CACHE-VERSION SELECTION AND CONTENT PLACEMENT FOR MULTI-RESOLUTION VIDEO STREAMING IN INFORMATION-CENTRIC NETWORKS

A Thesis

by

ARCHANA SASIKUMAR

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	I-Hong Hou
Co-Chair of Committee,	Radu Stoleru
Committee Member,	Pierce Cantrell
Head of Department,	M. Begovic

December 2018

Major Subject: Computer Engineering

ABSTRACT

Information Centric Networks (ICN) is an infrastructure that focuses on information retrieval rather than end to end connections. ICN uses 2 features - name based routing and in-network caching in order to attain better performance. Named Data Networks (NDN) is an architecture for Information Centric Networks (ICN). In this thesis, we implement a version selection cum content placement policy (CaVe-CoP) that takes advantage of both features. We focus on multi-resolution video streaming and implement a scheme where only an optimal set of resolutions of videos need to be cached in order to obtain higher network utility. This distinction between multiple resolutions of the same video is possible today because of the varied devices available for video streaming that have different resolution constraints. We first formulate and solve an optimization problem for version selection and content placement in a generic network that supports multi-resolution video streaming and has in-network caches. Next, we implement the solution in an NDN-compliant framework (ndnSIM) as a distributed algorithm. We compare our policy against 2 other policies - 1) where all resolutions of a content are cached, and 2) where the user opts for a greedy version selection. Our simulations on general network topologies show a fast convergence rate, higher utility and a lower stall time in comparison to both these policies.

ACKNOWLEDGMENTS

Working on my Masters Thesis with Dr. I-Hong Hou was a great learning opportunity for me. I thoroughly appreciate and admire Dr. Hou's endless push to explore more and aim for perfection. It was pivotal to this learning process. Thank you Dr. Hou. I would also like to express my gratitude to Dr. Shakkottai for his consistent support and valuable inputs for improvements.

The support that I received from their Ph.D students was also very significant in enabling me to complete this work. I am grateful to their Ph.D students Tao Zao and Desik Rangarajan for helping me throughout this project.

Last but not the least, I would like to thank my family and friends for their constant support and motivation throughout my academic journey.

CONTRIBUTORS AND FUNDING SOURCES

This work was supported by the research groups of Professor I-Hong Hou and Professor Srinivas Shakkottai of the Department of Electrical and Computer Engineering.

All other work conducted for the thesis was completed by the student independently.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

1. INTRODUCTION

Video streaming has become the dominant application for modern network traffic. In order to alleviate the burden of backbone network and to reduce latency, content delivery networks (CDNs) have been deployed to store popular videos at content servers close to the users. Simultaneously, as users are accessing videos from a variety of devices, ranging from smart phones to 4K televisions, multi-resolution video streaming, which encodes the same video into versions with different resolutions, has been used to deliver the most appropriate version to each user based on its service requirement and network congestion. For example, YouTube currently encodes each video into five different versions.

In this thesis, we study the interplay between three important components in multi-resolution video streaming: *cache selection*, where each user determines which content 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 content 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 resolutions of users, the communication capacities of links, and the storage capacities of content 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. We then solve the CoP problem by considering its influence to solutions for the CaVe problem. For both problems, we propose simple algorithms and prove that they converge to the optimal solutions.

We further study the implementation of our algorithms on Named Data Networking (NDN), one of the most popular architectures of information-centric networking (ICN). We demonstrate that our algorithms can be implemented in a fully distributed and fully NDN-compliant fashion.

In particular, while NDN does not allow users to explicitly specify its selected content server, our implementation contains a distributed routing protocol that ensures users always obtain their selected video versions from the optimal content server. Moreover, we show that the overhead of our algorithms is minimal since the update of many parameters can be directly inferred by the local information at each node under the NDN architecture.

We also implement and evaluate our algorithms in ndnSIM, the standard network simulator for NDN. To demonstrate the utility of our algorithms, we evaluate two other policies. One policy uses our optimal solution for CaVe and a standard policy for CoP. The other policy users a standard policy for CaVe and our optimal solution for CoP. Simulation results show that our algorithms significantly outperform these two policies.

The rest of the thesis is organized as follows. Section 3 introduces our system model and the formulation of CaVe-CoP. Solutions to the two problems CaVe and CoP are introduced in Section 4 and 5, respectively. In Section 6, we discuss the implementation of our algorithms in NDN. Section 7 demonstrates the simulation results. Section 2 reviews some related literature. Finally, Section 8 concludes the paper.

2. RELATED WORK

Information-centric network (ICN) has been a hot research topic in recent years since it is expected to better match the usage for Internet nowadays and in the foreseeable future, i.e. delivering information or content users want. Several architectures for ICN have been proposed, including named-data networking (NDN) [1, 2] and MobilityFirst [3]. A comprehensive survey on ICN has been conducted by Xylomenos *et al.* [4].

Content caching is a crucial part of ICN and is fundamentally coupled with packet forwarding and routing, which naturally calls for a joint optimization approach. Yeh *et al.* [5] proposed a framework for joint forwarding and caching in named data networks (NDN), where algorithms are designed on a virtual control plane and then mapped to the actual interest and data packet plane in NDN. The framework has been extended to deal with interest suppression in NDN in Lai *et al.* [6]. They focused on throughput-optimal analysis rather than utility maximization. Wang *et al.* [7] employed stochastic network utility maximization and developed a distributed forwarding and caching algorithm. Ioannidis and Yeh [8] studied the routing cost minimization problem of joint routing and caching, where the cost is incurred per link. In contrast, we consider utility per user per content version, which leads to a distinct problem formulation. Besides, the above studies did not consider the content version selection problem.

Regarding content version selection, there has been rich literature on adaptive video streaming in various networks. An early work identified a cross layer framework for adaptive video streaming in IP networks [9]. Jurca *et al.* [10] presented media delivery architectures over P2P networks for adaptive video streaming. More recently, experiment-based investigations have been conducted on content delivery networks (CDN) run by Akamai [11] and Netflix [12] respectively. Lederer *et al.* [13] presented implementation study on adaptive multimedia streaming in ICN. However, to the best of our knowledge, no existing studies have explored the interaction between content placement, cache selection, and content version selection in ICN.

Our work formulates the CaVe-CoP (joint cache version selection and cache placement) prob-

lem as a network utility maximization (NUM) problem, and uses the well-known primal dual approach and dual decomposition [14, 15]. However, there are notable differences between our work and traditional NUM research. Existing studies have explored various scenarios including time varying channel with delay constraints [16], spatiotemporally coupled constraints [17], multiple flow classes [18], multiple gateways [19], multiple protocols [20] and so on, while assuming a static source-destination pair per user (flow). In contrast, in our work a user could obtain its desired content from in-network caches as well as the content producer. The content version selection problem is intrinsically intertwined with the content placement and cache selection problem. The joint optimization problem is naturally an integer programming problem, instead of a convex programming problem.

3. SYSTEM MODEL

We consider an information-centric network (ICN) where a group of network caches and routers jointly host a set of videos and serve a set of video streaming users. 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. Fig. 3.1 illustrates an example of such a network. There is a route between each user s and each network cache c, and we define $H^l_{s,c}$ as the indicator function that link l is on the route between s and c.

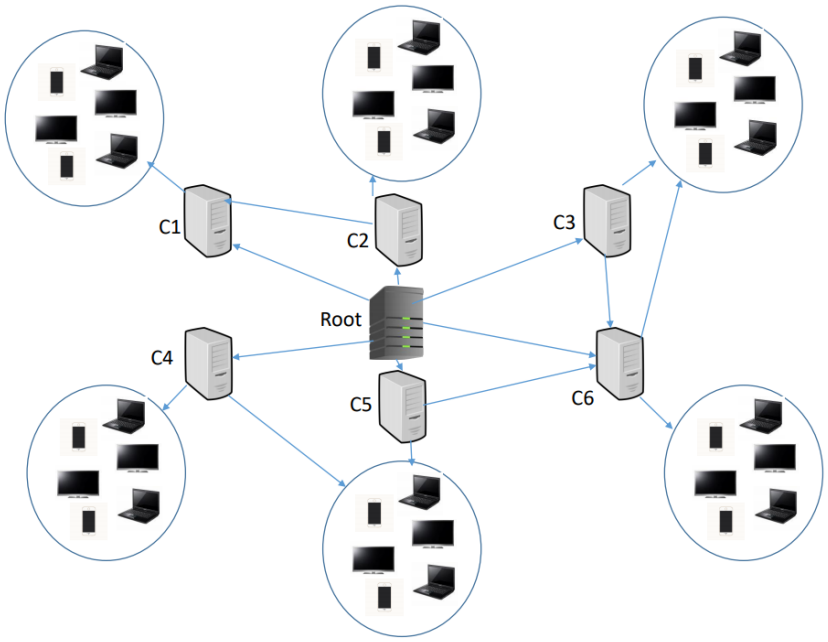

Figure 3.1: General ICN Layout

We consider multi-resolution video streaming where each video is encoded into multiple different versions, with different versions corresponding to 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 bit rate of v and Y_v to denote the file size of v, which can be computed as the product of X_v and the duration of the video. Obviously, low-resolution versions have small

X_v and Y_v, and high-resolution versions have large X_v and Y_v. For the ease of theoretical analysis, we also 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 also assume that there exists at least a network cache c with infinite storage, $B_c = \infty$, and stores all video versions. Such an assumption is to ensure that at least one copy of each video version exists in the ICN. We refer to the problem of determining $p_{c,v}$ as the *content placement (CoP) problem*.

On the other hand, 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 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 $H^l_{s,c} = 1$ 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 $\sum_{s,c,v} X_v H^l_{s,c} z_{s,c,v}$. We consider that each link l has a finite capacity of R_l, and hence we require that $\sum_{s,c,v} X_v H^l_{s,c} z_{s,c,v} \leq R_l$, for all $l \in \mathbb{L}$.

Finally, each user obtains some utility based on its perceived video quality. Specifically, we consider that each user s has a utility function $U_s(\cdot)$ and it obtains a utility of $U_s(X_v)$ when it is 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 are much less sensitive to low resolutions than those watching videos on televisions.

We aim to maximize the total utility in the ICN by choosing the optimal $\vec{p} := [p_{c,v}]$ and $\vec{z} := [z_{s,c,v}]$, subject to all aforementioned constraints. Specifically, we aim to solve the following optimization problem:

CaVe-CoP:

$$\max \sum_{s,c,v \in \mathbb{I}_s} U_s(X_v) z_{s,c,v} \tag{3.1}$$

$$s.t. \sum_{v} Y_v p_{c,v} \leq B_c, \forall c \in \mathbb{C}, \tag{3.2}$$

$$\sum_{c,v \in \mathbb{I}_s} z_{s,c,v} = 1, \forall s \in \mathbb{S}, \tag{3.3}$$

$$z_{s,c,v} \leq p_{c,v}, \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}, \tag{3.4}$$

$$\sum_{s,c,v} X_v H^l_{s,c} z_{s,c,v} \leq R_l, \forall l \in \mathbb{L}, \tag{3.5}$$

$$p_{c,v} \in \{0,1\}, z_{s,c,v} \in \{0,1\}, \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}. \tag{3.6}$$

While the the utility maximization problem studied in this thesis 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 the same video version in ICN 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. To the best of our knowledge, there are no existing studies that explore the interaction between content placement, cache selection, and version selection.

The decision variable in (3.1) – (3.6) are $\vec{p}$ and $\vec{z}$. We note that there is a practical timescale separation between the update for $\vec{p}$ and that for $\vec{z}$. When a user changes its values for $\vec{z}$, it simply establishes a connection to a different network cache and adjusts its own playback resolution. Hence, $\vec{z}$ can be updated as frequent as, 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, $\vec{p}$ can only be updated infrequently.

Our proposed solution for CaVe-CoP is based on the observation of the timescale separation between the update for $\vec{p}$ and that for $\vec{z}$. In Section 4, we will first consider the CaVe problem by finding the optimal $\vec{z}$ for given $\vec{p}$. Next, in Section (5), we will consider the CoP problem. In order to find the optimal $\vec{p}$, we will introduce pseudo-variables $\vec{z'} := [z'_{s,c,v}]$ that are updated at the same frequency as $\vec{p}$ to address the issue with timescale separation.

Finally, we note that (3.1) – (3.6) is an integer programming problem since $p_{c,v}$ and $z_{s,c,v}$ are integers. To obtain tractable results, we will relax (3.6) 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 4, 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 5.

4. 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 (3.1) – (3.6), we focus on finding the optimal $\vec{z} := [z_{s,c,v}]$ to maximize total utility in the ICN when $\vec{p} := [p_{c,v}]$ is given and fixed.

4.1 Overview of the Solution

We begin by rewriting the optimization problem (3.1) – (3.6) for the CaVe problem. Since $\vec{p}$ is given and fixed, constraint (3.2) no longer applies. Further, we relax the constraint (3.6) 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:

$$\max \sum_{s,c,v \in \mathbb{I}_s} U_s(X_v) z_{s,c,v} \tag{4.1}$$

$$s.t. \sum_{c,v \in \mathbb{I}_s} z_{s,c,v} = 1, \forall s \in \mathbb{S}, \tag{4.2}$$

$$z_{s,c,v} \leq p_{c,v}, \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}, \tag{4.3}$$

$$\sum_{s,c,v} X_v H^l_{s,c} z_{s,c,v} \leq R_l, \forall l \in \mathbb{L}, \tag{4.4}$$

$$0 \leq z_{s,c,v} \leq 1, \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}. \tag{4.5}$$

We will consider a dual problem to CaVe-Primal. We associate a Lagrange multiplier, λ_l, for each link capacity constraint (4.4), for all $l \in \mathbb{L}$. Let $\vec{\lambda} := [\lambda_l]$ be the vector of Lagrange

multipliers. The Lagrangian is obtained as follows:

$$
\begin{aligned}
& L(\vec{z}, \vec{\lambda}) \\
&= \sum_{s,c,v \in \mathbb{I}_s} U_s(X_v) z_{s,c,v} - \sum_{l} \lambda_l (\sum_{s,c,v} z_{s,c,v} H^l_{s,c} X_v - R_l)
\end{aligned} \tag{4.6}
$$

The dual objective function, which we call CaVe-Lagrangian, is to maximize the Lagrangian with the constraints (4.2),(4.3) and (4.5). The CaVe-Lagrangian can thus be written as follows:

CaVe-Lagrangian:

$$
\begin{aligned}
\max \quad & L(\vec{z}, \vec{\lambda}) && (4.7) \\
s.t. & \sum_{c,v} z_{s,c,v} = 1 \quad \forall s \in \mathbb{S}, && (4.8) \\
& z_{s,c,v} \leq p_{c,v}, \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}, && (4.9) \\
& 0 \leq z_{s,c,v} \leq 1, \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}. && (4.10)
\end{aligned}
$$

Remark 1. *We note that, in defining the CaVe-Lagrangian problem, we only relax the constraint (4.4) by associating Lagrange multipliers for it, and we keep other constraints (4.2), (4.3) and (4.5) intact. This is because constraint (4.4) specifies the constraint of link capacity. It can be temporarily violated as packets that cannot be served immediately will simply wait in the queue for service. On the other hand, constraint (4.2) states that each user needs to obtain exactly one video version, and constraint (4.3) states that each user can only obtain a video version from a network cache that stores it. These two constraints need to be satisfied at all time in practical systems, and hence we do not relax them.* □

Let $D(\vec{\lambda})$ be the maximum value of $L(\vec{z}, \vec{\lambda})$ under the constraints (4.8) – (4.10). The dual problem is to minimize $D(\vec{\lambda})$ while ensuring that all Lagrange multipliers λ_l are non-negative. We call this the CaVe-Dual and mathematically write it as:

CaVe-Dual:

$$\min \quad D(\vec{\lambda}) \tag{4.11}$$

$$s.t. \lambda_l \geq 0, \forall \lambda_l \in \mathbb{L}. \tag{4.12}$$

We first show that strong duality holds for CaVe-Primal and CaVe-Dual.

Theorem 1. *CaVe-Primal and CaVe-Dual have the same optimal value.*

Proof. The objective function of CaVe-Primal is a linear function, and hence is convex. It is straightforward to verify that the set of $\vec{z}$ that satisfies the three constraints that are not relaxed in the formulation of CaVe-Lagrangian, namely, (4.2), (4.3), and (4.5), is convex. Furthermore, the relaxed constraint (4.4) is a linear one, and strict inequality holds for the constraint if all users decide to watch the null version v_0 with $X_{v_0} = 0$, which is equivalent to setting $z_{s,c,v} = 0$ if $X_v > 0$. Hence, this theorem holds following Theorem 6.2, Chapter 6 in [21]. □

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

4.2 The Solution to CaVe-Lagrangian

We rewrite (4.6) as:

$$\begin{aligned} & L(\vec{z}, \vec{\lambda}) \\ =& \sum_{s,c,v \in \mathbb{I}_s} U_s(X_v) z_{s,c,v} - \sum_l \lambda_l (\sum_{s,c,v} z_{s,c,v} H^l_{s,c} X_v - R_l) \\ =& \sum_s \left(\sum_{c,v \in \mathbb{I}_s} z_{s,c,v} [U_s(X_v) - X_v \sum_{l: H^l_{s,c}=1} \lambda_l] \right) + \sum_l \lambda_l R_l \end{aligned} \tag{4.13}$$

We note that the above expression provides a natural user-by-user decomposition. Specifically,

by defining $\vec{z_s}$ as the vector containing all $[z_{s,c,v}]$, for a given s, and

$$L_s(\vec{z_s}, \vec{\lambda}) := \sum_{c,v \in \mathbb{I}_s} z_{s,c,v}[U_s(X_v) - X_v \sum_{l:H^l_{s,c}=1} \lambda_l], \tag{4.14}$$

then we have

$$L(\vec{z}, \vec{\lambda}) = \sum_s L_s(\vec{z_s}, \vec{\lambda}) + \sum_l \lambda_l R_l. \tag{4.15}$$

As $\vec{\lambda}$ is given in CaVe-Lagrangian, the last term $\sum_l \lambda_l R_l$ is a constant. Hence, $L(\vec{z}, \vec{\lambda})$ is maximized if one can maximize $L_s(\vec{z_s}, \vec{\lambda})$ 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 (4.9) 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$_s$:

$$\max \quad \sum_{c,v:v \in \mathbb{I}_s, p_{c,v}=1} z_{s,c,v}[U_s(X_v) - X_v \sum_{l:H^l_{s,c}=1} \lambda_l] \tag{4.16}$$

$$s.t. \sum_{c,v:v \in \mathbb{I}_s, p_{c,v}=1} z_{s,c,v} = 1, \tag{4.17}$$

$$0 \leq z_{s,c,v} \leq 1, \forall c \in \mathbb{C}, v \in \mathbb{V}. \tag{4.18}$$

It is clear that the optimal vector $\vec{z}$ that solves CaVe-User$_s$, for all s, is also the optimal vector that solves CaVe-Lagrangian.

Also, note that the only control variable in CaVe-User$_s$ is the vector $\vec{z_s}$, and all other variables, including $U_s(X_v), X_v$, and λ_l are fixed. Hence, the following algorithm solves CaVe-User$_s$: First, find (c^*, v^*) that has the maximum value of $U_s(X_v) - X_v \sum_{l:H^l_{s,c}=1} \lambda_l$ 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 describes 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, Alg. 1 shows that the optimal solution to CaVe-Lagrangian is always an integer solution.

Algorithm 1 CaVe-User$_s$ Algorithm

Obtain $\vec{p}$ and $\vec{\lambda}$
$z_{s,c,v} \leftarrow 0, \forall c, v$
$(c^*, v^*) \leftarrow \arg\max_{c,v \in \mathbb{I}_s : p_{c,v} \geq 0} (U_s(X_v) - X_v \sum_{l:H^l_{s,c}=1} \lambda_l)$
$z_{s,c^*,v^*} \leftarrow 1.$

Algorithm 2 CaVe-Link$_l$ Algorithm

$t \leftarrow 0$
$\lambda_l \leftarrow 0$
while true **do**
  Obtain $\vec{z}$ from Alg. 1
  $\lambda_l \leftarrow \left(\lambda_l + h_t[\sum_{s,c,v} X_v H^l_{s,c} z_{s,c,v} - R_l]\right)^+$
  $t \leftarrow t + 1$
end while

4.3 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. The proofs of both of them are virtually identical to Lemma 2 and Theorem 2 in [16], and are hence omitted.

Lemma 1. *Given $\vec{\lambda}$, let $\vec{z^*}$ be the vector that solves CaVe-User$_s$, then $[\sum_{s,c,v} X_v H^l_{s,c} z_{s,c,v} - R_l]$ is a subgradient of $D(\vec{\lambda})$.*

Theorem 2. *Let $\{h_t\}$ be a sequence of non-negative numbers with $\sum_{t=1}^{\infty} h_t = \infty$ and $\lim_{t\to\infty} h_t = 0$, then Alg. 2 solves CaVe-Dual.*

5. 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 3, a major challenge to our optimization problem (3.1) – (3.6) is that the vector $\vec{p}$ needs to be updated much less frequently than the vector $\vec{z}$. To address this challenge, we introduce a pseudo-variable $\vec{z'} := [z'_{s,c,v}]$, which can be updated as frequent as $\vec{p}$, to replace $\vec{z}$. Also, we relax (3.6) by allowing $p_{c,v}$ and $z'_{s,c,v}$ to be any real number between 0 and 1. We can now rewrite (3.1) – (3.6) as:

CoP-Primal

$$\max \sum_{s,c,v \in \mathbb{I}_s} U_s(X_v) z'_{s,c,v} \tag{5.1}$$

$$s.t. \sum_{v} Y_v p_{c,v} \leq B_c, \forall c \in \mathbb{C}, \tag{5.2}$$

$$\sum_{c,v \in \mathbb{I}_s} z'_{s,c,v} = 1, \forall s \in \mathbb{S}, \tag{5.3}$$

$$z'_{s,c,v} \leq p_{c,v}, \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}, \tag{5.4}$$

$$\sum_{s,c,v} X_v H^l_{s,c} z'_{s,c,v} \leq R_l, \forall l \in \mathbb{L}, \tag{5.5}$$

$$0 \leq p_{c,v} \leq 1, 0 \leq z'_{s,c,v} \leq 1, \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}. \tag{5.6}$$

5.1 Overview of the Solution

Similar to our solution to the CaVe problem, we will also consider a dual problem to the CoP-Primal problem. Let $\vec{\mu'} := [\mu'_{s,c,v}]$, and $\vec{\lambda'} := [\lambda'_l]$ be the vectors of Lagrange multipliers associated

with each constraint in (5.4), and (5.5), respectively. The Lagrangian is then

$$\begin{aligned} & L'(\vec{p}, \vec{z'}, \vec{\lambda'}, \vec{\mu'}) \\ := & \sum_{s,c,v \in \mathbb{I}_s} U_s(X_v) z'_{s,c,v} - \sum_l \lambda'_l (\sum_{s,c,v} X_v H^l_{s,c} z'_{s,c,v} - R_l) \\ & - \sum_{s,c,v} \mu'_{s,c,v} (z'_{s,c,v} - p_{c,v}). \end{aligned} \tag{5.7}$$

The dual objective function, which we call CoP-Lagrangian, is to maximize $L'(\vec{p}, \vec{z'}, \vec{\lambda'}, \vec{\mu'})$ subject to constraints (5.2), (5.3) and (5.6), for given vectors $\vec{\lambda'}$ and $\vec{\mu'}$:

CoP-Lagrangian

$$\max L'(\vec{p}, \vec{z'}, \vec{\lambda'}, \vec{\mu'}) \tag{5.8}$$

$$s.t. \sum_v Y_v p_{c,v} \leq B_c, \forall c \in \mathbb{C}, \tag{5.9}$$

$$\sum_{c,v \in \mathbb{I}_s} z'_{s,c,v} = 1, \forall s \in \mathbb{S}, \tag{5.10}$$

$$0 \leq p_{c,v} \leq 1, 0 \leq z'_{s,c,v} \leq 1, \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}. \tag{5.11}$$

Remark 2. *We note that an important difference between Cop-Lagrangian and Cave-Lagrangian is that Cop-Lagrangian relaxes the constraint (5.4) as well. Since CP-Primal uses the pseudo-variable $z'_{s,c,v}$ that has no direct physical meaning to replace $z_{s,c,v}$, this constraint can now be temporarily violated.* □

Let $D'(\vec{\lambda'}, \vec{\mu'})$ be the maximum value of $L'(\vec{p}, \vec{z'}, \vec{\lambda'}, \vec{\mu'})$ subject to constraints (5.9), (5.10) and (5.11). The dual problem, which we call CoP-Dual, is to find the Lagrange multipliers that minimize $D'(\vec{\lambda'}, \vec{\mu'})$:

CoP-Dual

$$\min D'(\vec{\lambda'}, \vec{\mu'}) \tag{5.12}$$

$$s.t.\ \lambda'_l \geq 0, \forall l \in \mathbb{L}., \tag{5.13}$$

$$\mu'_{s,c,v} \geq 0, \forall s \in \mathbb{S}, c \in \mathbb{C}, v \in \mathbb{V}. \tag{5.14}$$

Similar to Theorem 1, it is straightforward to show the following theorem:

Theorem 3. *CoP-Primal and CoP-Dual have the same optimal value.*

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

5.2 The Solution to CoP-Lagrangian

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

$$\begin{aligned} & L'(\vec{p}, \vec{z'}, \vec{\lambda'}, \vec{\mu'}) \\ =& \sum_s \left(\sum_{c,v} z'_{s,c,v}[U_s(X_v) - X_v \sum_{l:H^l_{s,c}=1} \lambda'_l - \mu'_{s,c,v}] \right) \\ &+ \sum_c [\sum_v p_{c,v} (\sum_s \mu'_{s,c,v})] + \sum_l \lambda'_l R_l. \end{aligned} \tag{5.15}$$

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

$$\begin{aligned} & L'(\vec{p}, \vec{z'}, \vec{\lambda'}, \vec{\mu'}) \\ =& \sum_s \bar{L}_s(\vec{z'_s}, \vec{\lambda'}, \vec{\mu'}) + \sum_c \hat{L}_c(\vec{p_c}, \vec{\mu'}) + B(\vec{\lambda'}), \end{aligned} \tag{5.16}$$

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 $\vec{\lambda'}$

and $\vec{\mu'}$, CoP-Lagrangian can be solved by solving CoP-User$_s$ for each s and CoP-Cache$_c$ for each c.

CoP-User$_s$

$$\max \sum_{c,v} z'_{s,c,v}[U_s(X_v) - X_v \sum_{l:H^l_{s,c}=1} \lambda'_l - \mu'_{s,c,v}] \tag{5.17}$$

$$s.t. \sum_{c,v\in\mathbb{I}_s} z'_{s,c,v} = 1, \forall s \in \mathbb{S}, \tag{5.18}$$

$$0 \leq z'_{s,c,v} \leq 1, \forall c \in \mathbb{C}, v \in \mathbb{V}. \tag{5.19}$$

CoP-Cache$_c$

$$\max \sum_{v} p_{c,v}(\sum_{s} \mu'_{s,c,v}) \tag{5.20}$$

$$s.t. \sum_{v} Y_v p_{c,v} \leq B_c, \forall c \in \mathbb{C}, \tag{5.21}$$

$$0 \leq p_{c,v} \leq 1, \forall v \in \mathbb{V}. \tag{5.22}$$

CoP-User$_s$ can be solved by the following algorithm: First, find (c^*, v^*) that has the maximum value of $U_s(X_v) - X_v \sum_{l:H^l_{s,c}=1} \lambda'_l - \mu'_{s,c,v}$ 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 $\frac{\sum_s \mu'_{s,c,1}}{Y_1} \geq \frac{\sum_s \mu'_{s,c,2}}{Y_2} \geq \ldots$. 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$_c$.

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*

Algorithm 3 CoP-User$_s$ Algorithm

1: Obtain $\vec{p}$, $\vec{\mu'}$ and $\vec{\lambda'}$
2: $z'_{s,c,v} \leftarrow 0, \forall c, v$
3: $(c^*, v^*) \leftarrow \arg\max_{c,v \in \mathbb{I}_v : p_{s,c,v} \geq 0} U_s(X_v) - X_v \sum_{l:H^l_{s,c}=1} \lambda'_l - \mu'_{s,c,v}$
4: $z'_{s,c^*,v^*} \leftarrow 1$.

Algorithm 4 Cop-Link Algorithm

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

practice, we make each network cache c store only video versions with $p_{c,v} = 1$. Since all but one versions have integer $p_{c,v}$, this approach is close to the optimal solution. □

5.3 The Solution to CoP-Dual

The CoP-Dual problem involves two Lagrange multipliers, $\vec{\lambda'}$ and $\vec{\mu'}$. 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 $\vec{\lambda'}$ and $\vec{\mu'}$, let $\vec{z'^*}$ and $\vec{p^*}$ be the vectors that solve CoP-User$_s$ and CoP-Cache$_c$, then the vector containing $[\sum_{s,c,v} X_v H^l_{s,c} z'_{s,c,v} - R_l]$ for all l and $[z'_{s,c,v} - p_{c,v}]$ for all c and v is a subgradient of $D'(\vec{\lambda'}, \vec{\mu'})$.*

Theorem 4. *Let $\{h_t\}$ be a sequence of non-negative numbers with $\sum_{t=1}^{\infty} h_t = \infty$ and $\lim_{t\to\infty} h_t = 0$, then Alg. 4 and 5 together solve CoP-Dual.*

Algorithm 5 CoP-Cache$_c$ Algorithm

$t \leftarrow 0$
$\mu'_{s,c,v} \leftarrow 0$
while true **do**
  Obtain $\vec{z'}$ from Alg. 3
  $\mu'_{s,c,v} \leftarrow \left(\mu'_{s,c,v} + h_t[z'_{s,c,v} - p_{c,v}]\right)^+ \forall s, v$
  Sort all versions so that $\frac{\sum_s \mu'_{s,c,1}}{Y_1} \geq \frac{\sum_s \mu'_{s,c,2}}{Y_2} \geq \ldots$
  $B' \leftarrow B_c$
  for $v = 1 \rightarrow |\mathbb{V}|$ **do**
    $p_{c,v} \leftarrow \min\{1, \frac{B'}{Y_v}\}$
    $B' \leftarrow B' - Y_v p_{c,v}$
  end for
  $t \leftarrow t + 1$
end while

6. IMPLEMENTATION ON NAMED DATA NETWORKING

In this section, we discuss our implementations under Named Data Networking (NDN). NDN is one of the most popular standards for information centric network (ICN), which focuses on retrieving information rather than establishing end-to-end connections. In NDN, every piece of information, such as a packet of video content is associated with a name, and is hence called a piece of named data. When a user wants to obtain a piece of named data, it sends out an *interest packet*, which contains the address of the user and the name of the data. Note that the interest packet does not specify the destination address. Routing decisions are solely based on the names in interest packets, and routers aim to forward each interest packet to the closest network cache that stores the data. When a network cache receives an interest packet for a piece of data that it stores, the network cache replies with the data. The data packet follows the reverse route of the interest packet to the user.

In this section, we demonstrate that our solution to CaVe-CoP can be directly implemented under NDN without any changes to the standard. Moreover, we show that the updates of all Lagrange multipliers can be done by simply counting the number of interest packets that the corresponding entities receive without the need of additional messages exchange.

6.1 Placement of Data

In our implementation, there are three types of data: packets of video contents, decision variables ($z_{s,c,v}$, $z'_{s,c,v}$, and $p_{c,v}$), and Lagrange multipliers (λ_l, λ'_l, and $\mu'_{s,c,v}$). Each of these pieces of data is associated with a unique name. Obviously, packets of video contents are stored at network caches that store the corresponding video versions. Decision variables $z_{s,c,v}$ and $z'_{s,c,v}$ are stored and updated at the corresponding user s. Decision variable $p_{c,v}$ and Lagrange multiplier $\mu'_{s,c,v}$ are stored and updated at the corresponding network cache c. Finally, Lagrange multipliers λ_l and λ'_l are stored and updated at the sender of link l. Hence, the names of decision variables and Lagrange multipliers indicate the entities that store them. In addition, each network cache also maintains a

pseudo-data with name $p'_{c,v}$, for each v.

6.2 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 $\mu'_{s,c,v}$. It periodically sends out interest packets for these named data. Since the names of these data indicate the entities that store them, routers can easily route the interest packets to the correct destinations. Further, as data packets traverse in the reverse route of their corresponding interest packets, each router can store all values of $p_{c,v}$ and λ_l that pass through it.

After deciding $z_{s,c,v}$, user s sends out interest packets for video version v at a rate indicated by X_v. Note that this interest packet only contains information about the video version v, and the user cannot specify the destination v under NDN. When a router receives an interest packet for video version v, it finds the network cache c^* that has the smallest distance, where the distance is defined as the sum of λ_l over all links on the path, among those that store v, i.e., $p_{c,v} = 1$, and then forwards the interest packet to the next router on the path toward c^*. Since routers store all values of $p_{c,v}$ and λ_l that pass through it, routers can easily find c^*.

After deciding $z'_{s,c,v}$, user s sends out an interest packet for the pseudo-data $p'_{c,v}$. Since the name indicates the corresponding c, routers can forward the interest packet to c. On the other hand, when a network cache c receives an interest for $p'_{c,v}$, it does not reply with a data packet, since $p'_{c,v}$ is a pseudo-data.

6.3 Implementations for Routers and Caches

We now discuss the implementations of Alg. 2, 4, and 5. We note that interest packets are typically much smaller than their corresponding data packets. Therefore, interest packets alone cannot cause severe network congestion and packet delays/losses. Based on this observation, we assume that all interest packets reach their destinations immediately. As we shall see in the next section, our implementation based on this assumption offers the optimal performance.

In Alg. 2, each router only needs to know $\sum_{s,c,v} X_v H^l_{c,v} z_{s,c,v}$ to update λ_l for its links. We note that $\sum_{s,c,v} X_v H^l_{c,v} z_{s,c,v}$ can be estimated by (the rate of interest packets going through l) $\times$ (the

size of a data packet). As the router obviously knows the rate of interest packets going through l, it can update λ_l directly without the need for requesting further information. Likewise, Alg. 4, and 5 can be carried out if one knows $z'_{s,c,v}$. As user s sends out an interest packet for the pseudo-data $p'_{c,v}$ when $z'_{s,c,v} = 1$, the network cache c and all routers between c and s can infer $z'_{s,c,v}$ by observing the presence of interest packets for $p'_{c,v}$.

7. SIMULATION

We present our simulation results in this section. All simulations are conducted on ndnSIM [22], the standard NDN simulator that is running on top of NS-3.

7.1 Simulation Set-up

We consider systems with 20 different videos. The popularity of these videos follow the Zipf distribution with parameter 1. Each video has 5 available versions with data rates of 1Mbps, 2.5Mbps, 5Mbps, 8Mbps and 16Mbps, which are the standard data rates for streaming videos of resolutions 360p, 480p,720p, 1080p and 1440p, respectively. Each video is one-hour long, and the file sizes of video versions are calculated accordingly. The size of each of network cache is 43875MB.

We consider that users can be of three types: smart phones, laptops, or televisions. The resolution of a smart phone screen is 720p. Even if a smart phone user receives a video version with resolution higher than 720p, it still only experiences 720p quality due to the limitation of its screen. Hence, we set the utility function of a smart phone user to be $20\ln(\min(X_v, 5))$. The resolution of a laptop is 1080p, and its utility function is $40\ln(\min(X_v, 8))$. Finally, the resolution of a television is 1440p and its utility function is $60\ln X_v$.

We consider two topologies as shown in Figure 7.1 and Figure 7.2. Each topology consists of a root node that stores all video versions and three network caches, each of which is connected to different group of users. The number of smart phones, laptops, and televisions in each group is marked in the figures.

We simulate and compare the following four policies:

- **Optimal**: This is the optimal solution to the CaVe-CoP problem by directly solving it as a linear programming problem.
- **CaVe-CoP**: This is our proposed solution to the CaVe-CoP problem.

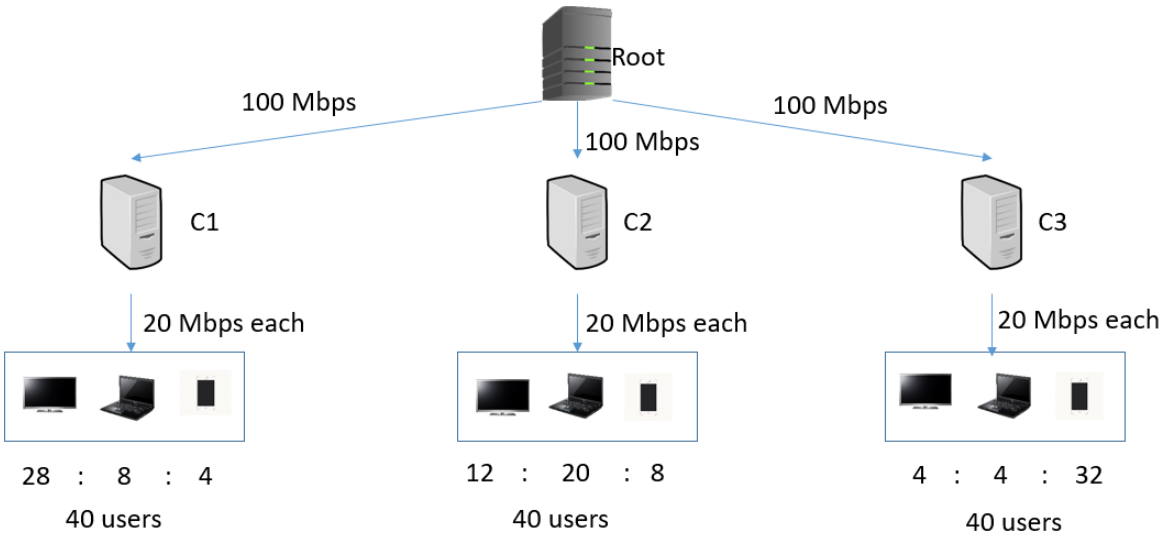

Figure 7.1: Simulation Topology 1

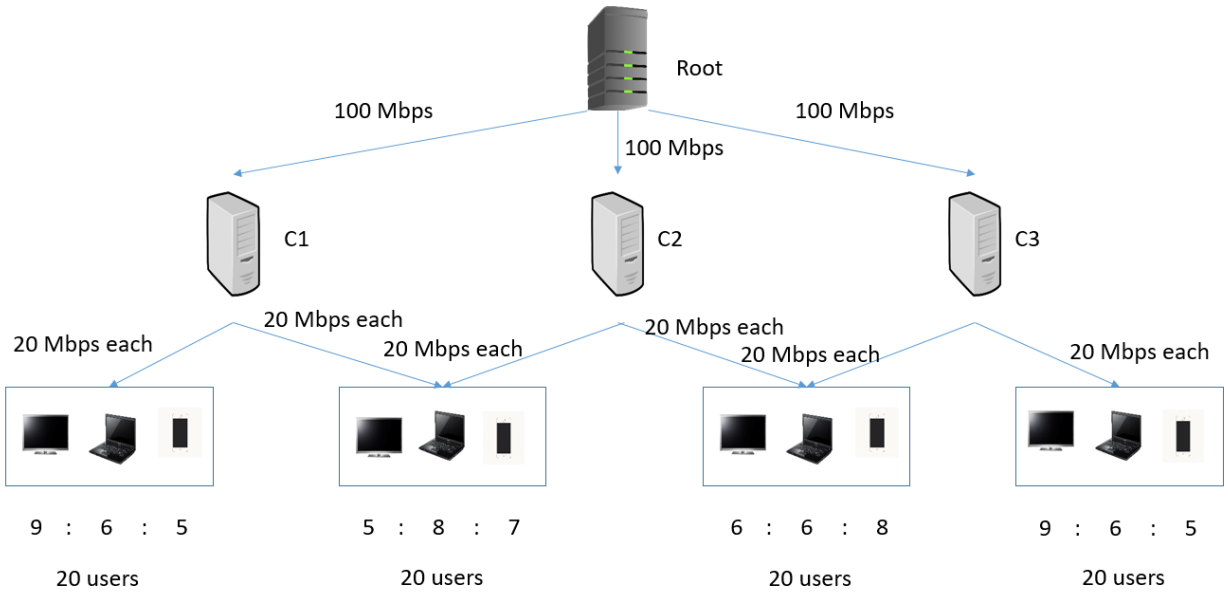

Figure 7.2: Simulation Topology 2

- **Cache All Versions**: In this policy, if a network cache stores a video version, it needs to store all versions of the same video. As a result, each network cache simply stores the most popular videos, subject to its storage constraint. This is the standard approach of CDN. Each

user employs our solution for cache-version selection. In other words, this is a policy that employs the optimal solution to CaVe and a standard but suboptimal solution to CoP.

- **Greedy Version**: In this policy, each user chooses the version that matches its screen resolution. Network caches employs our solution for content placement. In other words, this is a policy that employs the optimal solution to CoP and a standard but suboptimal solution to CaVe.

For each simulation, we use the video contents that each user actually receives to calculate two performance metrics: the utility that each user obtains and the amount of time that each user suffers from video stall in each second.

Fig. 7.3 - 7.6 shows the simulation results for the two topologies. It can be easily observed that our solution significantly outperforms Cache All Versions and Greedy Version. Moreover, our solution converges in less than 15 seconds, which suggests our solution is adaptive to network dynamics when users may change the videos they are watching.

The Cache All Versions policy performs poorly because it makes content placement decisions solely based on the popularities of videos, but has no considerations about the various versions of the same video. As users are accessing videos with a variety of devices, it becomes increasing important to treat different versions differently. For example, a network cache whose users are mostly using smart phones should not waste its storage by storing 1440p video versions.

The Greedy Version policy always chooses the version that matches users' screen resolutions. It is not adaptive to network congestion. As a result, not only does it have low utility, it also suffers from high video stall times.

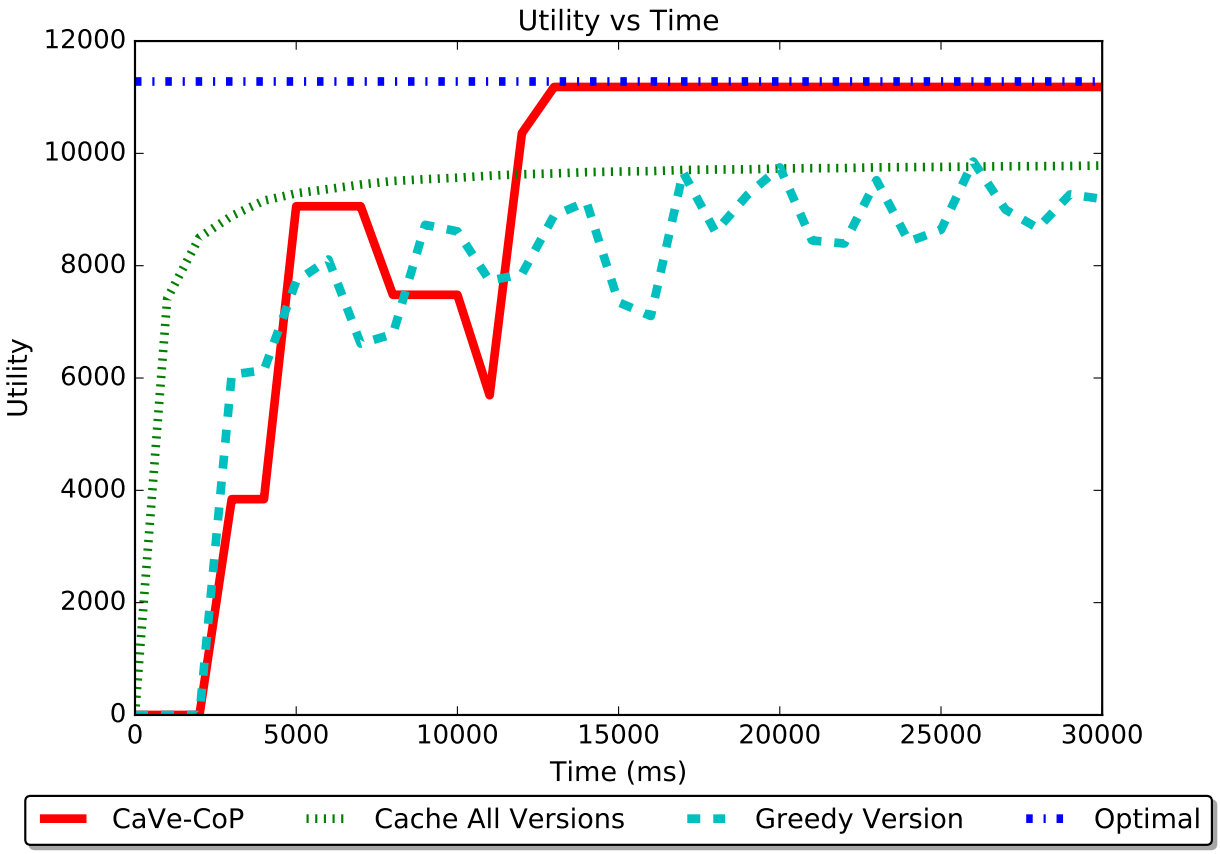

Figure 7.3: Utility vs Time Graph for Topology 1

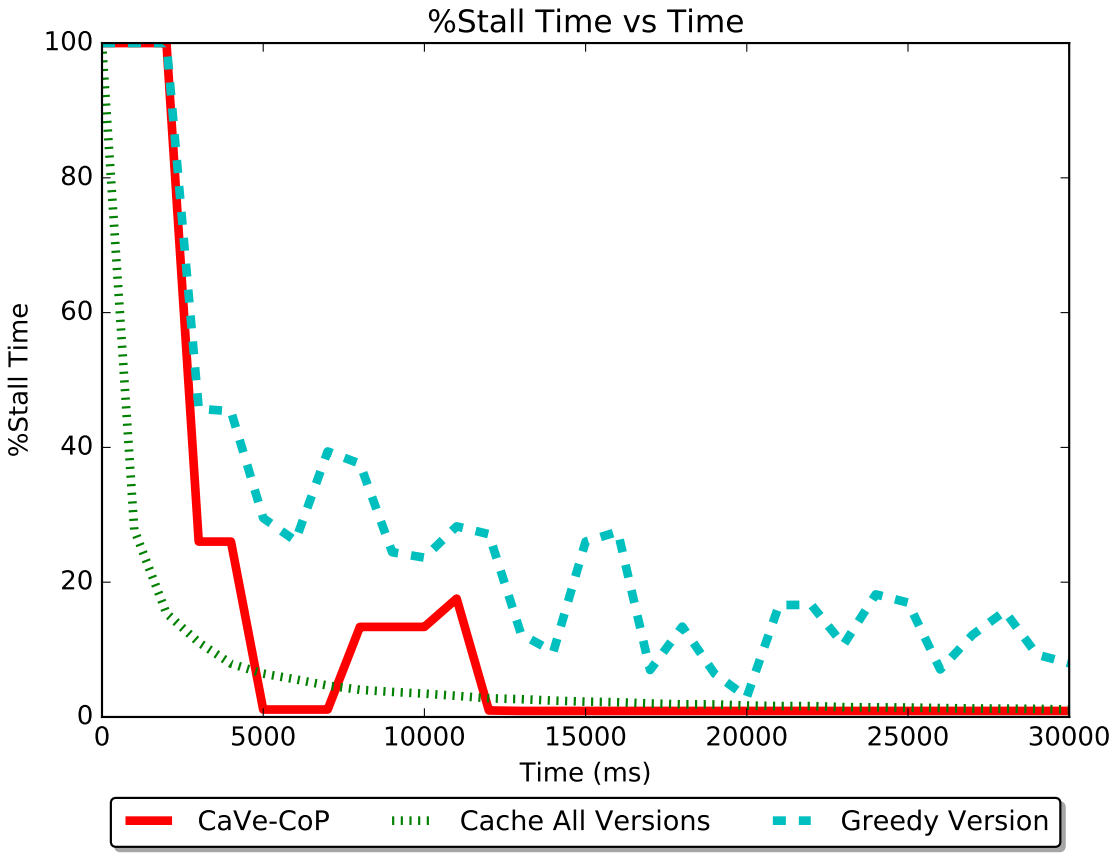

Figure 7.4: %Stall Time vs Time Graph for Topology 1

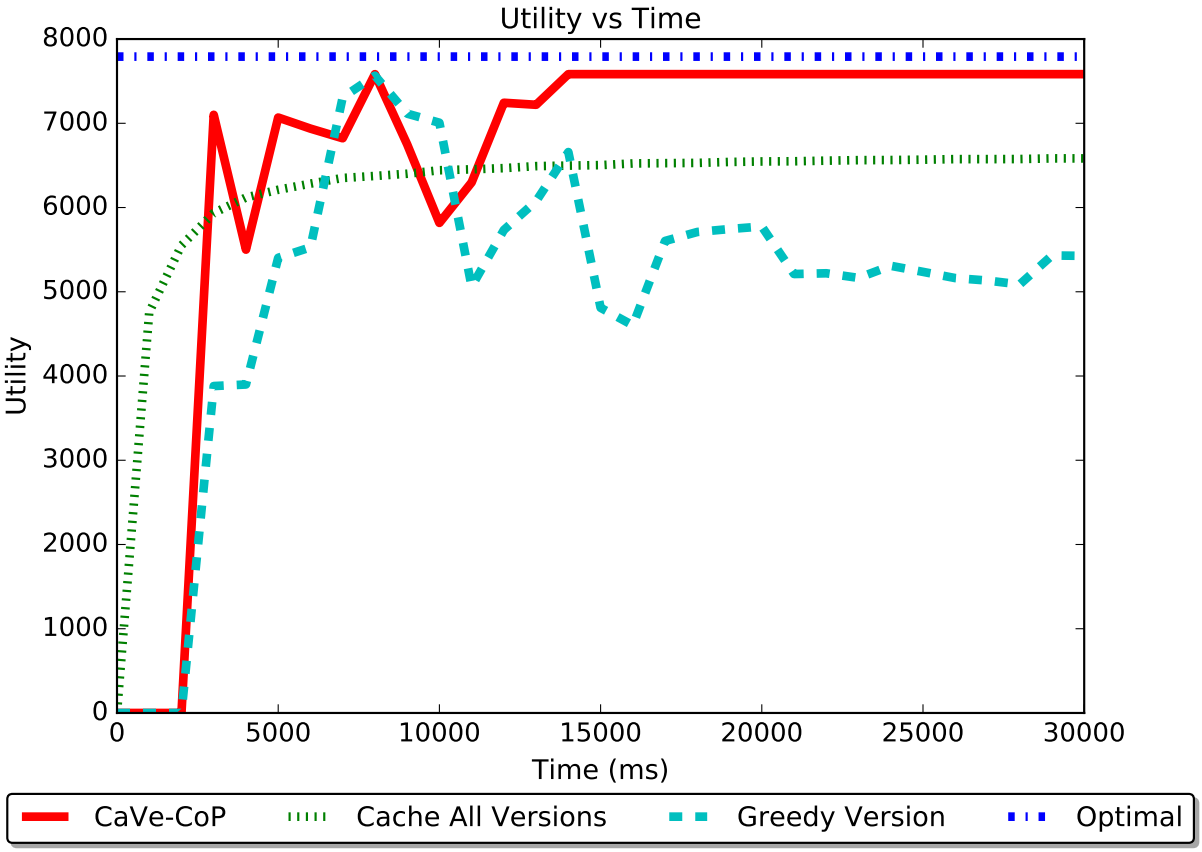

Figure 7.5: Utility vs Time Graph for Topology 2

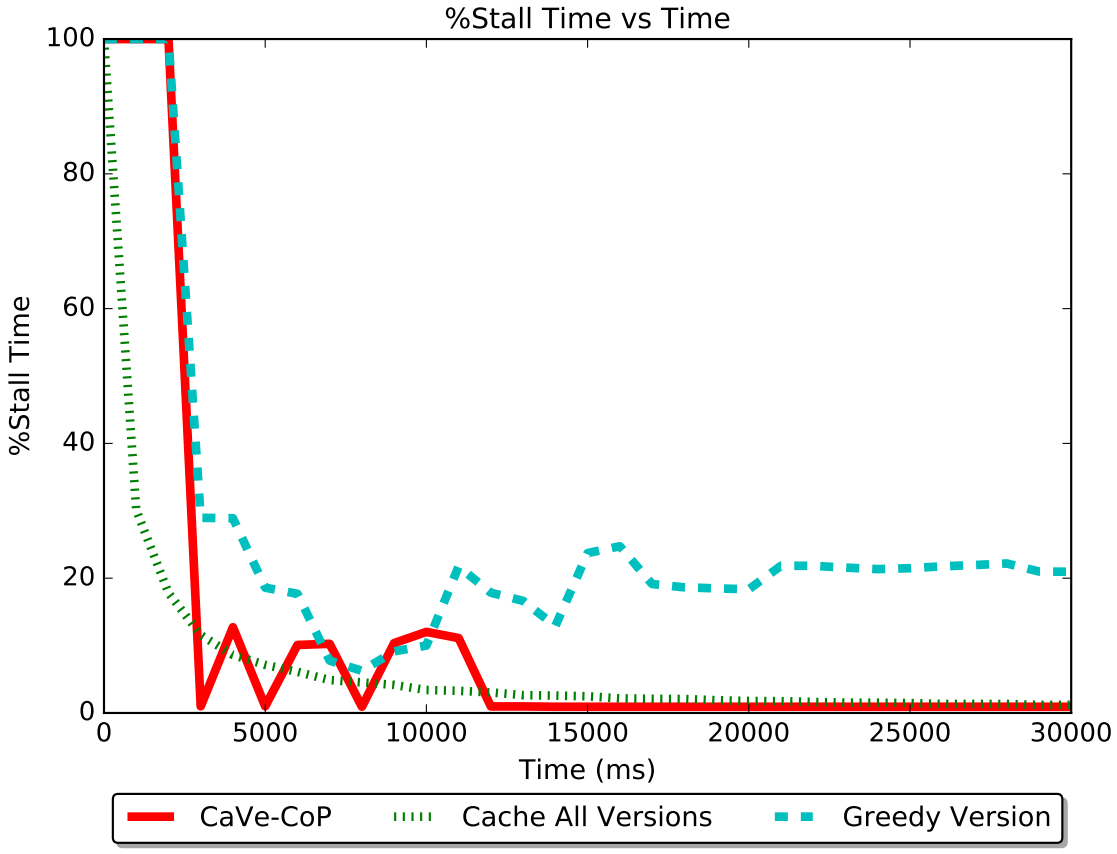

Figure 7.6: %Stall Time vs Time Graph for Topology 2

8. CONCLUSION

In this thesis we introduced the CaVe-CoP policy. Our policy addresses the cache version selection and content placement problem together. We modeled an ICN to form our optimization problem. Our solution focused on a timescale separation between the version selection variable and the content placement variable. We discussed the solution in 2 steps - first solving the cache version selection problem by keeping the contents at fixed locations. Next, we solved the content placement problem by introducing a pseudo-variable for version selection. We implemented our policy as a distributed algorithm in Named Data Networks. This implementation lets us exploit in-network caching along with a name based routing scheme. Our simulations show a better performance as compared to policies where all-version caching is done based on popularity and versions are selected greedily. Considering the perpetually rising demand for video streaming over the internet, the CaVe-CoP policy would be very useful in providing a better user experience and increasing the overall utility received from the network.

REFERENCES

[1] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, K. C. Claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, and E. Yeh, "Named data networking (NDN) project," tech. rep., Palo Alto Research Center (PARC), 2010.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. C. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 66–73, jul 2014.

[3] I. Seskar, K. Nagaraja, S. Nelson, and D. Raychaudhuri, "MobilityFirst future internet architecture project," in *Proc. 7th Asian Internet Engineering Conference (AINTEC '11)*, ACM Press, 2011.

[4] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," vol. 16, no. 2, pp. 1024–1049, 2014.

[5] 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.

[6] F. Lai, F. Qiu, W. Bian, Y. Cui, and E. Yeh, "Scaled VIP algorithms for joint dynamic forwarding and caching in named data networks," in *Proc. 2016 3rd ACM Conf. Information-Centric Networking (ICN '16)*, ACM Press, 2016.

[7] Y. Wang, W. Wang, Y. Cui, K. G. Shin, and Z. Zhang, "Distributed packet forwarding and caching based on stochastic network utility maximization," vol. 26, pp. 1264–1277, June 2018.

[8] 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.

[9] T. Ahmed, A. Mehaoua, R. Boutaba, and Y. Iraqi, "Adaptive packet video streaming over IP networks: a cross-layer approach," vol. 23, pp. 385–401, feb 2005.

[10] D. Jurca, J. Chakareski, J.-P. Wagner, and P. Frossard, "Enabling adaptive video streaming in p2p systems [peer-to-peer multimedia streaming]," vol. 45, pp. 108–114, jun 2007.

[11] L. De Cicco and S. Mascolo, "An experimental investigation of the Akamai adaptive video streaming," in *HCI in Work and Learning, Life and Leisure* (G. Leitner, M. Hitz, and A. Holzinger, eds.), (Berlin, Heidelberg), pp. 447–464, Springer Berlin Heidelberg, 2010.

[12] 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.

[13] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, "Adaptive multimedia streaming in information-centric networks," vol. 28, pp. 91–96, nov 2014.

[14] D. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," vol. 24, pp. 1439–1451, aug 2006.

[15] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, Mar. 1998.

[16] I.-H. Hou and P. R. Kumar, "Utility-optimal scheduling in time-varying wireless networks with delay constraints," in *Proc. 11th ACM Int. Symp. Mobile Ad Hoc Networking and Computing*, (Chicago, Illinois, USA), pp. 31–40, ACM, 2010.

[17] R. Deng, Y. Zhang, S. He, J. Chen, and X. Shen, "Maximizing network utility of rechargeable sensor networks with spatiotemporally coupled constraints," vol. 34, pp. 1307–1319, may 2016.

[18] R. Gupta, L. Vandenberghe, and M. Gerla, "Centralized network utility maximization over aggregate flows," in *2016 14th Int. Symp. Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2016.

[19] A. Zhou, M. Liu, Z. Li, and E. Dutkiewicz, "Joint traffic splitting, rate control, routing, and scheduling algorithm for maximizing network utility in wireless mesh networks," vol. 65, pp. 2688–2702, apr 2016.

[20] V. Ramaswamy, D. Choudhury, and S. Shakkottai, "Which protocol? mutual interaction of heterogeneous congestion controllers," vol. 22, pp. 457–469, Apr. 2014.

[21] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[22] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the evolution of ndnSIM: an open-source simulator for NDN experimentation," *ACM Computer Communication Review*, July 2017.