serving of all layers $l' \in \{1, \cdots, l\}$. This also affects the delay incurred by a video request, as it is determined by the maximum delay incurred by all layers serving that request.

Given the set of current users served by the RAN, the controller estimates the number of concurrent user requests for each video content and each bitrate version in each cell. Formally, we use $\psi_\iota^{j,l}$ to denote the estimated number of users at BS B_ι that requests video V_j's bitrate version l or higher,

in other words, the number of users whose request requires layer l based on SVC's layering constraint. For a video layer requested from B_ι and served from B_i, we call B_i the *serving BS*, and B_ι the *home BS* of the request.

B. Problem Statement

Given the information of BSs, videos and estimated video requests, the SDRAN controller decides 1) which video layers are to be cached, 2) where to cache the selected video layers, 3) what is the QoE that users at each BS will get with regard to their requests, and 4) where should all layers for a given request be fetched. These decisions are constrained by the limited resources in the RAN, including limited cache storage and backhaul bandwidth at each BS. The following integer program formulates this problem.

Variables:

- $\mathbf{x} = \{x_i^{j,l}\}$, where $x_i^{j,l} \in \{0,1\}$ is defined for each BS $B_i \in \mathcal{B} \cup \{B_0\}$, video $V_j \in \mathcal{V}$, and layer $l \in \mathcal{L}_j$, denoting whether video V_j's layer l is cached at BS B_i ($x_i^{j,l} = 1$) or not ($x_i^{j,l} = 0$);
- $\mathbf{z} = \{z_{i,\iota}^{j,l}\}$, where $z_{i,\iota}^{j,l} \in \{0, \cdots, \psi_\iota^{j,l}\}$ denotes the number of copies of video V_j's layer l requested by users from BS B_ι that are served from BS B_i's cache;

Objective:

$$\text{maximize} \quad R = \sum_{\iota=1}^{M} \sum_{j=1}^{N} \left(\sum_{l=1}^{L_j} r_{j,l}^b \cdot \sum_{i=0}^{M} z_{i,\iota}^{j,l} - c_j^d \cdot d_\iota^j \right) \quad (1)$$

Constraints:

- **Capacity constraints**:

$$\sum_{j=1}^{N} \sum_{l=1}^{L_j} x_i^{j,l} \cdot s_j^l \leq c_i, \forall B_i \in \mathcal{B} \quad (2)$$

$$\sum_{\iota | \iota \neq i} \sum_{j=1}^{N} \sum_{l=1}^{L_j} z_{i,\iota}^{j,l} \cdot \beta_j^l \leq b_i^u, \forall B_i \in \mathcal{B} \quad (3)$$

$$\sum_{i | i \neq \iota} \sum_{j=1}^{N} \sum_{l=1}^{L_j} z_{i,\iota}^{j,l} \cdot \beta_j^l \leq b_\iota^d, \forall B_\iota \in \mathcal{B} \quad (4)$$

- **Caching & scheduling constraints**:

$$z_{i,\iota}^{j,l} \leq x_i^{j,l} \cdot \psi_\iota^{j,l}, \forall B_i \in \mathcal{B}, B_\iota \in \mathcal{B}, V_j \in \mathcal{V}, l \in [1, L_j] \quad (5)$$

$$\sum_{i=0}^{M} z_{i,\iota}^{j,l} \leq \psi_\iota^{j,l}, \forall B_\iota \in \mathcal{B}, V_j \in \mathcal{V}, l \in [1, L_j] \quad (6)$$

$$\sum_{i=0}^{M} z_{i,\iota}^{j,l} \leq \sum_{i=0}^{M} z_{i,\iota}^{j,l-1}, \forall B_\iota \in \mathcal{B}, V_j \in \mathcal{V}, l \in [2, L_j] \quad (7)$$

- **Delay constraints**:

$$\lambda_{i_p^\iota,\iota}^j = \max_{l \in \mathcal{L}_j} \left\{ \sum_{q=0}^{p} z_{i_q^\iota,\iota}^{j,l} \right\} - \sum_{q=0}^{p-1} \lambda_{i_q^\iota,\iota}^j, \quad \forall B_\iota \in \mathcal{B}, V_j \in \mathcal{V}, p \in \{0, \cdots, M\} \quad (8)$$

$$d_\iota^j = \sum_{i=0}^{M} \lambda_{i,\iota}^j \cdot d_{i,\iota}, \forall B_\iota \in \mathcal{B}, V_j \in \mathcal{V} \quad (9)$$

Explanations:

- Our objective is measured in terms of both the delay incurred by the users at each BS and the number of bitrates (layers) available to the users. Let $r_{j,l}^b$ be the unit reward for serving user with video V_j's bitrate l, and c_j^d be the unit cost for incurring delay for serving video V_j. The objective is to maximize the reward for serving users with high bitrates, while minimizing the incurred delays (defined in (8) and (9)), shown in (1).
- Constraints (2), (3) and (4) enforce resource limits at BS B_i. Constraint (2) states that all cached items at B_i cannot exceed its cache capacity c_i. Constraint (3) states that the total transmission bandwidth of requests targeting users outside B_i cannot exceed its upstream backhaul b_i^u. Similarly, Constraint (4) states that all users fetching video layers from outside its home BS B_ι is bounded by B_ι's downstream backhaul b_ι^d. Note that video layers fetched from the home BS do not consume its backhaul once the layer is cached. Bandwidth consumed for retrieving contents for caching is not considered in this paper.
- Constraint (5) states that a video layer can be fetched from B_i only if B_i caches that layer. Constraint (6) states that the number of served copies of video layers from all BSs should not exceed the number of users requesting that layer, in other words, no video layer is transmitted without being requested by some user. Constraint (7) guarantees that the number of copies of video V_j's layer l served to users at BS B_ι is bounded by the number of served copies of its preceding layer, thus enforcing the SVC layering constraint and avoiding unnecessary serving of higher video layers that cannot turn into higher bitrate versions.
- Constraints (8) and (9) jointly define the aggregate delay d_ι^j received by all users at BS B_ι requesting video V_j. Intermediate variable $\lambda_{i,\iota}^j$ is defined for each video V_j, home BS B_ι and serving BS B_i, denoting the (minimum) number of users requesting V_j at B_ι that incur the delay of $d_{i,\iota}$, *i.e.*, fetching their farthest layer from BS B_i. To define $\lambda_{i,\iota}^j$, we further derive a sorted list for each B_ι consisting of all potential serving BSs $B_i \in \mathcal{B} \cup \{B_0\}$, in *descending* order of their distance $d_{i,\iota}$, and denote the list as $\mathcal{B}_\iota = (B_{i_0^\iota}, B_{i_2^\iota}, \cdots, B_{i_M^\iota})$. Note that $i_M^\iota = \iota$, as the nearest potential serving BS to a cell (user) is always itself (the home BS). Given $\mathcal{B}_\iota$, Constraint (8) inductively defines $\lambda_{i,\iota}^j$ for each B_i along the sorted list $\mathcal{B}_\iota$. Specifically, $\lambda_{i_p^\iota,\iota}^j$ is defined as the number of users incurring *no less* delay than $d_{i_p^\iota,\iota}$, taken as the maximum number of users fetching any layer from $B_{i_p^\iota}$ or farther BSs (first term), subtracting the number of users incurring delay *higher than* $d_{i_p^\iota,\iota}$, taken as the sum of users incurring the delay from farther serving BSs $B_{i_q^\iota}$ (second term).
- Given variables $\lambda_{i,\iota}^j$, the delay incurred by users requesting V_j at B_ι is then defined as the sum of delays incurred from all serving BSs, as shown in Constraint (9).

Note that the above formulation has one non-linear constraint: the delay constraint (8). However, the following theorem transforms this non-linear constraint into a set of linear constraints.

Theorem 3.1: Given the above program, for any $B_\iota \in \mathcal{B}$, $V_j \in \mathcal{V}$ and $p \in \{0, \cdots, M\}$, Constraint (8) is equivalent to the

following set of constraints:

$$\sum_{q=0}^{p} \lambda^{j}_{i^{\iota}_{q},\iota} \geq \sum_{q=0}^{p} z^{j,l}_{i^{\iota}_{q},\iota}, \qquad \text{for } \forall l \in \mathcal{L}_j \tag{10}$$

Proof: Assume the optimal solution of the original program is S^* with objective v^*, and that of the program after replacing (8) with (10) is S with objective v. We prove that $v = v^*$. First of all, $v \geq v^*$, because Inequality (10) includes Equation (8) and thus the solution space is enlarged by the replacement. Assume (to prove by contradiction) that $v > v^*$. Let $\Delta^{j}_{p,\iota} = \sum_{q=0}^{p} \lambda^{j}_{i^{\iota}_{q},\iota} - \max_{l \in \mathcal{L}_j} \{\sum_{q=0}^{p} z^{j,l}_{i^{\iota}_{q},\iota}\}$, and thus Constraint (8) can be rewritten as $\Delta^{j}_{p,\iota} = 0$, and Constraint (10) as $\Delta^{j}_{p,\iota} \geq 0$. According to (8) and (10), $v \neq v^*$ can happen only when $\Delta^{j}_{p,\iota} > 0$ for some ι, j and p; otherwise the resulting solution S is also a feasible solution for the original program, meaning that $v \leq v^*$ which contradicts our assumption. If so, however, we can construct a new solution S' with objective v', by reducing $\lambda^{j}_{i^{\iota}_{p},\iota}$ and increasing $\lambda^{j}_{i^{\iota}_{p+1},\iota}$ (if $p+1 \leq M$) by $\Delta^{j}_{p,\iota}$ simultaneously. This does not violate Constraint (10) for ι, j and p based on the definition of $\Delta^{j}_{p,\iota}$. For any $p_1 > p$ (if exists), $\Delta^{j}_{p_1,\iota}$ does not change due to the equal decrease and increase in $\lambda^{j}_{i^{\iota}_{p},\iota}$ and $\lambda^{j}_{i^{\iota}_{p+1},\iota}$ respectively. No other variable is affected, hence S' is still a feasible solution of the program after replacement. And since we have $d_{i^{\iota}_{p},\iota} \geq d_{i^{\iota}_{p+1},\iota}$ according to the sorting of $\mathcal{B}_\iota$, the value of d^{ι}_{j} will not increase between S and S' in Constraint (9), hence $v \leq v'$. Note that after this construction, in S' we have $\Delta^{j}_{p,\iota} = 0$. We apply this construction for any $\Delta^{j}_{p,\iota} > 0$, in increasing order of p as in $\mathcal{B}_\iota$, until $\forall \Delta^{j}_{p,\iota} = 0$. The final S' is now a feasible solution of the original program, and hence we have $v \leq v' \leq v^*$. This contradicts our assumption that $v > v^*$, and thus completes the proof. ∎

By substituting Constraint (8) with Constraint (10), the resulting program is an integer linear program (ILP).

The above problem is $\mathcal{NP}$-hard. A simple reduction is from the *Multiple Unbounded Knapsack* problem, which remains $\mathcal{NP}$-complete even when the number of knapsacks is one [7]. Due to the page limit, in this paper we focus on designing an efficient algorithm for the problem rather than the detailed $\mathcal{NP}$-hardness proof. In the next section, we propose our algorithm.

IV. Coordinated Video Caching and Scheduling

Given the hardness of the problem, we seek to find an effective and efficient algorithm that approaches the optimal solution in polynomial time. In this section, we propose a 2-stage rounding-based algorithm for the problem. Our proposed algorithm utilizes the relaxation of the integer program formulation. In the first stage, the relaxation is used to make *effective* video caching decisions based on the user request information. Given the result of the first stage, the algorithm then fixes the value of every caching variable in the program, and uses the updated program to decide the scheduling of user requests.

A. Rounding-based Caching

The first stage is to decide the caching of videos, in other words, to decide the value of $x^{j,l}_i$ for any $B_i \in \mathcal{B}$, $V_j \in \mathcal{V}$ and $l \in \mathcal{L}_j$. To achieve this, we rely on the relaxed linear program (LP) of the above integer program formulation, where variables $x^{j,l}_i$ and $z^{j,l}_{i,\iota}$ can take continuous values within their ranges. Solving this LP takes only polynomial time. Based on the resulting (fractional) values of the variables, we further employ the rounding technique to obtain an integral and feasible solution for video caching. The algorithm is shown in Algorithm 1.

Algorithm 1: Caching of video layers at BSs

Input: BSs $\mathcal{B}$, videos $\mathcal{V}$, and estimated requests $\psi^{j,l}_i$
Output: Caching decisions $\dot{\mathbf{x}} = \{\dot{x}^{j,l}_i\}$
1 Initialize $\dot{x}^{j,l}_i = 0$ for $\forall B_i \in \mathcal{B}, V_j \in \mathcal{V}, l \in \mathcal{L}_j$, and $\dot{x}^{j,l}_0 = 1$ for $\forall V_j \in \mathcal{V}, l \in \mathcal{L}_j$;
2 Form relaxed LP $\mathcal{L}$ and solve using standard method;
3 **for** $\forall B_i \in \mathcal{B}, V_j \in \mathcal{V}, l \in \{1, \cdots, L_j\}$ **do**
4 $\quad \rho^{j,l}_i \leftarrow \sum_{\iota=1}^{M} \left(\sum_{l'=l}^{L_j} r^{b}_{j,l'} \cdot z^{j,l'}_{i,\iota} - c^{d}_{j} \cdot d_{i,\iota} \cdot z^{j,l}_{i,\iota} \right)$;
5 **end**
6 Initialize $\bar{c}_i = c_i$ for each $B_i \in \mathcal{B}$;
7 Sort all variables $x^{j,l}_i$ in descending order of $\rho^{j,l}_i$;
8 **for** *each* $x^{j,l}_i$ *in sorted order* **do**
9 $\quad$ **if** $\bar{c}_i \geq s^{l}_{j}$ **then**
10 $\quad\quad \dot{x}^{j,l}_i \leftarrow 1$;
11 $\quad\quad \bar{c}_i \leftarrow \bar{c}_i - s^{l}_{j}$;
12 $\quad$ **end**
13 **end**
14 **return** $\dot{\mathbf{x}}$

The algorithm starts from solving the relaxed program in polynomial time, which obtains the (fractional) values of variables $\mathbf{x}$ and $\mathbf{z}$. After that, all caching variables are sorted based on their *effectiveness* values, denoted by $\rho^{j,l}_i$. The effectiveness value measures how the caching of video V_j's layer l at BS B_i can serve to increase bitrate rewards of users while minimize delay costs. It is defined as the sum reward from all (tentatively) served copies of either layer l or higher layers l' (which are dependent on layer l due to the layering constraint) of video V_j, minus the possible delay penalty incurred for those copies, as shown in Line 4. The higher value a caching decision's effectiveness is, the better it contributes to our objective. Hence, we sort all variables $x^{j,l}_i$ in descending order of $\rho^{j,l}_i$, and assign $\dot{x}^{j,l}_i = 1$ as long as the caching capacity constraint is satisfied in Line 9, and 0 otherwise. Continue this until all variables are assigned, and we get a feasible (integral) caching solution.

B. Video Scheduling

The second stage algorithm takes the result of Algorithm 1 as input, and produces a feasible scheduling solution that serves users with as high bitrate versions and as low delay as possible. The algorithm is shown in Algorithm 2.

Like in the previous stage, Algorithm 2 first solves the relaxed program, but with fixed caching decisions $\dot{\mathbf{x}}$ as input, and obtains the (fractional) values of variables $\mathbf{z}$. Note that we re-solve the program rather than directly taking the solution from Algorithm 1, because the rounded caching solution may affect the second stage. The resulting (fractional) solution is

Algorithm 2: Video scheduling through backhaul

Input: BSs $\mathcal{B}$, videos $\mathcal{V}$, estimated requests $\psi_i^{j,l}$, and caching decisions $\dot{\mathbf{x}} = \{\dot{x}_i^{j,l}\}$

Output: Scheduling decisions $\dot{\mathbf{z}} = \{\dot{z}_{i,\iota}^{j,l}\}$

1 Initialize $\dot{z}_{i,\iota}^{j,l} = 0$ for $\forall B_i \in \mathcal{B} \cup \{B_0\}$, $B_\iota \in \mathcal{B}$, $V_j \in \mathcal{V}, l \in \mathcal{L}_j$;

2 Form relaxed LP $\mathcal{L}$, and set $x_i^{j,l} = \dot{x}_i^{j,l}$ for every $x_i^{j,l}$, then solve $\mathcal{L}$ using standard method;

3 Round $\dot{z}_{i,\iota}^{j,l} = \lfloor z_{i,\iota}^{j,l} \rfloor$ to obtain a basic solution;

4 Reduce $\dot{z}_{i,\iota}^{j,l}$ which violates the layering constraint (7) for each B_ι and V_j, from $i = i_0^\iota$ to i_M^ι and $l = 2$ to L_j;

5 Let $\overline{b}_i^u$ and $\overline{b}_i^d$ be the residual upstream and downstream backhaul at B_i respectively;

6 Compute $\xi_\iota^{j,l} = \sum_{i=0}^{M} \dot{z}_{i,\iota}^{j,l}$ for all triples (B_ι, V_j, l);

7 **for** $\xi_\iota^{j,l} < \min\{|\psi_\iota^{j,l}|, \xi_\iota^{j,l-1}\}$ *increasingly of* l **do**

8 **for** $B_i \in \mathcal{B}_\iota$ *in reversed order such that* $\dot{x}_i^{j,l} = 1$ **do**

9 **if** $\min\{\overline{b}_i^u, \overline{b}_\iota^d\} \geq \beta_j^l$ **then**

10 $\alpha \leftarrow \min\{|\psi_\iota^{j,l}|, \xi_\iota^{j,l-1}\} - \xi_\iota^{j,l}$;

11 $\beta \leftarrow \min\{\overline{b}_i^u, \overline{b}_\iota^d\}/\beta_j^l$;

12 $\dot{z}_{i,\iota}^{j,l} \leftarrow \dot{z}_{i,\iota}^{j,l} + \min\{\alpha, \beta\}$;

13 Update $\xi_\iota^{j,l}$, $\overline{b}_i^u$ and $\overline{b}_i^d$ accordingly;

14 **end**

15 **end**

16 **end**

17 **return** $\dot{\mathbf{z}}$

then rounded *down* to obtain a basic scheduling solution. Note that during the rounding, the layering constraint (7) may be violated (the rounded fraction of a lower layer is greater than the rounded fraction of the higher layer). It is then enforced by reducing the violating servings, beginning from the second lowest layers and the farthest BSs, in Line 4.

Since all variables are rounded *down*, there may still remain resources aggregated from the rounded fractions that can serve more user requests. To utilize these resources, we further iterate on all videos from the lowest to the highest layers, and attempt to serve more requests at each layer (Lines 7–16). Note that BSs in $\mathcal{B}_\iota$ are sorted in descending order of delay to B_ι, hence the list is traversed in reversed order, in order to seek the minimum delay given home BS B_ι. The residual resources are updated after obtaining the basic solution and scheduling any further layers not served in the basic solution. In the final solution, either all requests have been served up to the desired bitrate, or no layer can be further offered given the caching solution and the residual backhaul.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed method. The performance is evaluated in two dimensions: RAN capacity and user QoE. The capacity of the RAN is measured by the *number of active users* that can be served by the system, *i.e.*, the number of users receiving at least the lowest bitrate version of its requested video. The users' QoE is measured by both the *average bitrate* received each user, and the *average transmission delay* incurred by each user. Note that both bitrate and delay are incorporated in our objective function in (1). The number of active users is implicitly defined as the number of base layers served for all videos, implying the number of users receiving at least the lowest bitrate version of their requested videos.

TABLE I: Experiment results with default parameters

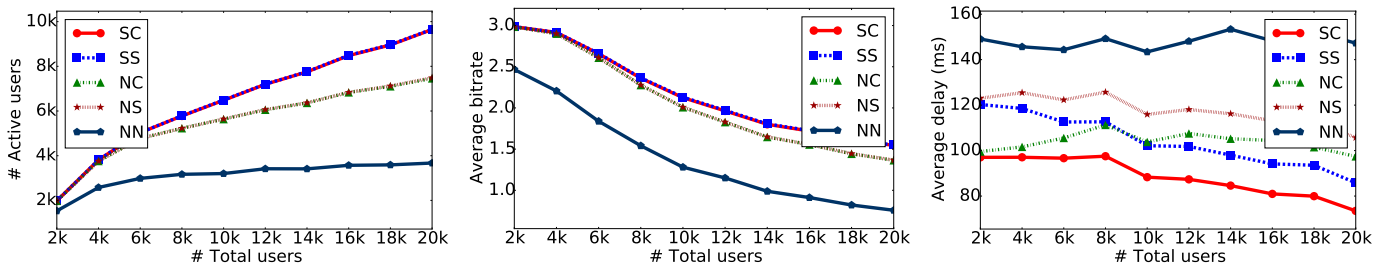
Combination	Act users	Avg layers	Avg delay
SVC + Collaborative (SC)	6480.9	2.1249	88.31
SVC + Non-collaborative (SS)	6492.1	2.1306	102.14
Non-SVC + Collaborative (NC)	5635.2	2.0077	103.80
Non-SVC + Non-collabor. (NS)	5642.4	2.0092	115.90
No caching (NN)	3201.1	1.2808	143.41

We use randomly generated RAN environment for evaluation. By default, the RAN consists of 15 BSs, and 10000 users uniformly distributed among all BSs. Each BS has cache capacity c_i uniformly generated from $[500, 5000]$ (MB), and upstream and downstream backhaul bandwidth b_i^u and b_i^d both uniformly generated from $[100, 1000]$ (Mbps). The delays of fetching video content within local BS's cache, from other BS's cache and from the Internet are uniformly generated from $[5, 10]$ (ms), $[20, 50]$ (ms) and $[100, 200]$ (ms) respectively [12]. The number of available videos is 5000, with popularity following a Zipf distribution with exponent $\gamma = 0.95$, following [11]. Each video has an overall size uniformly generated from $[100, 1000]$ (MB) and overall bandwidth demand from $[0.5, 5]$ (Mbps). Each video has 5 bitrate versions, with equal probability of being requested. Following [15], the five layers each contains $0.45, 0.1, 0.12, 0.15, 0.18$ of packets of the original video respectively. To account for the coding overhead of SVC, each layer imposes an additional 10% overhead on the packets over the previous layer.

We use the proposed algorithm to evaluate different combinations of using SVC and collaborative caching in SDRAN (Table I). The formulation and the algorithm are modified accordingly to adapt to the different combinations. The modifications are trivial, hence we omit the details in this paper. In the objective function, we set the unit bitrate reward as $r_{j,l}^d = 1000$ for each video layer, and unit delay penalty as $c_j^d = 1$ for all videos.

The algorithm is implemented using C++. The Gurobi Optimizer [2] is used to solve the linear programs in the algorithm. All experiments are conducted on a Ubuntu Linux PC, with 3.4GHz Quad-Core CPU and 16GB memory. Each experiment is repeated for 20 times under the same experiment setting, and the results are taken as the average of all runs.

Table I shows the comparison of different combinations of collaborative caching and SVC. As shown, both SVC and collaborative caching can increase the capacity of the RAN and the users' QoE. Compared to Non-SVC, using SVC achieves about 15% increase in the number of active users served and decrease in average delay. Using collaborative caching, significant delay reduction can be achieved, but the number of active users and the average bitrate do not increase. This is because in both collaborative and non-collaborative caching, the bottleneck is basically the downstream backhaul at the home BSs, which limits the number of requests that can be served. The slight decrease in the number of active users or average layers is because our algorithm trades-off a little capacity to achieve delay reduction for users when



(a) # Active users (b) Average # layers per user (c) Average delay per user

Fig. 1: Experiment with increasing user demands.

collaborative caching enables such adjustments.

Fig. 1 shows the comparison of different combinations when facing increasing user demands. Both collaborative and non-collaborative caching with SVC achieve a high number of active users and average layers. The number of active users of the four caching solutions overlap when the user load is small, because all of them can serve almost all user requests. When the user load increases, the gap between using SVC and not using SVC increases, because SVC enhances the RAN capacity and thus can serve more users with more layers when congested. On the other hand, SVC + Collaborative achieves the lowest average delay. Using SVC can decrease delay compared to Non-SVC, because more layers can be fetched from remote BSs rather than from the Internet. Note that the delay decreases as the load increases with all four caching-based solutions, because as the number of requests increases, there are more choices for the algorithm to serve those requests that may incur low delay. All caching-based solutions result in much larger capacity and lower delay than fetching all videos from the Internet (NN).

To summarize, the simulation results show that both SVC and collaborative caching can lead to improved RAN capacity and user QoE. The combination of SVC and collaborative caching achieves the best results compared to using either SVC or collaborative caching alone, or other combinations.

VI. Conclusions

In this paper, we explored how to use SVC and collaborative caching to enhance the video caching capacity of an SDRAN and the user QoE. Using SVC, each video bitrate version is served as a set of layers rather than a single video bitstream. Different layers of a video can be cached at and fetched from caches at different BSs, which enables the flexibility of video caching and scheduling, and avoids duplicate caching of data of the same video. We formulated the joint video caching and scheduling problem in an SDRAN as an integer linear program. Based on the program, we proposed a 2-stage rounding-based algorithm that approaches the optimal solution efficiently. Simulation experiments show that combining SVC with collaborative caching can greatly improve the SDRAN capacity, as well as the QoE received by end-users.

References

[1] Cisco Global Mobile Data Traffic Forecast. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html.

[2] Gurobi Optimizer. http://www.gurobi.com/products/gurobi-optimizer.

[3] H. Ahlehagh and S. Dey. Video caching in radio access network: impact on delay and capacity. In *IEEE WCNC*, pp. 2276–2281, 2012.

[4] H. Ahlehagh and S. Dey. Video-aware scheduling and caching in the radio access network. *IEEE/ACM TON*, 22(5): 1444–1462, 2014.

[5] E. Bastug, M. Bennis, and M. Debbah. Living on the edge: the role of proactive caching in 5G wireless networks. *IEEE Commun. Mag.*, 52(8): 82–89, 2014.

[6] R. Bhatia, T. V. Lakshman, A. Netravali, and K. Sabnani. Improving mobile video streaming with link aware scheduling and client caches. In *IEEE INFOCOM*, pp. 100–108, 2014.

[7] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman & Co., 1990.

[8] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash. A provably efficient online collaborative caching algorithm for multicell-coordinated systems. To appear in *IEEE Trans. Mob. Comput.*, 2015.

[9] A. Gudipati, D. Perry, L. E. Li, S. Katti, and B. Labs. SoftRAN : software defined radio access network. In *HotSDN*, pp. 25–30, 2013.

[10] J. Ding, R. Yu, Y. Zhang, S. Gjessing, and D. Tsang. Service providers competition and cooperation in cloud-based software defined wireless networks. In *IEEE Commun. Mag.*, 53(11): 134–140, 2015.

[11] A. Khreishah and J. Chakareski. Collaborative caching for multicell-coordinated systems. In *INFOCOM WKSHPS*, pp. 257–262, 2015.

[12] X. Li, X. Wang, S. Xiao, and V. C. M. Leung. Delay performance analysis of cooperative cell caching in future mobile networks. In *IEEE ICC*, pp. 5652–5657, 2015.

[13] S. Mao, S. Lin, S. Panwar, Y. Wang, and E. Celebi. Video transport over ad hoc networks: multistream coding with multipath transport. *IEEE JSAC*, 21(10): 1721–1736, 2003.

[14] H. A. Pedersen and S. Dey. Mobile device video caching to improve video QoE and cellular network capacity. In *ACM MSWiM*, pp. 103–107, 2014.

[15] H. A. Pedersen and S. Dey. Enhancing mobile video capacity and quality using rate adaptation, RAN caching and processing. To appear in *IEEE/ACM Trans. Networking*, 2015.

[16] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Trans. Circuits Syst. Video Technol.*, 17(9): 1103–1120, 2007.

[17] I. Stefanovici, E. Thereska, G. O'Shea, B. Schroeder, H. Ballani, T. Karagiannis, A. Rowstron, and T. Talpey. Software-defined caching. Tech. Rep, University of Toronto.

[18] X. Wang, M. Chen, Z. Han, D. O. Wu, and T. T. Kwon. TOSS: Traffic offloading by social network service-based opportunistic sharing in mobile social networks. In *IEEE INFOCOM*, pp. 2346–2354, 2014.

[19] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung. Cache in the air: exploiting content caching and delivery techniques for 5G systems. *IEEE Commun. Mag.*, 52(2): 131–139, 2014.

[20] S. Woo, E. Jeong, S. Park, J. Lee, S. Ihm, and K. Park. Comparison of caching strategies in modern cellular backhaul networks. In *ACM MobiSys*, pp. 319–332, 2013.

[21] Y. Xu, Y. Li, Z. Wang, T. Lin, G. Zhang, and S. Ci. Coordinated caching model for minimizing energy consumption in radio access network. In *IEEE ICC*, pp. 2406–2411, 2014.