

Cache-Version Selection and Content Placement for Adaptive Video Streaming in Wireless Edge Networks

Archana Sasikumar^{*†}, Tao Zhao^{*‡}, I-Hong Hou[‡], and Srinivas Shakkottai[‡]

[†]Juniper Networks

[‡]Dept. of ECE, Texas A&M University, College Station, TX 77843

Email: asasi@juniper.net, {alick,ihou,sshakkot}@tamu.edu

Abstract—Wireless edge networks are promising to provide better video streaming services to mobile users by provisioning computing and storage resources at the edge of wireless network. However, due to the diversity of user interests, user devices, video versions or resolutions, cache sizes, network conditions, etc., it is challenging to decide where to place the video contents, and which cache and video version a mobile user device should select. In this paper, we study the joint optimization of cache-version selection and content placement for adaptive video streaming in wireless edge networks. We propose practical distributed algorithms that operate at each user device and each network cache to maximize the overall network utility. In addition to proving the optimality of our algorithms, we implement our algorithms as well as several baseline algorithms on ndnSIM, an ns-3 based Named Data Networking simulator. Simulation evaluations demonstrate that our algorithms significantly outperform conventional heuristic solutions.

I. INTRODUCTION

Video streaming has become the dominant application for modern Internet traffic. In order to provide better quality of service (QoS) and quality of experience (QoE) to mobile users, content delivery networks (CDNs) have been deployed to store popular videos at cache servers close to the users. This aligns with the trend of wireless edge networks, where computing and storage resources are provisioned at the edge of the wireless network [1]. Meanwhile, as users are accessing videos from a variety of devices, ranging from smartphones to 4K televisions (TVs), adaptive video streaming, which encodes the same video content into multiple versions with different resolutions, has been widely used to deliver arguably the best video version to each user based on device types and network conditions.

In this paper, we study the interplay between three important components for adaptive video streaming in wireless edge networks: *cache selection*, where each user device determines which cache server to retrieve videos from, *version selection*, which determines the version that each user watches, and *content placement*, which entails the caching strategy of each

cache server. We formulate CaVe-CoP, a Cache-Version selection and Content Placement problem that jointly optimizes these three components by taking into account the preferred video versions of users, the communication capacities of network links, and the storage capacities of cache servers. Our goal is to develop a new network algorithm for CaVe-CoP that is not only provably optimal, but also practical and implementable.

Our proposed solution is based on the observation that there is a practical timescale separation between cache-version selection (CaVe) and content placement (CoP), as the former can be updated much more frequently. Hence, we first solve the CaVe problem by fixing the solution to the CoP problem, and prove the optimality of our CaVe algorithms. We then solve the CoP problem by considering its influence to solution to the CaVe problem, and prove our CoP algorithms are optimal when fractional solutions are allowed.

While our algorithms can be practically implemented under the current Internet architecture with TCP/IP, we demonstrate that our algorithms can also be implemented in a distributed fashion on Named Data Networking (NDN) [2], a future Internet architecture designed with video streaming applications in mind. Since NDN forwards packets by content names instead of location IDs such as IP addresses, we present a distributed forwarding strategy that ensures user devices always obtain their selected video versions from their selected cache server. Moreover, we show that the overhead of our algorithms is negligible by exploiting local information and built-in caching. We evaluate our algorithms on ndnSIM [3], an ns-3 based NDN simulator. Simulation results depict that our algorithms significantly outperform baseline policies that employ conventional heuristic solutions and subsets of our algorithms.

The rest of the paper is organized as follows. Section II introduces our system model and the formulation of CaVe-CoP. Solutions to the two problems CaVe and CoP are introduced in Section III and IV, respectively. In Section V, we discuss the implementation of our algorithms in NDN. Section VI demonstrates the simulation results. Section VII reviews some related literature. Finally, Section VIII concludes the paper.

^{*}These authors contributed equally to this work.

A. Sasikumar was with Texas A&M University when conducting this work.

This research was supported in part by grants NSF CNS 1149458, AST 1443891, NSF-Intel CNS 1719384, ARO W911NF-18-1-0331, and ONR N00014-18-1-2048.

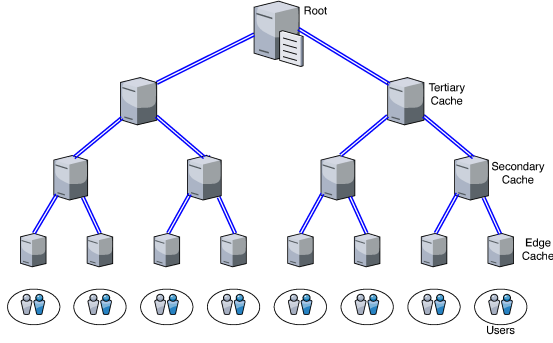


Fig. 1. A wireless edge network with a root node holding all videos and three layers of caches. Each edge cache serves a group of users with different user devices.

II. SYSTEM MODEL

We consider a wireless edge network where a group of network caches jointly host a set of videos and serve a set of video streaming users.¹ Fig. 1 illustrates an example of such a network, which is consistent with the YouTube video delivery system [4]. We use $\mathbb{C}$ to denote the set of network caches, $\mathbb{S}$ to denote the set of users, and $\mathbb{L}$ to denote the set of communication links that connect the network caches, routers, and users. We assume there is a route² between each user s and each network cache c , and define $H_{s,c}^l$ as the indicator function that link l is on the route between s and c .

We consider multi-version video streaming where each video is encoded into multiple versions for different resolutions of the same video content. We use $\mathbb{V}$ to denote the set of all versions of all videos. For each video version $v \in \mathbb{V}$, we use X_v to denote the average bit rate of v and Y_v to denote the file size of v , i.e. the product of X_v and the duration of the video. For the ease of theoretical analysis, we assume that there exists a *null version* v_0 with $X_{v_0} = 0$ and $Y_{v_0} = 0$. If a user decides not to watch any video, then we say that the user watches the null version v_0 . With the introduction of the null version, we can assume that each user always watches a video version.

Each network cache $c \in \mathbb{C}$ has a storage of size B_c to store some video versions. Specifically, let $p_{c,v}$ be the indicator function that v is present in the storage of c , then we have $\sum_v Y_v p_{c,v} \leq B_c$, for all c . Each network cache c determines which video versions to store, and thereby determines the values of $p_{c,v}$, subject to its storage constraint. We assume that there exists at least a network cache c with infinite storage $B_c = \infty$ and stores all video versions, which we call the root node. Such an assumption is to ensure that at least one copy of each video version exists in the network. We refer to the problem of determining $p_{c,v}$ as the *content placement (CoP) problem*.

At the user end, each user s is interested in watching a video. Let $\mathbb{I}_s$ be the set of video versions that correspond to

the interested video of user s . Each user device s needs to determine which video version to watch, as well as which network cache to obtain the video version from. Let $z_{s,c,v}$ be the indicator function that user s decides to watch video version v , and to obtain it from network cache c . We refer to the problem of determining $z_{s,c,v}$ as the *cache-version selection (CaVe) problem*. Since user s needs to obtain exactly one video version, we require that $\sum_{c,v \in \mathbb{I}_s} z_{s,c,v} = 1$, for all s . Moreover, user s can only obtain video version v from network cache c if c indeed stores v , that is, $p_{c,v} = 1$. Hence, we also need $z_{s,c,v} \leq p_{c,v}$, for all s, c, v .

Recall that the bit rate of video version v is X_v and $H_{s,c}^l = 1$ if link l is on the route between s and c . When user s obtains v from c , it incurs an amount of X_v traffic on each link along the route between s and c . The total amount of traffic on link l can then be expressed as $\sum_{s,c,v} X_v H_{s,c}^l z_{s,c,v}$. We consider that each link l has a finite capacity of R_l , and hence we require that $\sum_{s,c,v} X_v H_{s,c}^l z_{s,c,v} \leq R_l$, for all $l \in \mathbb{L}$.

Finally, each user obtains some utility based on its perceived video quality. In particular, we consider that each user s obtains a utility of $U_s(X_v)$ when watching a video version with bit rate X_v . We assume that $U_s(\cdot)$ is a non-decreasing and concave function. Different users may have different utility functions since they may be watching videos on different types of devices. For example, users watching videos on smartphones typically enjoy lower utility than those watching videos on TVs.

We aim to maximize the total utility of all users in the network by choosing the optimal $\mathbf{p} := [p_{c,v}]$ and $\mathbf{z} := [z_{s,c,v}]$, subject to all aforementioned constraints. Formally, we have the following CaVe-CoP optimization problem.³

CaVe-CoP

$$\max \sum_{s,c,v \in \mathbb{I}_s} U_s(X_v) z_{s,c,v} \quad (1a)$$

$$\text{s.t.} \quad \sum_v Y_v p_{c,v} \leq B_c, \quad \forall c \in \mathbb{C}, \quad (1b)$$

$$\sum_{c,v \in \mathbb{I}_s} z_{s,c,v} = 1, \quad \forall s \in \mathbb{S}, \quad (1c)$$

$$z_{s,c,v} \leq p_{c,v}, \quad \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}, \quad (1d)$$

$$\sum_{s,c,v} X_v H_{s,c}^l z_{s,c,v} \leq R_l, \quad \forall l \in \mathbb{L}, \quad (1e)$$

$$p_{c,v} \in \{0, 1\}, z_{s,c,v} \in \{0, 1\}, \quad \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}. \quad (1f)$$

While the utility maximization problem studied in this paper may look similar to many existing studies on network utility maximization (NUM), we note that there are two major challenges that distinguish our problem from other NUM problems: First, most existing studies on NUM problems assume that the source and destination of each flow is fixed and given. In contrast, multiple network caches may store

¹The terms “user” and “user device” are used interchangeably.

²Our model and algorithms can be generalized to the multi-route scenario.

³In practice the CaVe-CoP problem will be solved repeatedly over time with different parameters to cope with network changes.

the same video version depending on the solution to the content placement problem. Hence, not only does a user have multiple choices of network caches to obtain the video version from, but the problem of selecting cache is fundamentally intertwined with the problem of content placement. Second, although the problem of version selection may seem to be a special case of the rate control problem, we note that the problem of version selection is fundamentally intertwined with the problem of selecting cache since each cache may only store a subset of versions for a given video. The possibility of placing different versions of the same video at different caches also distinguishes this work from some recent studies on throughput-optimal algorithms with caches. Araldo *et al.* [5] studied a similar problem to ours. However, they only derived heuristics without meaningful performance guarantees.

The decision variables in CaVe-CoP are $\mathbf{p}$ and $\mathbf{z}$. We note that there is a practical timescale separation between the update for $\mathbf{p}$ and that for $\mathbf{z}$. When a user device changes its values for $\mathbf{z}$ due to e.g. network congestion, it simply requests new packets from a different network cache and/or with a different video version. Hence, $\mathbf{z}$ can be updated rather frequently, for example, once every 100 milliseconds. On the other hand, when a network cache changes its values for $p_{c,v}$, it needs to obtain all video versions with $p_{c,v} = 1$. Hence, $\mathbf{p}$ can only be updated infrequently.

Our proposed solution for CaVe-CoP is based on the observation of the timescale separation between the update for $\mathbf{p}$ and that for $\mathbf{z}$. In Section III, we will first consider the CaVe problem by finding the optimal $\mathbf{z}$ for given $\mathbf{p}$. Next, in Section IV, we will consider the CoP problem. In order to find the optimal $\mathbf{p}$, we will introduce pseudo-variables $\mathbf{z}' := [z'_{s,c,v}]$ and $\mathbf{p}' := [p'_{c,v}]$ that are updated at the same frequency as $\mathbf{p}$ to address the issue with timescale separation.

Finally, we note that CaVe-CoP is an integer programming problem since $p_{c,v}$ and $z_{s,c,v}$ are integers. To obtain tractable results, we will relax (1f) and allow $p_{c,v}$ and $z_{s,c,v}$ to be any real number between 0 and 1. As we will demonstrate in Section III, our solution to the CaVe problem will always yield integer values for $z_{s,c,v}$. We will also discuss how to obtain integer solutions for $p_{c,v}$ in Section IV.

III. THE CACHE-VERSION SELECTION PROBLEM (CAVE)

In this section, we study the CaVe problem. We consider that the contents that each network cache store are given and fixed, and aims to determine both the video version to watch and the network cache to obtain contents from for each user. In terms of the optimization problem (1a)–(1f), we focus on finding the optimal $\mathbf{z} := [z_{s,c,v}]$ to maximize total utility in the network when $\mathbf{p} := [p_{c,v}]$ is given and fixed.

A. Overview of the Solution

We begin by rewriting the optimization problem (1a)–(1f) for the CaVe problem. Since $\mathbf{p}$ is given and fixed, constraint (1b) no longer applies. Further, we relax the constraint (1f) by allowing $z_{s,c,v}$ to be any real number between 0 and 1. The

resulting optimization problem, which we call CaVe-Primal, can then be described as follows:

CaVe-Primal

$$\max \sum_{s,c,v \in \mathbb{I}_s} U_s(X_v) z_{s,c,v} \quad (2a)$$

$$\text{s.t.} \quad \sum_{c,v \in \mathbb{I}_s} z_{s,c,v} = 1, \quad \forall s \in \mathbb{S}, \quad (2b)$$

$$z_{s,c,v} \leq p_{c,v}, \quad \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}, \quad (2c)$$

$$\sum_{s,c,v} X_v H_{s,c}^l z_{s,c,v} \leq R_l, \quad \forall l \in \mathbb{L}, \quad (2d)$$

$$0 \leq z_{s,c,v} \leq 1, \quad \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}. \quad (2e)$$

We will consider a dual problem to CaVe-Primal. We associate a Lagrange multiplier, λ_l , for each link capacity constraint (2d), for all $l \in \mathbb{L}$. Let $\boldsymbol{\lambda} := [\lambda_l]$ be the vector of Lagrange multipliers. The Lagrangian is obtained as follows:

$$\begin{aligned} L(\mathbf{z}, \boldsymbol{\lambda}) \\ := \sum_{s,c,v \in \mathbb{I}_s} U_s(X_v) z_{s,c,v} - \sum_l \lambda_l \left(\sum_{s,c,v} z_{s,c,v} H_{s,c}^l X_v - R_l \right) \end{aligned} \quad (3)$$

The dual objective, $D(\boldsymbol{\lambda})$, is defined as the maximum value of $L(\mathbf{z}, \boldsymbol{\lambda})$ over $\mathbf{z}$ subject to the constraints (2b), (2c), and (2e). We call the underlying optimization problem CaVe-Lagrangian. It can be written as follows:

CaVe-Lagrangian

$$\max L(\mathbf{z}, \boldsymbol{\lambda}) \quad (4a)$$

$$\text{s.t.} \quad \sum_{c,v \in \mathbb{I}_s} z_{s,c,v} = 1, \quad \forall s \in \mathbb{S}, \quad (4b)$$

$$z_{s,c,v} \leq p_{c,v}, \quad \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}, \quad (4c)$$

$$0 \leq z_{s,c,v} \leq 1, \quad \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}. \quad (4d)$$

Remark 1: In defining the CaVe-Lagrangian problem, we only relax the link capacity constraint (2d), and keep other constraints (2b), (2c) and (2e) intact. This is because the link capacity constraint (2d) can be temporarily violated as packets that cannot be served immediately can be queued in the buffer. On the other hand, constraints (2b) and (2c) need to be satisfied at all time in practical systems.

The dual problem is to minimize $D(\boldsymbol{\lambda})$ while ensuring that all Lagrange multipliers λ_l are non-negative. We call this the CaVe-Dual and mathematically write it as:

CaVe-Dual

$$\min D(\boldsymbol{\lambda}) \quad (5a)$$

$$\text{s.t.} \quad \lambda_l \geq 0, \quad \forall \lambda_l \in \mathbb{L}. \quad (5b)$$

Theorem 1 (Strong Duality): CaVe-Primal and CaVe-Dual have the same optimal value.

Proof: See [6]. ■

Based on Theorem 1, we can solve the CaVe-Primal problem by solving CaVe-Dual. Solving CaVe-Dual involves two steps: First, for a given vector λ , we need to find $D(\lambda)$ by solving CaVe-Lagrangian. Second, we need to find the optimal λ to solve CaVe-Dual. We introduce our solutions to these two steps below.

B. The Solution to CaVe-Lagrangian

We rewrite (3) as:

$$\begin{aligned} L(z, \lambda) &= \sum_{s, c, v \in \mathbb{I}_s} U_s(X_v) z_{s, c, v} - \sum_l \lambda_l \left(\sum_{s, c, v} z_{s, c, v} H_{s, c}^l X_v - R_l \right) \\ &= \sum_s \sum_{c, v \in \mathbb{I}_s} z_{s, c, v} (U_s(X_v) - X_v \sum_{l: H_{s, c}^l = 1} \lambda_l) + \sum_l \lambda_l R_l \end{aligned} \quad (6)$$

We note that the above expression provides a natural user-by-user decomposition. Specifically, by defining z_s as the vector containing all $[z_{s, c, v}]$ for a given s , and defining

$$L_s(z_s, \lambda) := \sum_{c, v \in \mathbb{I}_s} z_{s, c, v} (U_s(X_v) - X_v \sum_{l: H_{s, c}^l = 1} \lambda_l), \quad (7)$$

we have

$$L(z, \lambda) = \sum_s L_s(z_s, \lambda) + \sum_l \lambda_l R_l. \quad (8)$$

As λ is given in CaVe-Lagrangian, the last term $\sum_l \lambda_l R_l$ is a constant. Hence, $L(z, \lambda)$ is maximized if one can maximize $L_s(z_s, \lambda)$ for each user s . Moreover, recall that $p_{c, v}$ is the indicator function that network cache c stores video version v . Therefore, the constraint (4c) is equivalent to saying that $z_{s, c, v}$ needs to be 0 if $p_{c, v} = 0$. We can now define CaVe-User_s as follows:

CaVe-User_s

$$\max \sum_{c, v: v \in \mathbb{I}_s, p_{c, v} = 1} z_{s, c, v} (U_s(X_v) - X_v \sum_{l: H_{s, c}^l = 1} \lambda_l) \quad (9a)$$

$$\text{s.t.} \sum_{c, v: v \in \mathbb{I}_s, p_{c, v} = 1} z_{s, c, v} = 1, \quad (9b)$$

$$0 \leq z_{s, c, v} \leq 1, \quad \forall c \in \mathbb{C}, v \in \mathbb{V}. \quad (9c)$$

It is clear that the optimal vector z that solves CaVe-User_s, for all s , is also the optimal vector that solves CaVe-Lagrangian. To solve CaVe-User_s, note that the only decision variable in CaVe-User_s is the vector z_s , while $U_s(X_v)$, X_v , and λ_l are all constants. Hence, the following algorithm solves CaVe-User_s: First, find (c^*, v^*) that has the maximum value of $U_s(X_v) - X_v \sum_{l: H_{s, c}^l = 1} \lambda_l$ among all (c, v) with $v \in \mathbb{I}_s$ and $p_{c, v} = 1$. Ties can be broken arbitrarily. Second, set $z_{s, c^*, v^*} = 1$, and $z_{s, c, v} = 0$ for all other (c, v) . Alg. 1 summarizes the algorithm. We note that, even though we have relaxed the constraint and allowed $z_{s, c, v}$ to be any real number between 0 and 1, the optimal solution produced by Alg. 1 is always an integer one. Besides, note that c^* and v^* are

updated iteratively as λ is updated. It means the cache-version selection of each user is dynamic and adaptive to the network congestion.

Algorithm 1 CaVe-User_s Algorithm

Obtain p and λ

$z_{s, c, v} \leftarrow 0, \forall c, v$

$(c^*, v^*) \leftarrow \arg\max_{c, v \in \mathbb{I}_s: p_{c, v} = 1} U_s(X_v) - X_v \sum_{l: H_{s, c}^l = 1} \lambda_l$

$z_{s, c^*, v^*} \leftarrow 1$

C. The Solution to CaVe-Dual

Our solution to CaVe-Dual is shown in Alg. 2, where each link l updates its own λ_l . We have the following lemma and theorem.

Algorithm 2 CaVe-Link_l Algorithm

$t \leftarrow 0, \lambda_l \leftarrow 0$

while true do

Obtain z from Alg. 1

$\lambda_l \leftarrow \left[\lambda_l + h_t (\sum_{s, c, v} X_v H_{s, c}^l z_{s, c, v} - R_l) \right]^+$

$t \leftarrow t + 1$

Lemma 1: Given λ , let z^* be the vector that solves CaVe-User_s. Then $g := [g_l] := [R_l - \sum_{s, c, v} X_v H_{s, c}^l z_{s, c, v}^*]$ is a subgradient of $D(\lambda)$.

Proof: See [6]. ■

Theorem 2: Let $\{h_t\}$ be a sequence of non-negative numbers with $\sum_{t=0}^{\infty} h_t = \infty$ and $\lim_{t \rightarrow \infty} h_t = 0$, then Alg. 2 solves CaVe-Dual.

Proof: See [6]. ■

D. The Solution to CaVe-Primal

We have the following theorem regarding the optimality of z obtained by running Alg. 1 and Alg. 2 iteratively:

Theorem 3: Let z^* be the vector that solves CaVe-Primal, λ^k be the vector produced by Alg. 2 after k iterations, and z^k be the vector produced by Alg. 1 when $\lambda = \lambda^k$. Let $\bar{z}^t$ be the weighted average of z^k after the first t iterations, i.e. $\bar{z}^t := \lim_{T \rightarrow \infty} \frac{\sum_{k=t+1}^{t+T} h_k z^k}{\sum_{k=t+1}^{t+T} h_k}$. Then, for any $\varepsilon > 0$, there exists an integer K such that for every $t > K$,

- 1) $\bar{z}^t$ satisfies all CaVe-Primal constraints;
- 2) $\sum_{s, c, v} U_s(X_v) z_{s, c, v}^* - \sum_{s, c, v} U_s(X_v) \bar{z}_{s, c, v}^t \leq \varepsilon$.

Proof: See [6]. ■

IV. THE CONTENT PLACEMENT PROBLEM (COP)

We now discuss the content placement (CoP) problem, which entails deciding $p_{c, v}$, the indicator function that network cache c stores video version v , for all c and v . As discussed in Section II, a major challenge to our optimization problem (1a)–(1f) is that the vector p needs to be updated much less frequently than the vector z . To address this challenge, we introduce pseudo-variables $z' := [z'_{s, c, v}]$ and $p' := [p'_{s, c, v}]$, which can be updated much more frequently than p , to replace

z and p .⁴ We only update p , the real content placement, after p' converges. Also, we relax (1f) by allowing $p'_{c,v}$ and $z'_{s,c,v}$ to be any real number between 0 and 1. We can now rewrite (1a)–(1f) as:

CoP-Primal

$$\max \sum_{s,c,v \in \mathbb{I}_s} U_s(X_v) z'_{s,c,v} \quad (10a)$$

$$\text{s.t.} \quad \sum_v Y_v p'_{c,v} \leq B_c, \quad \forall c \in \mathbb{C}, \quad (10b)$$

$$\sum_{c,v \in \mathbb{I}_s} z'_{s,c,v} = 1, \quad \forall s \in \mathbb{S}, \quad (10c)$$

$$z'_{s,c,v} \leq p'_{c,v}, \quad \forall s, c, v, \quad (10d)$$

$$\sum_{s,c,v} X_v H_{s,c}^l z'_{s,c,v} \leq R_l, \quad \forall l \in \mathbb{L}, \quad (10e)$$

$$0 \leq p'_{c,v} \leq 1, 0 \leq z'_{s,c,v} \leq 1, \quad \forall s, c, v. \quad (10f)$$

A. Overview of the Solution

Similar to our solution to the CaVe problem, we will consider a dual problem to the CoP-Primal problem. Let $\mu' := [\mu'_{s,c,v}]$, and $\lambda' := [\lambda'_l]$ be the vectors of Lagrange multipliers associated with each constraint in (10d) and (10e) respectively. The Lagrangian is then

$$\begin{aligned} L'(\mathbf{p}', \mathbf{z}', \boldsymbol{\lambda}', \boldsymbol{\mu}') \\ := \sum_{s,c,v \in \mathbb{I}_s} U_s(X_v) z'_{s,c,v} - \sum_l \lambda'_l \left(\sum_{s,c,v} X_v H_{s,c}^l z'_{s,c,v} - R_l \right) \\ - \sum_{s,c,v} \mu'_{s,c,v} (z'_{s,c,v} - p'_{c,v}). \end{aligned} \quad (11)$$

The dual objective, $D'(\boldsymbol{\lambda}', \boldsymbol{\mu}')$, is defined as the maximum value of $L'(\mathbf{p}', \mathbf{z}', \boldsymbol{\lambda}', \boldsymbol{\mu}')$ over $\mathbf{p}'$ and $\mathbf{z}'$ subject to constraints (10b), (10c) and (10f). We call the optimization problem CoP-Lagrangian:

CoP-Lagrangian

$$\max L'(\mathbf{p}', \mathbf{z}', \boldsymbol{\lambda}', \boldsymbol{\mu}') \quad (12a)$$

$$\text{s.t.} \quad \sum_v Y_v p'_{c,v} \leq B_c, \quad \forall c \in \mathbb{C}, \quad (12b)$$

$$\sum_{c,v \in \mathbb{I}_s} z'_{s,c,v} = 1, \quad \forall s \in \mathbb{S}, \quad (12c)$$

$$0 \leq p'_{c,v} \leq 1, 0 \leq z'_{s,c,v} \leq 1, \quad \forall s, c, v. \quad (12d)$$

Remark 2: We note that an important difference between CoP-Lagrangian and CaVe-Lagrangian is that CoP-Lagrangian relaxes the constraint (10d) as well. Since the pseudo-variable $z'_{s,c,v}$ in CoP-Primal bears no physical meaning, this constraint can now be temporarily violated in practice.

The dual problem, which we call CoP-Dual, is to find the Lagrange multipliers that minimize $D'(\boldsymbol{\lambda}', \boldsymbol{\mu}')$:

⁴The pseudo-variables carry state information that needs to be shared between user applications and the network in the implementation.

CoP-Dual

$$\min D'(\boldsymbol{\lambda}', \boldsymbol{\mu}') \quad (13a)$$

$$\text{s.t.} \quad \lambda'_l \geq 0, \quad \forall l \in \mathbb{L}, \quad (13b)$$

$$\mu'_{s,c,v} \geq 0, \quad \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}. \quad (13c)$$

It is straightforward to show the following theorem:

Theorem 4: CoP-Primal and CoP-Dual have the same optimal value.

Proof: See [6]. ■

We will solve CoP-Primal by solving CoP-Dual. We discuss our solutions to CoP-Lagrangian and CoP-Dual below.

B. The Solution to CoP-Lagrangian

We first rewrite $L'(\mathbf{p}', \mathbf{z}', \boldsymbol{\lambda}', \boldsymbol{\mu}')$ as:

$$\begin{aligned} L'(\mathbf{p}', \mathbf{z}', \boldsymbol{\lambda}', \boldsymbol{\mu}') \\ = \sum_s \sum_{c,v} z'_{s,c,v} \left(U_s(X_v) - X_v \sum_{l: H_{s,c}^l = 1} \lambda'_l - \mu'_{s,c,v} \right) \\ + \sum_c \sum_v p'_{c,v} \sum_s \mu'_{s,c,v} + \sum_l \lambda'_l R_l. \end{aligned} \quad (14)$$

Let $\mathbf{z}'_s$ be the vector containing all $[z'_{s,c,v}]$ for a given s and $\mathbf{p}'_c$ be the vector containing all $[p'_{c,v}]$ for a given c . Also, let $\bar{L}_s(\mathbf{z}'_s, \boldsymbol{\lambda}', \boldsymbol{\mu}') := \sum_{c,v} z'_{s,c,v} [U_s(X_v) - X_v \sum_{l: H_{s,c}^l = 1} \lambda'_l - \mu'_{s,c,v}]$, $\hat{L}_c(\mathbf{p}'_c, \boldsymbol{\mu}') := \sum_v p'_{c,v} (\sum_s \mu'_{s,c,v})$, and $B(\boldsymbol{\lambda}') := \sum_l \lambda'_l R_l$. Then, we have

$$\begin{aligned} L'(\mathbf{p}', \mathbf{z}', \boldsymbol{\lambda}', \boldsymbol{\mu}') \\ = \sum_s \bar{L}_s(\mathbf{z}'_s, \boldsymbol{\lambda}', \boldsymbol{\mu}') + \sum_c \hat{L}_c(\mathbf{p}'_c, \boldsymbol{\mu}') + B(\boldsymbol{\lambda}'), \end{aligned} \quad (15)$$

which gives rise to a natural decomposition among all users and network caches. Specifically, consider the two subproblems, namely, CoP-User_s and CoP-Cache_c, below. For fixed vectors $\boldsymbol{\lambda}'$ and $\boldsymbol{\mu}'$, CoP-Lagrangian can be solved by solving CoP-User_s for each s and CoP-Cache_c for each c .

CoP-User_s

$$\max \sum_{c,v} z'_{s,c,v} (U_s(X_v) - X_v \sum_{l: H_{s,c}^l = 1} \lambda'_l - \mu'_{s,c,v}) \quad (16a)$$

$$\text{s.t.} \quad \sum_{c,v \in \mathbb{I}_s} z'_{s,c,v} = 1, \quad (16b)$$

$$0 \leq z'_{s,c,v} \leq 1, \quad \forall c \in \mathbb{C}, v \in \mathbb{V}. \quad (16c)$$

CoP-Cache_c

$$\max \sum_v p'_{c,v} \sum_s \mu'_{s,c,v} \quad (17a)$$

$$\text{s.t.} \quad \sum_v Y_v p'_{c,v} \leq B_c, \quad (17b)$$

$$0 \leq p'_{c,v} \leq 1, \quad \forall v \in \mathbb{V}. \quad (17c)$$

CoP-User_s can be solved by the following algorithm: First, find (c^*, v^*) that has the maximum value of $U_s(X_v) -$

$X_v \sum_{l: H_{s,c}^l=1} \lambda'_l - \mu'_{s,c,v}$ among all (c, v) with $v \in \mathbb{I}_s$. Ties can be broken arbitrarily. Second, set $z'_{s,c^*,v^*} = 1$, and $z'_{s,c,v} = 0$ for all other (c, v) . Alg. 3 shows the algorithm.

On the other hand, CoP-Cache_c can be solved by the following greedy algorithm: First, sort all video versions v in decreasing order of $\frac{\sum_s \mu'_{s,c,v}}{Y_v}$ so that $\frac{\sum_s \mu'_{s,c,1}}{Y_1} \geq \frac{\sum_s \mu'_{s,c,2}}{Y_2} \geq \dots$. Second, starting from $v = 1$, set $p_{c,v}$ to be the largest possible value without violating any constraints. Specifically, set $p'_{c,v} = \min\{1, (B_c - \sum_{v' < v} Y_{v'} p'_{c,v'}) / Y_v\}$. It is straightforward to verify that this greedy algorithm achieves the optimal solution for CoP-Cache_c, since it is a fractional knapsack problem.

Remark 3: Recall that $p_{c,v}$ is the indicator function that c stores v , which needs to be an integer. The optimal solution to CoP-Cache_c may not be integer. However, from the description of our greedy algorithm, it is obvious that, for each c , there is at most one v with non-integer $p_{c,v}$. In practice, we make each network cache c store only video versions with $p_{c,v} = 1$. Since all but one version have integer $p_{c,v}$, this approach is close to optimal.

C. The Solution to CoP-Dual

The CoP-Dual problem involves two Lagrange multipliers, λ' and μ' . They are updated as in Alg. 4 and 5. The following lemma and theorem, whose proofs are omitted due to space constraint, show that these algorithms solve CoP-Dual.

Lemma 2: Given λ' and μ' , let z'^* and p'^* be the vectors that solve CoP-User_s and CoP-Cache_c. Then the vector $g' := [R_l - \sum_{s,c,v} X_v H_{s,c}^l z'^*_{s,c,v}, [p'^*_{c,v} - z'^*_{s,c,v}]]$ is a subgradient of $D'(\lambda', \mu')$.

Proof: See [6]. ■

Theorem 5: Let $\{h_t\}$ be a sequence of non-negative numbers with $\sum_{t=0}^{\infty} h_t = \infty$ and $\lim_{t \rightarrow \infty} h_t = 0$, then Alg. 4 and 5 together solve CoP-Dual.

Proof: See [6]. ■

Algorithm 3 CoP-User_s Algorithm

- 1: Obtain μ' and λ'
 - 2: $z'_{s,c,v} \leftarrow 0, \forall c, v$
 - 3: $(c^*, v^*) \leftarrow \operatorname{argmax}_{c,v \in \mathbb{I}_s} U_s(X_v) - X_v \sum_{l: H_{s,c}^l=1} \lambda'_l - \mu'_{s,c,v}$
 - 4: $z'_{s,c^*,v^*} \leftarrow 1$
-

Algorithm 4 CoP-Link_l Algorithm

- 1: $t \leftarrow 0, \lambda'_l \leftarrow 0$
 - 2: **while** true **do**
 - 3: Obtain z' from Alg. 3
 - 4: $\lambda'_l \leftarrow \left[\lambda'_l + h_t \left(\sum_{s,c,v} X_v H_{s,c}^l z'_{s,c,v} - R_l \right) \right]^+$
 - 5: $t \leftarrow t + 1$
-

V. IMPLEMENTATION ON NAMED DATA NETWORKING

In this section, we discuss the implementation of our algorithms on Named Data Networking (NDN). We first

Algorithm 5 CoP-Cache_c Algorithm

- 1: $t \leftarrow 0, \mu'_{s,c,v} \leftarrow 0$
 - 2: **while** true **do**
 - 3: Obtain z' from Alg. 3
 - 4: $\mu'_{s,c,v} \leftarrow [\mu'_{s,c,v} + h_t(z'_{s,c,v} - p'_{c,v})]^+ \forall s, v$
 - 5: Sort all versions so that $\frac{\sum_s \mu'_{s,c,1}}{Y_1} \geq \frac{\sum_s \mu'_{s,c,2}}{Y_2} \geq \dots$
 - 6: $B' \leftarrow B_c$
 - 7: **for** $v = 1 \rightarrow |\mathbb{V}|$ **do**
 - 8: $p'_{c,v} \leftarrow \min\{1, \frac{B'}{Y_v}\}$
 - 9: $B' \leftarrow B' - Y_v p'_{c,v}$
 - 10: $t \leftarrow t + 1$
-

introduce the NDN architecture briefly, and then show how we implement our algorithms following the NDN philosophy.

A. NDN Architecture

NDN is a future Internet architecture where every piece of data is associated with a unique hierarchical name. When a user wants to obtain a piece of named data, the user device sends out an *interest packet* with the name of the data. Note that usually the interest packet does not specify the destination location. NDN routers have built-in caches. When a router receives an interest packet, it first checks whether the named data is cached or not. If cached, it directly replies with the corresponding data packet. Otherwise, it forwards the interest packet to the next hop according to the employed forwarding strategy. The content producer e.g. video service provider is responsible for generating data packets for a certain name space. The data packet follows the reverse route of the interest packet to the user.

B. Placement of Data

In our implementation, there are three types of data: packets of video contents, decision variables ($z'_{s,c,v}$ and $p_{c,v}$), and Lagrange multipliers (λ_l , λ'_l , and $\mu'_{s,c,v}$). We assign each of them a unique name. For example, a video version has a name prefix such as */r/file1/v1*, and $\mu'_{1,2,3}$ has */mu2/1_3*. Each prefix is appended a sequence number to uniquely identify video packets and variables in different iterations. Naturally, video contents are placed at network caches according to the video versions.⁵ Decision variables $z'_{s,c,v}$ are stored and updated at the corresponding user s . Decision variables $p_{c,v}$ and Lagrange multipliers $\mu'_{s,c,v}$ are stored and updated at the corresponding network cache c . Finally, Lagrange multipliers λ_l and λ'_l of link l from node A to B are stored and updated at node A that is closer to the cache.

C. Implementation of User Algorithms

From Alg. 1 and 3, we can see that each user s needs to know the values of $p_{c,v}$, λ_l , λ'_l , and $\mu'_{s,c,v}$. Each user periodically sends out interest packets for the named data of these variables. Since the names of these data indicate the entities that store them, routers can easily route the interest

⁵Videos are cached in full rather than at the packet level.

packets to the correct destinations. Further, as data packets traverse in the reverse route of their corresponding interest packets, each router can cache all latest values of $p_{c,v}$ and λ_l that pass through it.

With the information of $p_{c,v}$ and λ_l , each user s can find the best video version v^* and cache c^* via Alg. 1. User s then sends out interest packets for video version v^* at a rate indicated by X_{v^*} . Note that these interest packets only contain information about the video version v^* , and not the destination c^* . Nevertheless, the following forwarding strategy ensures the interest packet will be eventually forwarded to c^* assuming no link failure or topology change: When a router receives an interest packet for video version v^* , it finds the network cache $c^\dagger$ that has the smallest *cost*, where the cost is defined as $\sum_l \lambda_l$ over all link l on the path to the network cache c , among those that store v^* , i.e., $p_{c,v^*} = 1$. It then forwards the interest packet to the next router on the path toward $c^\dagger$. Note that routers store all values of $p_{c,v}$ and λ_l that pass through it and thus do not need additional message passing.

With the information of $p_{c,v}$, λ_l and μ'_l , each user s can decide the video version v^* and network cache c^* such that $z'_{s,c^*,v^*} = 1$ via Alg. 3. Each user s then sends out a pseudo-interest packet with the name of z'_{s,c^*,v^*} . We call it a pseudo-interest packet since it is used to inform the caches the changes of $z_{s,c,v}$. The replied data packet from cache c^* carries no meaning payload and is ignored.

D. Implementations for Routers and Caches

We now discuss the implementations of Alg. 2, 4, and 5. In Alg. 2, each router needs to know $\sum_{s,c,v} X_v H_{c,v}^l z_{s,c,v}$ to update λ_l for its links. We note that $\sum_{s,c,v} X_v H_{c,v}^l z_{s,c,v}$ can be estimated by the product of the rate of interest packets going through the opposite link to l and video data packet size. As the router knows the rate of interest packets going through l , it can update λ_l directly without requesting additional information. Likewise, Alg. 4, and 5 can be carried out if one knows $z'_{s,c,v}$. This is achieved by user s sending out a pseudo-interest packet as explained in Section V-C. Besides, R_l , the maximum supportable data rate of link l , is obtained from stress tests.

VI. EVALUATIONS

We present our simulation evaluation results in this section. All simulations are conducted on ndnSIM [3], an ns-3 based NDN simulator.

We consider the wireless edge network in Fig. 1 for evaluation. Same as in [4], the topology of network caches follows the three-tier hierarchy of the YouTube video delivery system. There are 15 network routers with caches in total, including the root node and 8 edge caches. Each edge cache serves 20 users who have different types of devices and are interested in different videos.

We consider a catalog of 200 different videos, each with 5 different versions. The popularity of these videos follows the Zipf distribution with the shape parameter equal to 1. The 5 versions correspond to video resolutions of 360p, 720p, 1080p,

1440p (2K), and 2160p (4K) respectively.⁶ The data rate of streaming each video version is set based on measurement results for YouTube videos with H.264 codec [7]. The access link capacities between users and edge caches are 25 Mbps each so that one can stream a 4K video. The capacities of links between caches and the root node are 100 Mbps each so that the number of concurrent 4K streams is low. We assume each video is one-hour long, and the file sizes of video versions are calculated accordingly. The root node holds all video versions. Each edge (or primary), secondary, and tertiary cache is assumed to be able to hold all versions of one, two, and four videos respectively.

As for user utilities, we categorizes user devices into three types: smartphones, laptops or tablets, and TVs. The utility function of each user has the form $U(X_v) = \alpha \ln \min(X_v, \bar{X})$, where α is a scaling factor capturing the effect of the screen size, X_v is the data rate of video version v in Mbps, and $\bar{X}$ is a cutoff rate reflecting the limit of the device resolution. For the three types, we set a scaling factor of 20, 40, 60 and a cutoff rate corresponding to a 1080p, 2K, 4K video respectively. Besides, we set $U(0) = -100$, which is much smaller than all regular utilities.

To evaluate the performance of our algorithms, we implement and compare the following four policies:

- **Optimal:** This policy tries to find the optimal solution to the CaVe-CoP problem by solving the integer program numerically via the GLPK toolbox. Note that it is a centralized policy and involves solving a high-dimensional problem.
- **CaVe-CoP:** This refers to our algorithms Alg. 1–5.
- **CaVe-CAV:** In this policy, each user employs our algorithms for CaVe. For content placement, if a network cache decides to store a video version, it needs to cache all versions (CAV) of the same video. We note that this content placement strategy is consistent with design practices in commercial CDNs. As a result, each network cache simply stores the most popular videos, subject to its storage constraint.
- **Greedy-CoP:** In this policy, each user chooses the version that matches its cutoff rate. Network caches employ our algorithms for CoP.

For each simulation, we use the video contents that each user actually receives to calculate the total utility of all users and the average % stall time, i.e. the percentage of time that video streaming stalls⁷, of all users. The metrics are calculated at each CaVe iteration, i.e. every 0.1 s. We run CoP iterations every 0.2 s, and apply content placement results at 20 s.

Fig. 2 and Fig. 3 present our simulation results. For our simulated scenario, Optimal cannot find the exact integer solution for utility. Instead, it reports a upper bound from linear programming (LP) relaxation, and a lower bound by integer programming (IP) heuristics. Note that Optimal reports ideal

⁶The aspect ratio is assumed to be 16 : 9 as in YouTube. For example, a 720p video has a resolution of 1280 × 720.

⁷Video streaming stalls when all received video contents are consumed.

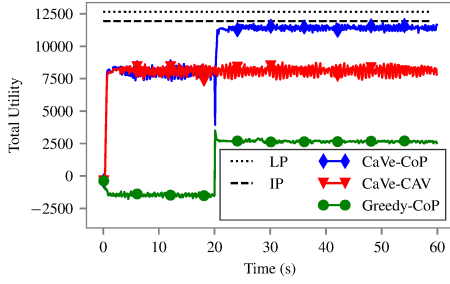


Fig. 2. Comparison of total utility.

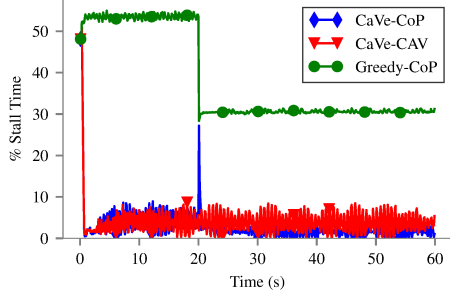


Fig. 3. Comparison of % stall time.

utility instead of perceived utility. We can observe that our Cave-CoP policy achieves near-optimal utility, significantly outperforming the two baseline policies, even though they involve subsets of our algorithms. Besides, our policy approaches zero stall time. Note that the jumps near 20s in the figures are due to applying content placement results.

VII. RELATED WORK

There has been rich literature on adaptive video streaming. An early work identified a cross layer framework for adaptive video streaming in IP networks [8]. More recently, experiment-based investigations have been conducted on YouTube [9] and Netflix [10]. Liu *et al.* [11] made a case for a coordinated control plane across CDNs for video streaming to provide high quality of experience. Wireless edge networks are promising to enhance the benefits of CDNs for adaptive video streaming [12].

Content caching is crucial for practical adaptive video streaming in wireless edge networks. Considering distributed caches, Ramadan *et al.* [4] proposed the abstraction of “BIG” cache to effectively utilize the resources. Applegate *et al.* [13] studied optimal content placement of videos with a focus on scalability. There have been many studies on joint optimization of content caching and packet routing. Yeh *et al.* [14] proposed a framework for joint forwarding and caching in NDN. Wang *et al.* [15] employed stochastic network utility maximization and developed a distributed forwarding and caching algorithm. Ioannidis and Yeh [16] studied the routing cost minimization problem of joint routing and caching, where the cost is incurred per link. These studies are not directly

applicable to multi-version video streaming since different versions of the same video can be stored in different caches.

VIII. CONCLUSION

In this paper, we have studied the CaVe-CoP problem, i.e. the joint optimization of cache-version selection and content placement, for adaptive video streaming in wireless edge networks. Realizing that there is a practical timescale separation between CaVe and CoP, we have proposed a set of algorithms that provably optimize CaVe and CoP respectively. Further, we show that our algorithms can be practically implemented on NDN in a distributed fashion. Simulation evaluations on ndnSIM demonstrate that our policy significantly outperforms baseline policies with conventional heuristics.

REFERENCES

- [1] P. Mach and Z. Becvar, “Mobile edge computing: A survey on architecture and computation offloading,” *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 2017.
- [2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, “Named data networking,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, Jul. 2014.
- [3] S. Mastroakis, A. Afanasyev, and L. Zhang, “On the evolution of ndnSIM: an open-source simulator for NDN experimentation,” *ACM Computer Communication Review*, Jul. 2017.
- [4] E. Ramadan, A. Narayanan, Z.-L. Zhang, R. Li, and G. Zhang, “BIG cache abstraction for cache networks,” in *2017 IEEE 37th Int. Conf. Distributed Computing Systems (ICDCS)*. IEEE, Jun. 2017.
- [5] A. Araldo, F. Martignon, and D. Rossi, “Representation selection problem: Optimizing video delivery through caching,” in *2016 IFIP Networking Conf. and Workshops*. IEEE, May 2016.
- [6] A. Sasikumar, T. Zhao, I.-H. Hou, and S. Shakkottai. (2019, Mar.) Cache-version selection and content placement for adaptive video streaming in wireless edge networks. Full paper with proofs. [Online]. Available: <https://arxiv.org/abs/1903.12164>
- [7] M. McFly. (2018, Jan.) Test video quality 720p 1080p 1440p 2160p 4320p max bitrate which compresses YouTube. [Online]. Available: <https://www.tutorialguidacomefare.com/test-video-quality-720p-1080p-1440p-2160p-max-bitrate-which-compresses-youtube/>
- [8] T. Ahmed, A. Mehaoua, R. Boutaba, and Y. Iraqi, “Adaptive packet video streaming over IP networks: a cross-layer approach,” *IEEE J. Sel. Areas Commun.*, vol. 23, no. 2, pp. 385–401, Feb. 2005.
- [9] V. K. Adhikari, S. Jain, Y. Chen, and Z.-L. Zhang, “Vivisectioning YouTube: An active measurement study,” in *2012 Proc. IEEE INFOCOM*, Mar. 2012.
- [10] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang, “Unreeling Netflix: Understanding and improving multi-CDN movie delivery,” in *Proc. IEEE INFOCOM*. IEEE, Mar. 2012.
- [11] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, “A case for a coordinated internet video control plane,” *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, p. 359, sep 2012.
- [12] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, “Collaborative multi-bitrate video caching and processing in mobile-edge computing networks,” in *2017 13th Annu. Conf. Wireless On-demand Network Systems and Services (WONS)*. IEEE, Feb. 2017.
- [13] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, “Optimal content placement for a large-scale VoD system,” *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2114–2127, aug 2016.
- [14] E. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, and D. Leong, “VIP: A framework for joint dynamic forwarding and caching in named data networks,” in *Proc. 1st Int. Conf. Information-Centric Networking (ICN '14)*. ACM Press, 2014.
- [15] Y. Wang, W. Wang, Y. Cui, K. G. Shin, and Z. Zhang, “Distributed packet forwarding and caching based on stochastic network utility maximization,” *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1264–1277, Jun. 2018.
- [16] S. Ioannidis and E. Yeh, “Jointly optimal routing and caching for arbitrary network topologies,” in *Proc. 4th ACM Conf. Information-Centric Networking (ICN '17)*. ACM Press, 2017.