Rate-Selective Caching for Adaptive Streaming Over Information-Centric Networks

Wenjie Li, *Student Member, IEEE*, Sharief M.A. Oteafy, *Member, IEEE*, and Hossam S. Hassanein, *Fellow, IEEE*

Abstract—The growing demand for video content is reshaping our view of the current Internet, and mandating a fundamental change for future Internet paradigms. A current focus on Information-Centric Networks (ICN) promises a novel approach to intrinsically handling large content dissemination, caching and retrieval. While ubiquitous in-network caching in ICNs can expedite video delivery, a pressing challenge lies in provisioning scalable video streaming over adaptive requests for different bit rates. In this paper, we propose novel video caching schemes in ICN, to address variable bit rates and content sizes for best cache utilization. Our objective is to maximize overall throughput to improve the Quality of Service (QoS). In order to achieve this goal, we model the dynamic characteristics of rate adaptation, deriving caps on average delay, and propose *DaCPlace* which optimizes cache placement decisions. Building on *DaCPlace*, we further present a heuristic scheme, *StreamCache*, for low-overhead adaptive video caching. We conduct comprehensive simulations on NS-3 (specifically under the ndnSIM module). Results demonstrate how *DaCPlace* enables users to achieve the least delay per bit and *StreamCache* outperforms existing schemes, achieving near-optimal performance to *DaCPlace*.

Index Terms—Information-centric networks, dynamic adaptive streaming, throughput optimization, queueing analysis, off-path caching

1 Introduction

The growing demand for video content is reshaping our view of the current Internet. According to Cisco's Visual Networking Index (VNI), by 2019, it is projected that global Internet traffic would surpass 162 exabytes per month, where video traffic would amount to a dominating 80% in conservative estimates [1].

This dramatic growth in video demand has motivated service providers (e.g., Netflix and YouTube) to adopt HTTP-based dynamic adaptive streaming (DASH) [2]. This dynamic approach provides a time-shift control on media requests in reaction to varying bandwidth conditions experienced by each user, and automatically adapts between versions of each video to download the one with the best possible quality. That is, as the adaptation decision reacts to real-time bandwidth measurements, users would be served with the most suitable bit rate to reduce stalls in the playback.

DASH has played a major role in improving video delivery services. However, DASH relies on HTTP connection protocol which is implemented over the current host-centric network. Ultimately, service degradation will dominate unless networking primitives are built on content rather than connections. A recent shift towards Information-Centric Networks (ICN) [3] promises to intrinsically handle large-scale content dissemination, caching and retrieval. Since video delivery services are projected to be even more influential as we steer to ICNs, we argue that dynamic adaptive streaming must be addressed as an essential component in this future Internet paradigm [4].

While Content Distribution Networks (CDN) [5] and Peer-to-Peer (P2P) Networks [6] adopted overlay architectures over the Internet to enhance content delivery, ICN exploits caching as a networking primitive by equipping each router with caching capacity, instead of pre-designated, sparse and static surrogate servers. This feature of ubiquitous in-network caching is recognized as an efficient way to reduce access delay. In the context of video streaming, the impact of in-network caching is even more important since it provides an opportunity to serve users with bit rates higher than the actual link bandwidth between the producer and consumer [7]. This means that high resolutions of video content, which may be impossible to deliver without caching, will effectively become accessible by users.

ICN's inherent capacity to dynamically cache content enables novel models for provisioning different services, content and interest groups. Many research efforts have thus investigated ICN caching [8]. Typical ICN cache schemes (e.g., [9], [10], [11], [12]), which minimize hop counts or cache-miss rates, are not particularly designed for video streaming applications. These generic schemes overlook the features of variable content sizes and adaptive video request pattern, which impede the system performance.

As dynamic adaptive streaming is pivotal to handling video traffic in ICN [4], we therefore argue the challenge of catering to video content necessitates applying a caching scheme which is aware of adaptive requests. The ultimate goal of this scheme is improving Quality of Service (QoS) in

• *The authors are with Queen's University, Kingston, ON K7L 3N6, Canada. E-mail: {liwenjie, oteafy, hossam}@cs.queensu.ca.*

Manuscript received 24 Aug. 2016; revised 16 Feb. 2017; accepted 13 Mar. 2017. Date of publication 26 Mar. 2017; date of current version 15 Aug. 2017. Recommended for acceptance by N. Kato.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2017.2687920

terms of the average throughput. To the best of our knowledge, the problem of caching variable video chunks of different bit rates, without incurring intensive data processing, stands unsolved.

We address the core video caching problem by presenting a novel cache placement scheme, *DaCPlace*, which handles variable bit rates and content sizes and is generalized to scenarios where users demand adaptive bit rates, and caters to the heterogeneity of user devices and link conditions. The objective of *DaCPlace* is to minimize access delay per bit of requested video, which influences QoS in terms of throughput. In order to achieve this goal, we model the dynamic characteristics of rate adaptation—deriving caps on average delay—and propose an iterative algorithm to cache content of different sizes and popularity in order to optimally utilize caching capacity. Based on *DaCPlace*, we design a heuristic scheme, *StreamCache*, where each router makes caching decisions using local statistics exchange to achieve low-overhead cache placement. *StreamCache* also targets improving video throughput, however, with much lower computational cost compared to the optimal *DaCPlace*. We conduct comprehensive simulations and demonstrate how, under various network settings (e.g., available cache storage and popularity distribution), *DaCPlace* enables users to achieve the least access delay per bit compared with other popular caching schemes and *StreamCache* achieves near-optimal performance in contrast to *DaCPlace*.

Our contributions in this paper are threefold: 1) we develop *DaCPlace* as a benchmark caching scheme, specifically optimized for adaptive video streaming applications; 2) we present a rigorous analytical model for adapting to heterogeneous requests in future Internet applications, and map them on ICN architecture to facilitate in-depth analysis of caching performance influenced by adaptive bit-rate selection, and 3) we present our heuristic scheme, *StreamCache*, which effectively decreases the caching overhead, as a close second to optimal performance, and is designed to scale with large-scale scenarios.

The remainder of this paper is organized as follows. In Section 2, we overview related work including recent research on adaptive video caching in ICN. Section 3 describes the system upon which our designed caching algorithm is applied. Section 4 presents the formulation of our adaptive video caching problem. We elaborate on our derivations for modeling the caching process, with specific emphasis on the queueing analysis, in Section 5. Section 6 describes the design of our cache placement schemes, namely the optimal *DaCPlace* and heuristic *StreamCache* schemes. Section 7 then presents our experiment setup and performance evaluation results. We conclude in Section 8 with our final remarks on future work directions.

2 Related Work

In the realm of ICNs, several ongoing projects are pursuing the ICN vision (e.g., DONA [13], PSIRP [14] and CCN [15]). Chief among these architectures are Content-Centric Networks (CCN), which witnessed significant uptake by the research community due to rapid benchmarking tools and insightful performance analyses. In this paper, we capitalize on CCNs' prevalence and build our dynamic adaptive streaming system over it.

Content in CCN is identified with unique *Names*. Moreover, *Interest* and *Data* are two specific types of packets in CCN, where *Interest* is dispatched by consumers (users) for information and *Data* contains requested video chunk, which could be cached anywhere in the network. At the heart of CCN/ICN, a core challenge is caching such *Data* to expedite response time to *Interest* requests.

In-network caching schemes in CCN have been heavily investigated [10], [11], [12], [16]. These schemes are generally categorized into *on-path* and *off-path* caching. *On-path* caching [12], [16] only utilizes caches located on the content's routing path (towards the content provider) and *off-path* caching [10], [11] uses nodes' neighborhood caches, which requires additional collaboration with *Interest* forwarding to reach cached *Data*. Some caching policies require coordination among routers for cache placement. For example, Li et al. in [16] aimed at saving as much data-traffic as possible, based on the popularity of content. Their approach required collecting statistics on content request frequencies and caching decisions exchanged among routers. However, other approaches make the caching placement/replacement decision of each router dependent on its own information. For example, Psaras et al. presented *ProbCache* [12] where the caching probability changes over routers according to their locations on the routing path.

In addition to cache placement schemes, recent efforts in the literature argue against ubiquitous caching in ICN. For example, Fayazbakhsh et al. [17] claimed that caching at the edge of the network (closer to users) yielded almost the same performance gain as ubiquitous caching. This observation was also demonstrated via a model proposed by Dabirmoghaddam et al. in [18]. However, these contributions overlook an important issue of content redundancy due to edge caching, especially with limited cache budgets and high number of edge routers. Their performance was measured under simplistic LRU schemes, which diminishes the effect of popularity-based caching, as later demonstrated in Section 7.

To handle dynamic streaming applications, the problem of caching for variable bit rates or service prioritization of video content has also been studied. In the domain of CDN, Lee et al. [19] investigated the interaction between a user and a cache. They proposed a rate adaptation algorithm, which is aware of the impact made by cached content, to achieve smooth switching between bit rates and fairness among users. However, their contributions lie in the control on the client side, which fails to leverage the caching capabilities to improve QoS.

The effect of ubiquitous in-network caching in ICN on adaptive video delivery has recently attracted more attention. Jia et al. [20] designed a control layer for optimal *Interest* forwarding and adaptive video caching based on the virtual queue of each bit rate. Lee et al. [21] extended the principal of value-based caching schemes and applied it over different layers of bit rates to increase cache reuse ratio. Kreuzberge et al. [22] solved the challenge of unfair bandwidth sharing and proposed a cache-aware traffic shaping policy for adaptive streaming. Liu et al. [7] studied caching behavior over CCN and demonstrated that clients could be served with bit rates even higher than their actual bandwidth, emphasizing the potential of ICN caching and

the importance of designing caching schemes for adaptive video content.

Grandl et al. [23] pointed out a core challenge in ICN video caching, whereby the same content encoded in different bit rates compete for limited caching storage. Consequently, they proposed *DASH-INC*, which only caches the highest bit rate of video to avoid content redundancy, and the requests for lower bit rates would be serviced via transcoding at each router. *DASH-INC* argues that it would improve cache utilization, since only the highest bit rate needs to be cached, without storing different bit rates of the same content. Jin et al. [24] proposed a partial transcoding scheme, where each node caches all the (bit-rate) versions for popular video content, but only caches the highest bit rate for unpopular data. The objective is to find the optimal configuration between caching, transmission and transcoding. However, as transcoding has to be done on a per-request basis, one cannot simply assume that with the increasing demand for video content, in-node processing would be scalable for real-time requests. Our work tackles this problem in a radically different way by caching video content with selective bit rates on routers in an optimized manner, by pursuing bit-rate-specific popularity. We conduct simulations to demonstrate the advantage of bit-rate-selective caching over in-node transcoding.

In our earlier work [25], we addressed a video caching scheme in ICN, and proposed an optimal *on-path* caching scheme *DASCache* in a constrained system. This work is built to tackle a more general system, where both *on-path* and *off-path* caching are incorporated. Moreover, compared to [25], we present a model which further considers the effect of *Interest Aggregation* on filtering video traffic.

3 System Description

In this section, we elaborate on the system design as a building block for the optimal cache placement scheme. We focus on describing the network architecture and default patterns when users make video requests.

3.1 Network Architecture

Our system primarily targets video delivery in CCN. As video traffic dominates the Internet [1], we assume all requests in our system are made for adaptive video content. In building this system, we assume that requested videos by all users are kept in one video producer without considering any replica. This assumption is derived from the current settings of the CCN architecture, since each producer would add a unique postfix to the names of content, which thus distinguishes each content from all replicas. The system contains two different types of routers: edge and intermediate routers. All clients are served exclusively by edge routers.

Caching is not a standalone component of CCN, but highly coupled with other features, such as routing and *Interest* forwarding. It is hence essential for caching to work with routing and forwarding in a cooperative way. The routing scheme we adopt in the system is based on the Open Shortest Path First routing scheme for Named-data (OSPFN) [26], which is the most of both popular and commonly used approach in ICN. OSPFN discovers the shortest routing path from each router to the video producer, as depicted in Fig. 1,

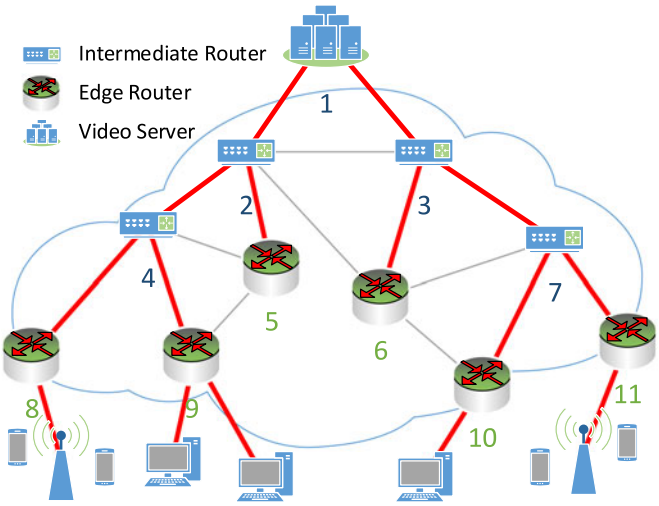


Fig. 1. Network topology. The bold lines indicate routing paths discovered by the OSPFN [26].

along which video requests can be satisfied by the cache of each node. However, we argue that the system would not reach the optimal performance when only this *on-path* caching is utilized. For example, in Fig. 1, if there is a request sent from users connecting to router 9 and the video content is not cached along the routing path to the producer but instead cached on router 8. *On-path* caching would not utilize the video content on router 8 which could respond to the request faster than the video producer. Thus, in order to benefit from *off-path* caching, in addition to the path to the video producer discovered by OSPFN, routing choices to off-path caches are also available at each router. This means that each router will be notified of caching decisions made on its downstream nodes. For example, the caching decision at router 9 should be relayed to routers 4 and 2.

According to our choice on routing scheme for optimal performance, each router may have more than one interface to forward a certain request. Rossini et al. [27] showed that the single-path forwarding which relay requests through only one interface, still achieves the lowest network load and smallest request hop counts, compared with existing multi-path forwarding schemes which send requests through multiple interfaces simultaneously. Therefore, in this system, we adhere to CCN's single-path forwarding. Each router is assumed to apply the *best route* forwarding strategy, which chooses an interface with the minimal delay, to retrieve the corresponding *Data*.

3.2 Video Request Patterns

Video request patterns can be described on two levels: file and chunk level. Requests made on the file level represent the popularity of the video content. Once users decide to watch a video file, the actual *Interest*s are tallied with the pattern on chunk level. These requests are generated sequentially, following the exact video playback.

Based on this pattern, gauging the video file request is an umbrella concept, which consists of a group of consecutive *Interest*s for content belonging to the same video file. Typically, there is no correlation between two video file requests. However, since video chunk requests are generated sequentially, two consecutive *Interest*s from the same user yield

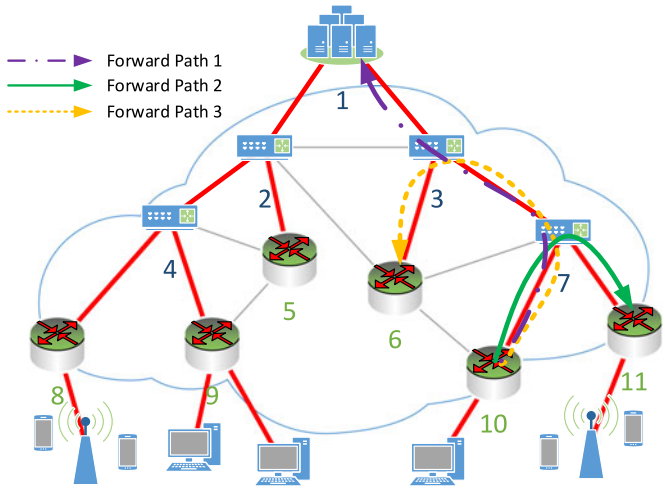


Fig. 2. The possible request delivery paths.

notable correlation. In our system, each request for a file contains a batch of requests for chunks, while the number in each batch is variable. This feature corresponds to the typical viewing behavior of most people: start playing from the beginning; keep watching for a period of time; terminate the media session when bored or out of time.

4 Adaptive Video Caching Formulation

Our *DaCPlace* system targets optimizing video access delay per bit. Cached content is updated periodically on each router, over the period of a pre-determined round. In order to optimize caching decisions, statistics are collected to capture request patterns and cater to popular videos. At the beginning of each round, operational parameters need to be derived based on the statistics from the last round and are used as inputs to *DaCPlace*; the system then updates caches based on decisions made by *DaCPlace* and refreshes the statistics, preparing for the next round.

4.1 Notations

All nodes in the network are modeled as a connected graph $G = (V, E)$ where nodes in V are composed of a set of edge nodes N and intermediate nodes I. Node j, where $j = 1, \ldots, |V|$ is equipped with content storage with capacity C_j which represents the class-specific capacity dedicated to video caching. The actual allocation of C_j has been investigated in related literature under the *cache space allocation* problem, which is well detailed and addressed by Wang et al. in [28]. We assume there is a total of F video files. For simplicity, all video files are assumed to be fragmented into the same number of chunks, denoted by K. That is, video chunk k of any file f, where $k = 1, \ldots, K$ contains the same length of playback time. There are B bit rates available for request, whereby a vector $S_{1\times B}$ denotes the size of any chunk encoded with different bit rates. For example, $S(b)$ denotes an element in vector S which represents the chunk size with bit rate b. Hence, each video chunk now is identified by a three-dimensional index (f, k, b).

π_j denotes the probability distribution of edge router j receiving video requests differed by bit rates. We use q to denote video content popularity, where q_f represents the probability of request for video file f.

Let x_j denote our *DaCPlace* cache placement decision, where $j = 1, \ldots, |V|$ and $x_j(f, k, b) \in \{0, 1\}$. Thereby a decision of $x_j(f, k, b) = 1$ indicates that video chunk (f, k, b) should be cached at node j in the next round.

The average request rate on video chunk (f, k, b) received by router j is presented by $\lambda_j(f, k, b)$, where $j = 1, \ldots, |V|$. For each edge router j, the sum of requests rates for all video content, is represented by λ'_j.

We use U_j to denote the set of possible endpoints of *Interest* packet forwarding paths starting from edge router j. Thus, the video producer is always contained in U_j for any $j \in N$. Moreover, based on the network settings described in Section 3.1, in order to utilize the *off-path* caching, any router would coordinate cached content with its upstream nodes. Therefore, a subset of edge routers may be contained in U_j. Let us take edge router $j = 10$ as an example. As shown in Fig. 2, in addition to the video producer, if the requested video chunk is also cached on routers 6 and 11, forwarding options on the corresponding upstream routers 3 and 7 would be added. As a result, there would be a total of three possible forwarding paths for a request from router 10 and $U_{10} = \{1, 6, 11\}$. However, it is important to note that other edge routers, such as routers 8, 9 and 5 are not contained in U_{10} because the *Interest* packet sent from edge router 10 would not reach them: the video producer cannot be an intermediate node in a forwarding path.

We define $L_{j,U_j(m)}$ as an array of routers on a possible *Interest* forwarding path, where $j \in N$, starting from edge node j to an endpoint m in the set U_j. For simplicity, we use $L_{j,m}$ to denote $L_{j,U_j(m)}$. We define the index of $L_{j,m}$ to start from 1. $L_{j,m}(1) = j$ and $L_{j,m}(n+1)$ denote the next-hop node to $L_{j,m}(n)$. For example, following the topology depicted in Fig. 2 and considering $j = 10$, $i = 6$, $L_{10,6}$ will be $(10, 7, 3, 6)$. We use $RT_{oh}(i, j)$ to denote the round trip delay between routers i and its one-hop neighbor j. It includes the time to deliver the *Interest* packet from i to j and the corresponding *Data* packet from j to i once the video content reaches router j. All of our system notations are summarized in Table 1.

4.2 Problem Formulation

In adaptive streaming, the throughput of video requests is used by the rate adaptation algorithm to estimate the maximum supported bit rate under the current link condition. This is typically based on measuring the round trip time (RTT) delay of the most recently requested video chunk. A user who wants to switch to a higher bit rate must achieve high throughput first. Improving the QoS is thus dependent on accurately predicting what bit rate and which video chunk would be requested over time, which is seldom possible. The problem is further compounded as users are highly sensitive to video delay. Therefore, we argue for focusing on minimizing the average access time per bit and optimize the ensuing cache placement.

The effect of cache on a router is not isolated, and must be evaluated over a forwarding path. Thus, we first define the caching indicator $\mathbb{I}$ as:

Definition 4.1. *$\mathbb{I}(f, k, b, j, m, n)$ is a binary indicator for whether the requested video content indexed by (f, k, b) is cached along the forwarding path m starting from edge router j ($j \in N$) and ending at nth router of $L_{j,m}$. Thus, $\mathbb{I}(f, k, b, j, m, n)$ is calculated using*

TABLE 1
Summary of Notations Used in the Dynamic Adaptive Streaming System

Notation	Meaning
V	Set of routers
E	Set of links
N	Set of edge routers
I	Set of intermediate routers
S	Sizes of video chunks differed by bit rates
C	Cache capacity
B	Number of available bit rates on video producer
F	Number of video files provided by producer
K	Number of video chunks in a certain file
q	Popularity distribution of video files
p	The probability of continuing watching the video
π	Stationary distribution on bit rate selection
x	Cache placement decision
λ	Average request arrival rate per video chunk
λ^{filt}	Average request arrival rate after filtering
λ'	The sum of request rate at edge router
U	Endpoints of *Interest* forwarding paths
L	Array of nodes on a forwarding path
RT_{oh}	Round trip one-hop delay
RTT	Round trip time delay of a request from an edge router
$rRTT$	Residual RTT delay of a request from any router
θ	Link bandwidth
Q	Queueing delay
μ	Average request miss/data arrival rate
ρ	Traffic load
φ	Data Packet service time

$$\mathbb{I}(f,k,b,j,m,n)=\begin{cases}1 & \text{if} \quad \sum_{i=1}^{n} x_{L_{j,m}(i)}(f,k,b)\geq 1\\ 0 & \text{if} \quad \sum_{i=1}^{n} x_{L_{j,m}(i)}(f,k,b)=0.\end{cases} \tag{1}$$

To simplify notations, we use $\mathbb{I}(j,m,n)$ to denote $\mathbb{I}(f,k,b,j,m,n)$, where f, k and b are implicit.

It is possible that the *Interest* packets never reach the endpoints of forwarding paths because *Interest*s may be satisfied by a cache in the network. Consider the case where $j=10$, $U_j(m)=1$ in Fig. 2. Delay on link segments $(7,3)$ and $(3,1)$ does not exist on the forwarding path $L_{10,1}$ once the video chunk has already been cached on router 7 or 10. $\mathbb{I}$ is determined based on the binary indicators of cache placement decisions (x) and returns **1** if the requested video chunk has been cached on the forwarding path. Our scheme evaluates all possible forwarding routes, where the endpoint m of route is either the video producer or edge router on the sibling branch. We define the following variable d to tag a certain route: if *Interest*s cannot be satisfied by all caches on this route, this route then would not be considered as an *Interest* forwarding path candidate.

Definition 4.2. *$d_j(m,f,k,b)$ defines the cost of a possible* Interest *forwarding path $L_{j,m}$ for video content (f,k,b). Calculated by*

$$d_j(m,f,k,b)=\begin{cases}0, & \text{if} \quad U_j(m) \text{ is the producer}\\ \infty\cdot(1-\mathbb{I}(j,m,|L_{j,m}|)), & \text{otherwise.}\end{cases} \tag{2}$$

To simplify notations, we similarly use $d_j(m)$ to denote $d_j(m,f,k,b)$. We highlight that $d_j(m)$ would return **0** for a forwarding path candidate, i.e., if a request for video chunk (f,k,b) reaches the video producer or is satisfied on a cache along that path. Otherwise, $d_j(m)$ is assigned ∞ which indicates that no cache could satisfy such requests and *Interest*s shall never be forwarded along that path.

Definition 4.3. *$\mathbb{E}[RTT_j^m(f,k,b)]$ is defined as the expected delay for a video request (f,k,b) on a given forwarding path m, starting from edge router j ($j\in N$), represented by*

$$\mathbb{E}[RTT_j^m(f,k,b)] = d_j(m)+\sum_{n=1}^{|L_{j,m}|-1}\mathbb{E}\Big[RT_{oh}\Big(L_{j,m}(n),L_{j,m}(n+1)\Big)\Big]\Big(1-\mathbb{I}(j,m,n)\Big). \tag{3}$$

Equation (3) sums round-trip single-hop delays on segments over a given forwarding path. Take $j=10$ and $U_j(m)=1$ in Fig. 2 as an example. Assuming that *Interest*s from router 10 cannot be satisfied by any cache along the forwarding path, $\mathbb{I}(j,m,n)$ would always return **0** for all n in $L_{j,m}$. Thus, the expected RTT should be the sum of delay on segments between routers $(10,7)$, $(7,3)$ and $(3,1)$, which are represented by $\mathbb{E}[RT_{oh}(10,7)]$, $\mathbb{E}[RT_{oh}(7,3)]$ and $\mathbb{E}[RT_{oh}(3,1)]$, respectively. $\mathbb{E}[RT_{oh}(i,j)]$ is a critical parameter and the approach to derive this value is detailed in Section 5.2.

Our system applies the best route forwarding strategy, whereby each router selects the next hop with the least data retrieval time. This feature ensures the smallest RTT to retrieve video content among all forwarding path candidates. It is defined as $\mathbb{E}[RTT_j(f,k,b)]$, formally:

Definition 4.4. *$\mathbb{E}[RTT_j(f,k,b)]$ is the expected delay of a request for a video chunk made by a user served under edge router j, computed using*

$$\mathbb{E}[RTT_j(f,k,b)]=\min_{m=1,\cdots,|U_j|}\Big\{\mathbb{E}[RTT_j^m(f,k,b)]\Big\}. \tag{4}$$

We choose to minimize the average access delay per bit, which reflects the performance of video downloading and relates to the throughput experienced by users, as outlined below

$$\min \quad \mathbb{E}[AccessTimePerBit], \tag{5}$$

$$=\sum_{j=1}^{|N|}\frac{\lambda'_j}{\sum_{j=1}^{|N|}\lambda'_j}\mathbb{E}[ATP_j], \tag{6}$$

where

$$\mathbb{E}[ATP_j]=\sum_{b=1}^{B}\sum_{f=1}^{F}q_f\pi_j(b)\frac{\mathbb{E}[RTT_j(f,b)]}{S(b)}, \tag{7}$$

and

$$\mathbb{E}[RTT_j(f,b)]=\sum_{k=1}^{K}(1-p)p^{k-1}\mathbb{E}[RTT_j(f,k,b)], \tag{8}$$

Subject to

$$\sum_{b=1}^{B}\sum_{f=1}^{F}\sum_{k=1}^{K}S(b)x_j(f,k,b)\leq C_j,\forall j\in V. \tag{9}$$

4.2.1 Objective

The access delay per bit of video is defined in Equation (5), where the $AccessTimePerbit$ (ATP) of a request made by any user, for a chunk k, in video file f, encoded with bit rate b, is denoted by

$$AccessTimePerbit(f,k,b) = \frac{RTT(f,k,b)}{S(b)}. \quad (10)$$

Equation (6) distinguishes the performance of users in the network. $\mathbb{E}[AccessTimePerbit]$ is expanded as the weighted average of ATP experienced by users grouped by edge router. Equation (7) then expands the intermediate result $\mathbb{E}[ATP_j]$ according to video files and encoded bit rates, where video file f is requested with probability q_f and request for video content encoded with bit rate b is received on the edge router j with probability $\pi_j(b)$. $\mathbb{E}[RTT_j(f,b)]$ is further detailed in Equation (8), considering video requests by chunk sequence. In Section 3.2, we described the most common pattern when people watch videos. Under this setting, the probability of requesting kth chunk follows the Geometric distribution, $P(X=k) = (1-p)p^{k-1}, k \geq 1, k \in \mathbb{Z}$, where p represents the probability for a user to continue watching the next chunk.

4.2.2 Constraints

It is important to note that bandwidth is not incorporated as a constraint in our system since dynamic adaptive streaming already takes bandwidth into account. For example, if an increase in the number of users drives the bandwidth allocations down, while we hold the selected bit rates for video content, more and more video packets will get queued or dropped in the network. That is, dropping will occur when requests made for a certain video quality cannot be sustained by the network. Many drops would yield low or even zero throughput. The rate adaptation algorithm will then be triggered to adjust and request lower bit rates, which will reduce the amount of data delivered in the network and effectively relieve the load.

Each router has limited cache capacity dedicated to video streaming applications. Since video chunks in our system span equal-lengths of playback time, the size of a chunk is identified by the encoded bit rate, thus we use $S(b)$ instead of $S(f,k,b)$ to represent chunk size. The constraint on cache capacity is represented using (9). The total size of cached content on router j (i.e., when $x_j(f,k,b)$ returns **1**) should not exceed the available cache capacity C_j.

5 Caching Model

To solve the cache optimization problem, we hereby elaborate on deriving the expected access delay per bit, and ensuing expressions on expected round trip time. Since the cache capacity of each router j (i.e., C_j) and video chunk size ($S(b)$) are input constants set by the network operator, this section details the approach to calculate the core parameters (i.e., $q_f, \lambda'_j, \pi_j(b)$ and $\mathbb{E}[RT_{oh}(i,j)]$).

As mentioned in Section 4, our system collects statistics on video requests. This information is then used to derive the popularity distribution (q_f) on video file and request arrival rates (λ'_j) received by router j. The probability of continuing to watch the next video chunk (p) is then determined using maximum likelihood estimation, since video chunk requests are modeled to follow the Geometric distribution.

We hence focus on the other two parameters, namely 1) stationary distribution on bit rate selection $\pi_j(b)$ and 2) expected round trip one-hop delay $\mathbb{E}[RT_{oh}(i,j)]$.

5.1 Users Bitrate Selection

In addition to selecting a certain video file and chunk, a video request sets which bit rate is preferred based on the decision made by the rate adaptation algorithm. However, since link conditions are not always stable, the chosen bit rate may change for each request depending on the current link condition. While the most recent bit rate choice reflects the current link condition, the bit rate choice of the next request is influenced only by its preceding request. Therefore, this process inherently satisfies the Markov property.

In order to represent the dynamic characteristics of bit rate selection, we model the process of adaptive video requests for different bit rates as a discrete-time Markov chain. The available bit rates (versions) of videos construct the state space of this chain. The process of rate adaptation, which switches from one bit rate to another, is equivalent to the transition between two states.

A stationary distribution can represent the dynamics of a Markov chain visiting states in a static way. As to this Markov chain described by users' bit rate selection, there exists a unique stationary distribution due to the following properties:

- *Finite State Space*: the number of states equals to the available bit rates, which is finite.
- *Time-homogeneity*: the rate adaptation algorithm would repeat the same decision when it faces the same network condition.
- *Irreducibility*: any bit rate could be selected for the next video request; the entire state space is a single communicating class.
- *Positive Recurrence*: the time interval between selections of the same bit rate is expected to be finite since the current link condition repeats to appear.

This stationary distribution reveals the probability for available bit rates to be selected by rate adaptation algorithm during the process of video request. Then, we could use this distribution to represent $\pi_j(b)$ in the optimization formulation. $\pi_j(b)$ demonstrates users' selection for a period of time without focusing on the real-time changes on bit rates made by the adaptation algorithm.

5.2 Expected Delay Derivation

In order to solve the optimal cache placement problem, delay analysis is essential to evaluate the performance of a caching scheme on video content delivery. Generally, there are four types of network delay: processing, propagation, transmission, queueing. We assume the video server and users are under the service of the same ISP, as a result, the propagation delay is relatively small and not considered. Processing delay is also omitted because of the *O(1)* hashing technique of lookup on unsatisfied requests [15]. Thus, to compute $\mathbb{E}[RT_{oh}(i,j)]$ in Definition 4.3, we only consider the transmission and queueing delays. The size of *Interest* packet is small since it mainly contains the name of the

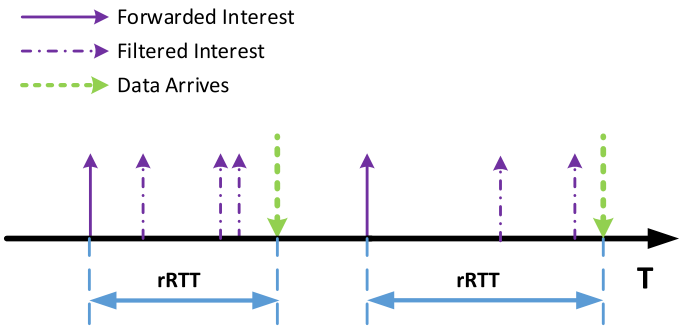


Fig. 3. Filtering effect of interest aggregation. The length of rRTT changes according to real-time network condition.

requested content so that the transmission delay of *Interest* packet is also omitted. Let θ_{ij} denote the bandwidth of link over which the *Data* packet is delivered from router i to j. We use $\mathbb{E}[Q_{ij}]$ to represent the average queueing delay at router i when the packet is delivered to router j. When video chunk encoded with bit rate b is transmitting, the expected round trip delay, $\mathbb{E}[RT_{oh}(i,j)]$, is calculated using

$$\mathbb{E}[RT_{oh}(i,j)] = \frac{S(b)}{\theta_{ij}} + \mathbb{E}[Q_{ij}] + \mathbb{E}[Q_{ji}]. \tag{11}$$

We detail the derivation of queueing delay, $\mathbb{E}[Q_{ij}]$, in Section 5.2.2.

5.2.1 Interest Aggregation

Quite often, large numbers of duplicate requests are witnessed in a short time frame for the same video. Under the current host-centric architecture, independent communication between a user and a producer must be maintained, which consumes a lot of resources (e.g., bandwidth) to repeatedly deliver the exact same content. CCN remedies this inefficiency via *Interest* aggregation, where each router keeps track of unsatisfied requests and discards duplicate ones to prune unnecessary traffic. When a new *Interest* packet arrives at a certain router, not only would it be forwarded to the next hop but also the name of the content in that packet would be recorded. Next time, when an *Interest* packet for the same content arrives before the corresponding *Data* is sent back to the router, this duplicate request would be discarded.

The video request pattern explained in Section 3.2 could be generalized as a batch renewal process [29]. We define $rRTT_j(f,k,b)$ as the residual round trip time delay of a router j, which represents the time interval between forwarding the request and receiving the corresponding data. Thus, there exists a filtering effect on requests received by any router, which is shown in Fig. 3.

Our fundamental goal is to determine optimal video caching by estimating the round trip time delay, not to model video traffic over CCN. Due to the complexity of analyzing the superposition of two renewal processes, we argue that a good approximation is critical to guiding the cache placement decision.

Carofiglio et al. [30] proposed an approximation, however, based on a fundamental assumption that once a video request is filtered, the following requests for the rest of video chunks in the same file will also be filtered. This assumption does not apply in the general scenario, since RTT depends on the cache placement and varies for video chunks of different bit rates. Thus, we analyze the *Interest* aggregation specifically on a chunk level. The process of requests after filtering is approximated by a Poisson Process.

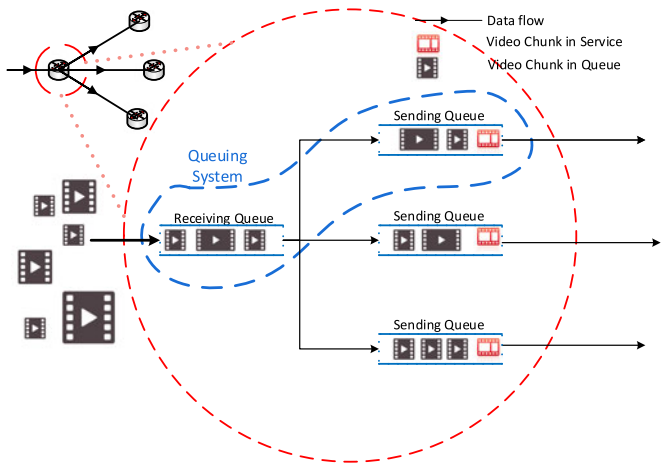


Fig. 4. Queueing model for adaptive streaming.

The action of filtering is captured as *thinning* in our model, with probability $p_j^{filt}(f,k,b)$, calculated by

$$\begin{aligned} p_j^{filt}(f,k,b) &= P(\text{interval time} > \mathbb{E}[rRTT_j(f,k,b)]) \\ &= e^{-\lambda_j(f,k,b)\mathbb{E}[rRTT_j(f,k,b)]}. \end{aligned} \tag{12}$$

This approximation could be understood as: The requests of the process after filtering are independently picked from the original process following p_j^{filt} such that, the interval of two subsequent requests is statistically guaranteed to be larger than $rRTT$. Thus, the average rate after filtering is modified using

$$\lambda_j^{filt}(f,k,b) = p_j^{filt}(f,k,b)\lambda_j(f,k,b). \tag{13}$$

5.2.2 Queueing Delay Analysis

For each video delivery path through a router, we model the queueing system to consist of the *Receiving Queue* and the corresponding *Sending Queue* on that path, as shown in Fig. 4. Both queues are assumed to be FIFO queues and are dedicated to serve streaming packets for ensuring QoS. The *Receiving Queue* dispatches the *Data* packet to a corresponding *Sending Queue* and that is where the queueing delay occurs.

As the process of *Interest*s received by any router for a certain video chunk is already approximated by independent Poisson Process, *Data* packets are assumed to follow the same process as the *Interest*, with the average input rate equal to the *Interest* arrival rate. The job service time of a *Data* packet is determined by its corresponding size. In adaptive streaming, each file is chopped into chunks, where each chunk contains the same length of playback time. The size of a *Data* packet, which contains a single video chunk, thus only varies according to its encoded bit rate. The number of bit rates a video producer serves is limited and each bit rate corresponds to one class of *Data* packets, sharing the same job service time. Based on the above characteristics of the adaptive streaming traffic, we adopt the multi-class M/G/1 model. This model can also be applied for general streaming traffic, where the size of packets can only vary within a set of values.

As explained in Section 4.2.2, the system load ρ_{ij} is guaranteed to satisfy $\rho_{ij} < 1$. We make the following proposition to derive $\mathbb{E}[Q_{ij}]$.

Proposition 5.1. *Following the multi-class M/G/1 model, the average queueing delay for packets delivered from router i to j, $\forall j \in V$, is given by*

$$\mathbb{E}[Q_{ij}] = \frac{1}{2}\frac{\sum_{b=1}^{B}\mu_j(b)S(b)^2\theta_{ij}^{-2}}{1-\sum_{b=1}^{B}\mu_j(b)S(b)\theta_{ij}^{-1}}. \tag{14}$$

Proof. Following the superposition property of an Independent Poisson Process [31], the rate of *Interest* packet for video chunks encoded with bit rate b missed by caches in router j, $\mu_j(b)$, is given by,

$$\mu_j(b) = \sum_{f=1}^{F}\sum_{k=1}^{K}\lambda_j^{filt}(f,k,b)(1-x_j(f,k,b)). \tag{15}$$

Since we are focusing on the queueing system on router i where packets are delivered to router j, the miss rate of *Interest* packet achieved in Equation (15), $\mu_j(b)$, for $b = 1,\ldots,B$, is the input rate of *Data* packet to the queueing system. If the requested video chunk is cached on router j (which means $x_j(f,k,b) = 1$), the corresponding request could be immediately satisfied and there will be no need to forward this request.

The expected job service time of data packets encoded with bit rate b on router i i.e., $\mathbb{E}[\varphi_{ij}(b)]$, is given by

$$\mathbb{E}[\varphi_{ij}(b)] = \varphi_{ij}(b) = \frac{S(b)}{\theta_{ij}}, \tag{16}$$

and then, the system load ρ_{ij} is,

$$\rho_{ij} = \sum_{b=1}^{B}\rho_{ij}(b) = \sum_{b=1}^{B}\mu_j(b)\mathbb{E}[\varphi_{ij}(b)]. \tag{17}$$

Since $\rho_j < 1$, the input rate to the queue is less than the output rate. Hence, the queueing model is not overloaded, i.e., the requested resource does not exceed the maximum that the network can provide for the long term.

We apply Little's Theorem and *Pollaczek-Khinchin (P-K)* formula [31] to calculate queueing delay. $\mathbb{E}[Q_{ij}]$ is given by,

$$\mathbb{E}[Q_{ij}] = \frac{\mathbb{E}[RS_{ij}]}{1-\rho_{ij}}, \tag{18}$$

where RS_{ij} denotes the Residual Service time, which is the remaining time seen by the new packet when it arrives in the queueing system until the current in-service packet is complete. We extend the derivation of average delay of the general M/G/1 model in [31] since the service time of packets differs by bit rates in our model. Consider a interval $[0,t]$

$$\mathbb{E}[RS_{ij}] = \frac{1}{t}\sum_{b=1}^{B}\sum_{i=1}^{M_b(t)}\frac{1}{2}\varphi_{ij}(b)^2, \tag{19}$$

where $M_b(t)$ denotes the number of packets, encoded with bit rate b, which complete their services during $[0,t]$. As $t \to \infty$, we have

$$\begin{aligned}\mathbb{E}[RS_{ij}] &= \frac{1}{2}\sum_{b=1}^{B}\lim_{t\to\infty}\frac{M_b(t)}{t}\cdot\lim_{t\to\infty}\frac{\sum_{i=1}^{M_b(t)}\varphi_{ij}(b)^2}{M_b(t)} \\ &= \frac{1}{2}\sum_{b=1}^{B}\mu_j(b)\mathbb{E}[\varphi_{ij}(b)^2].\end{aligned} \tag{20}$$

This is due to the service time of *Data* packets, $\varphi_{ij}(b)$, shown in Equation (16), is a constant value. Then, we have $\mathbb{E}[\varphi_{ij}(b)^2] = \mathbb{E}[\varphi_{ij}(b)]^2$. Therefore, synthesizing Equations (16-20), the expected queueing delay is

$$\mathbb{E}[Q_{ij}] = \frac{1}{2}\frac{\sum_{b=1}^{B}\mu_j(b)S(b)^2\theta_{ij}^{-2}}{1-\sum_{b=1}^{B}\mu_j(b)S(b)\theta_{ij}^{-1}}.$$

□

6 Video Cache Placement Schemes

To solve the video cache placement problem we first present an iterative algorithm, *DaCPlace*, which solves the optimal cache placement. However, considering the complexity of this approach as we scale to larger topologies and settings, we present a heuristic scheme, *StreamCache*, which is a real-time solution. *StreamCache* operates locally at each router, generating caching decisions without requiring global knowledge of the network, thereby serving scalability and minimal control-overhead goals.

6.1 *DaCPlace*: Optimal Scheme

A circular dependency exists between $\mathbb{E}[RT_{oh}]$ and caching placement decisions (x). In remedy, we devise *DaCPlace* to iterate, by updating parameters and caching decisions reciprocally, solving each iteration with Mixed Integer Linear Programming (MILP), to reach the optimal solution.

In Section 4, we presented the general procedure of our caching placement scheme, highlighting that caching decisions are made periodically according to the most recent bit-rate-specific statistics on video popularity. In order to differentiate between the iteration for popularity updates and for solving the optimal cache placement, we refer to the popularity update in Section 4 as the *outer iteration*, and the iteration within *DaCPlace* as the *inner iteration*.

We make the equivalent transformations to the original optimization objective by augmenting artificial variables to represent the choice for best *Interest* forwarding, and utilizing the property of routing topology to replace the nonlinear Definition 4.1. These transformations are detailed in Appendix A. However, the optimization objective after transformations is still nonlinear because of the dependency between cache placement (which decides $\mathbb{I}(j,m,n)$) and the delay $\mathbb{E}[RT_{oh}]$, as shown in Definition 4.3. The cache placement would influence the traffic load on each link which in result changes the round trip delay. We thus devised the *DaCPlace* algorithm which iteratively updates the delay and corresponding caching decision. Generally, this algorithm uses the cache placement decision of the last iteration to adjust the data input rate, which thereby yields an $\mathbb{E}[RT_{oh}]$ value, which is then plugged into the current iteration to update the cache placement decision. Thus, in each iteration, the formulation is in linear form which could be solved as a MILP. Since minimizing the video access delay is the optimization objective, the result of MILP ensures the delay

is non-increasing. Our simulations also show the execution of DaCPlace converge 100%.

Algorithm 1 details the steps to compute the optimal cache placement. For readability, the index (f, k, b) of variables are omitted and all steps should be repeated for each index. This algorithm describes the *inner iteration* we mentioned which updates the cache placement variable x. The iteration stops once predefined criterion (γ) is met, as shown in line 2, which measures the performance difference of two consecutive cache placements based on the optimization objective. These criterion should be set by the network provider considering the tradeoff between cache performance and time to achieve it. Each *inner iteration* is composed of two parts. Lines 3-23 calculate the expected delay and lines 24-25 update the cache placement result by solving a MILP problem.

Algorithm 1. *DaCPlace*

Input: Request rate (λ), Video chunk size (S), Bandwidth of links (θ), Threshold of performance difference (γ), Queueing delay convergence condition (ϕ);
Output: Cache placement x;

1 $x \leftarrow 0$; $\mathbb{E}[Q] \leftarrow 0$; // Initialization
2 **while** $|OBJ(x) - OBJ(x_{last})| > \gamma$ **do**
3 $\Delta\mathbb{E}[Q] \leftarrow \infty$;
4 **while** $\Delta\mathbb{E}[Q] > \phi$ **do**
5 **for** $\forall (i,j) \in E$ **do**
6 $\mathbb{E}[RT_{oh}(i,j)] \leftarrow Delay(S, \theta_{ij}, \mathbb{E}[Q_{ij}])$;
7 **end**
8 **for** $\forall j \in N$ **do**
9 $m \leftarrow$ shortest forwarding path from j;
10 $\lambda_{L_{j,m}(1)} \leftarrow \lambda_j$;
11 **for** $i = 1 \ldots |L_{j,m}|$ **do**
12 $p^{filt}_{L_{j,m}(i)} \leftarrow UpdateFilter(\lambda_{L_{j,m}(i)}, \mathbb{E}[RT_{oh}(i,j)])$;
13 $\lambda^{filt}_{L_{j,m}(i)} \leftarrow UpdateRate(p^{filt}_{L_{j,m}(i)}, \lambda^{filt}_{L_{j,m}(i)})$;
14 $\mu_{L_{j,m}(i)} \leftarrow MissRate(\lambda^{filt}_{L_{j,m}(i)}, x_{L_{j,m}(i)})$;
15 $\lambda_{L_{j,m}(i+1)} \leftarrow \mu_{L_{j,m}(i)}$;
16 **end**
17 **end**
18 **for** $\forall j \in V$ **do**
19 $\mathbb{E}[Q_{ij}] \leftarrow Queueing(\mu_j, S, \theta_{ij})$;
20 Update Difference $\Delta\mathbb{E}[Q_{ij}]$;
21 **end**
22 $\Delta\mathbb{E}[Q] \leftarrow \max\{\Delta\mathbb{E}[Q_{ij}]\}$;
23 **end**
 /*MILP($\mathbb{E}[RT_{oh}]$) solves the Mixed Integer Programming problem*/
24 $x_{last} \leftarrow x$;
25 $x \leftarrow MILP(\mathbb{E}[RT_{oh}])$;
26 **end**

Lines 5-17 update the video request rate received by each router. Lines 5-7 calculate the round trip delay of each link based on the Equation (11) which contains the queueing delay; as to each edge router, line 9 selects the best forwarding path with the shortest delay based on the calculation in line 6; and lines 11-16 then derive the video request rate on a particular forwarding path. At each hop, lines 12-13 calculate the filtered request rate caused by *Interest Aggregation* according to Equations (12) and (13). The request rate at next hop is then calculated based on the cache placement in the last iteration as shown in lines 14-15. Lines 18-21 update the queueing delay at each router, based on Equation (14).

6.2 *StreamCache*: Heuristic Scheme

The complexity of *DaCPlace* algorithm arises from the fact that *inner iteration* is required to update the cache placement decision, and in each iteration, a MILP formulated problem must be solved which is NP-hard.

Thanks to the architecture of CCN, routers in the network can keep track of *Interest* or *Data* packets in real time. Thus, instead of building a queueing model to derive the expected round trip delay, we utilize the forwarding table of routers, adding a timestamp to each record (*Interest*) and calculate the actual delay when corresponding *Data* packet arrives. Moreover, the heuristic design applies greedy selection, which works distributedly on each router, avoiding the *inner iteration* and MILP formulation. Therefore, based on these two changes, we design *StreamCache*,[1] which effectively reduces the complexity and overhead.

StreamCache still needs to update its cache decision periodically based on the bit-rate-specific popularity. Each router collects statistics and derives parameters (e.g., file popularity distribution (q), bit rate selection (π) and so on), which are needed in the *Cache Utility* ($\mathbb{U}$) function. For video chunk (f, k, b), the *Cache Utility* function is denoted as

$$\mathbb{U}(f,k,b) = \frac{q(f)p^{k-1}\pi(b) \cdot rRTT(b)}{S(b)}. \tag{21}$$

The cache utility reflects the contribution of each video chunk towards throughput calculation, and the cache placement decision is made based on this utility value using greedy selection. This utility value should be calculated for all video chunks on each router. *StreamCache* sorts these values and caches video chunks with high utility.

The greedy selection is based on an important observation: as to any video file(f) encoded with bit rate (b), if the αth video chunk is chosen to be cached, any video chunk k in the same file, where $1 \leq k < \alpha$, must have already been cached as well. It is because the caching utility for those video chunks is larger than the αth chunk based on the Equation (21) and they should have been cached prior to the αth chunk if there is enough cache capacity. Therefore, each router simply maintains a record of the maximum sequence number to indicate those video chunks which have already been cached however with smaller sequence numbers.

Let W denote the set of all video chunks, and $H_j(i)$ denote the video chunks cached on router i, which is the child node of router j. Our heuristic runs first on child nodes, and then delivers the caching result $H_j(i)$ to parent router j, following the default forwarding path. In order to cache those video content which are commonly requested by users, the available caching space for parent router j is thus $W - \bigcup_i H_j(i)$. Thus, video chunks already cached by sibling nodes are not considered by the parent router again,

1. An earlier embodiment of *StreamCache*, which only considered on-path caching, was presented in the IEEE International Conference on Communications (ICC), 2016 [32].

and requests for those chunks may be redirected, instead of being forwarding to the video producer, in order to utilize *off-path* caching.

Algorithm 2 details our *StreamCache* scheme. Lines 2-10 aggregate the caching decisions on router j from its downstream nodes by calculating the maximum sequence number in order to serve the common interests. Suppose the data structure like hash table is implemented which supports searching with $O(1)$ cost. The complexity of merging two decision tables is $O(FB)$ as the size of cache decision table is proportional to the number of bit rates and video files. Line 11 sorts the caching utility for all video chunks from the largest to smallest, where the complexity can be $O(FB\ log(FB))$ by applying quick sort. Lines 12-21 utilize the greedy selection to fill the cache capacity of router j, which scans the sorted table with complexity $O(FB)$. Thus, the overall complexity of *StreamCache* is $O(FB\ log(FB))$.

Algorithm 2. *StreamCache*

Input: Collected statistics ($stat_j$), Video chunk size (S), Cache capacity (C_j);
Output: The cache decision update, ΔCD_j;
1 $U_j \leftarrow$ CalculateUtility($stat_j$, S);
2 **for** $\forall i \in Downstream(j)$ **do**
3 **for** $\forall e \in CD_i$ **do**
4 **if** $e \notin CD_j$ **then**
5 Insert e in CD_j;
6 **else if** $CD_j(e).ChunkSeq < e.ChunkSeq$ **then**
7 $CD_j(e).\mathrm{ChunkSeq} = e.\mathrm{ChunkSeq}$;
8 **end**
9 **end**
10 **end**
11 $U_j \leftarrow$ Sort(U_j);
12 **for** $\forall u \in U_j$ **do**
13 **if** $C_j - Size(u.b) \geq 0$ **then**
14 **if** $(u.f, u.b) \notin CD_j \vee u.k > CD_j(u).ChunkSeq$ **then**
15 $\Delta CD_j = \Delta CD_j \cup \{u\}$;
16 $C_j = C_j - Size(u.b)$;
17 **end**
18 **end**
19 **end**

7 Performance Analysis

In this section, we evaluate the performance of the *DaCPlace* and *StreamCache* schemes under various experimental settings. We build our simulation environment over the ns-3 based simulator, ndnSIM [33].

7.1 Simulation Setup

To evaluate *DaCPlace*, the simulation is composed of two phases. The first phase calculates the optimal cache decisions using the Gurobi [34] to solve the MILP. The second phase simulates a CCN via ndnSIM, applying the caching decisions achieved in the first phase. To evaluate *StreamCache*, we implement the distributed algorithm on each router directly in ndnSIM.

The settings of the experiments mimic the dynamic adaptive streaming application where requests are generated for different bit rates of videos. We consider four common available bit rates: 250 Kbps, 400 Kbps, 600 Kbps and 900 Kbps. Without loss of generality, these four bit rates are typical viewing bit rates, which correspond to representative video quality levels [35]. We compare the performance of our proposed video caching schemes with three other cache placement schemes in the literature: *Cache Everything Everywhere (CE2)* [15], *ProbCache* [12] and the *on-path* optimal video caching scheme, *DASCache* [25]. We choose *CE2* as it is a commonly used baseline (e.g., [9], [36]). *ProbCache* is a popular approach in the literature (seen in [9], [11]) because of its effectiveness of reducing server hit ratio. *DASCache* is an optimal scheme which only utilizes the cache on the default forwarding path and we use it to compare with the performance of our proposed schemes in this more general system where both *on-path* and *off-path* caching are considered.

As explained in Section 6, *DaCPlace* and *StreamCache* require an *outer iteration* to update bit-rate-specific popularity. Our experiments only simulate one round in *outer iteration* since popularity and bit rate distributions remain as control parameters in the simulations. After this round the cached video content decided by *DaCPlace* or *StreamCache* would not be replaced, and then we start collecting statistics for performance evaluation. In order to make a fair comparison, simulation on *CE2* and *ProbCache* mimics this procedure and we disable cache replacement (*LRU*) before the evaluation.

As our *DaCPlace* scheme targets optimizing QoS in terms of throughput, we choose the *Access Time Per Bit* of all users in the system as a performance metric. The delay is measured between the *Interest* packet sending and corresponding *Data* packet arriving at the user's device. Another metric we choose is *Cache Hit*. This metric is commonly used to evaluate the performance of a caching system.

The routing scheme applied in our system is based on OSPFN. It generates a routing tree topology. At the same time, a tree is instructive because from the perspective of a video producer, the distribution topology is effectively a tree. Thus, in the simulations, we adopt a tree topology directly, with one layer of edge routers as leaf nodes and at least one layer of intermediate routers which connect to the producer (root). More routers between these two layers will only generate topologies with larger tree heights. We chose 20 nodes in our simulations to contrast performance results, as similar performance trends were observed for different network sizes.

7.2 Simulation Parameters

The parameters related to users' requests in the system are collected at each router. In the simulations, we use the following rules for choosing these parameters.

1) The average rate of video requests (λ) received by any edge router is chosen randomly, with the only constraint that the incurred load is less than the link capacity.
2) Any video request specifies the file index, chunk index and bit rate. The abstract requests for video files are determined by content popularity distribution (q). In the simulations, we use the Zipf-like distribution [37] where the probability of requesting the fth file is $q_f = \beta f^{-\alpha}$, $\beta = (\sum_{f=1}^{F} f^{\alpha})^{-1}$. The parameter

TABLE 2
Default Simulation Parameters

Parameter	Value
Number of video files (F)	20
Number of video chunks per file (K)	15
Number of routers ($\|V\|$)	20
Number of edge routers ($\|N\|$)	12
Video chunk playback duration	2 sec
Bandwidth	5 Mbps
Topology tree height	4
Skewness factor (α)	0.8
Cache capacity percentage (ω)	15%
Cache allocation ratio (ϵ)	1
The probability of continuing watching (p)	0.9

α controls the skewness of popularity distribution. A large α indicates that only few video files are frequently requested and a small α represents large number of video files have similar chance to be requested.

3) The probability of continuing to watch the next chunk (p) is set manually, where we discuss the impact of this parameter on the performance in Section 7.3.3.
4) The stationary distribution of bit rate selection (π) on each edge router, is generated randomly. Since video chunks encoded with different bit rates could incur different loads on the link, this distribution is determined before we choose the video request rate.

The default parameters used in the simulations are listed in Table 2. The cache capacity percentage (ω) indicates the total amount of cache for video streaming, which is calculated by $\omega FK \sum_{i=1}^{B} S(i)$. The cache capacity for each router C_j is influenced by the Cache allocation ratio (ϵ), where $\epsilon = C_i/C_j, i \in N, j \in I$. All edge or intermediate routers are allocated with the same cache capacity.

7.3 Performance Evaluation

We study the effect of cache capacity, cache allocation patterns and content popularity on video access time per bit and cache hits. Simulation results are presented at a 0.9 confidence level.

7.3.1 The Impact of Cache Capacity Percentage

This experiment evaluates cache utilization and efficiency of caching schemes under uniform (equal allocation) cache storage settings. Fig. 5 presents the average access delay per bit for different total cache capacity (budgets). *CE2* placement scheme yields relatively longer access delays across all test cases. The reason is that *CE2* with *LRU* replacement keeps the most recent *Data* but cannot distinguish among video traffic which are mixed with popular and unpopular content. The performance of *ProbeCache* outperforms *CE2* because *ProbCache* caters to frequently requested content which significantly reduces delay. For example, at $\omega = 25\%$, *ProbCache* outperforms *CE2* by 11.8%. However, *DaCPlace* still achieves lower access delay per bit by 26.7% over *ProbCache*. This is because *DaCPlace* optimizes video delivery, considering not only the delay but also storage utilization. Even though *DaCPlace* is a popularity-based scheme, it

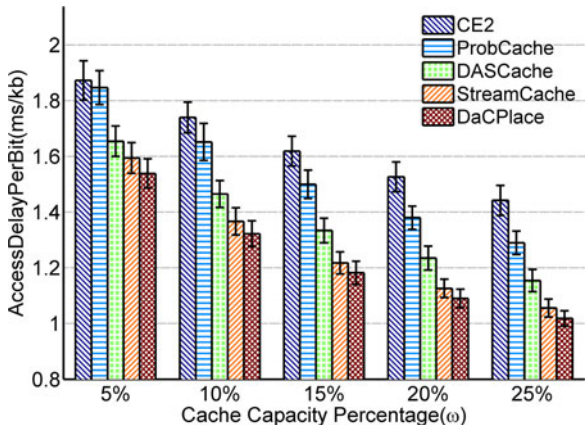


Fig. 5. Caching performance under different total cache capacity settings.

may not keep the most popular content. Such available cache storage is used for multiple less popular content with lower bit rates (smaller size), which thereby achieves higher throughput.

Fig. 5 also shows that *StreamCache* achieves close performance to *DaCPlace* (approximate 3% difference across all cases). This small difference represents our heuristic design considers important features of adaptive video caching system, resulting in a near-optimal performance for different total cache budgets. *DaCPlace* outperforms *DASCache*, especially for larger total cache capacity. For example, at $\omega = 5\%$, the access delay of *DaCPlace* is 7.5% lower than *DASCache* as opposed to 13.3% when $\omega = 25\%$. As mentioned earlier, *DASCache* scheme would not forward *Interest*s to routers which are not on the default routing path. Thus, once a request is missed by a cache, this request loses the opportunity to be satisfied on sibling branches. This loss increases with larger ω which explains the trend.

7.3.2 The Impact of Cache Allocation

We explore the distribution of cache allocation among edge routers and intermediate routers by using different allocation ratios, including homogeneous ($\epsilon = 1$) and heterogeneous ($\epsilon = 0.2, 0.5, 2, 5$) cases.

Cache storage on edge routers significantly impacts facilitating video streaming, since it is closest to users, thus could satisfy requests with minimal delay. Wang et al. [28] claim that when the content popularity distribution is highly skewed (i.e., where only a small portion of videos are frequently requested), edge routers should be allocated larger capacity. In contrast, when content has similar popularity score, more cache capacity should be allocated to intermediate routers to reduce cache redundancy. Figs. 6a and 6b show the access delay under these two scenarios with varying Zipf popularity skewness ($\alpha = 1.2$ and 0.4 respectively).

The performance of all caching schemes has a similar trend across different allocation ratios in both scenarios. The access delay when $\alpha = 1.2$ is lower, compared to the case when $\alpha = 0.4$. This is attributed to cached content being more frequently requested with larger α, which generates more cache hits and results in less average delay. This is further analyzed in Section 7.3.3.

It is straightforward that moving more cache storage to the edge would yield faster system response. However, this

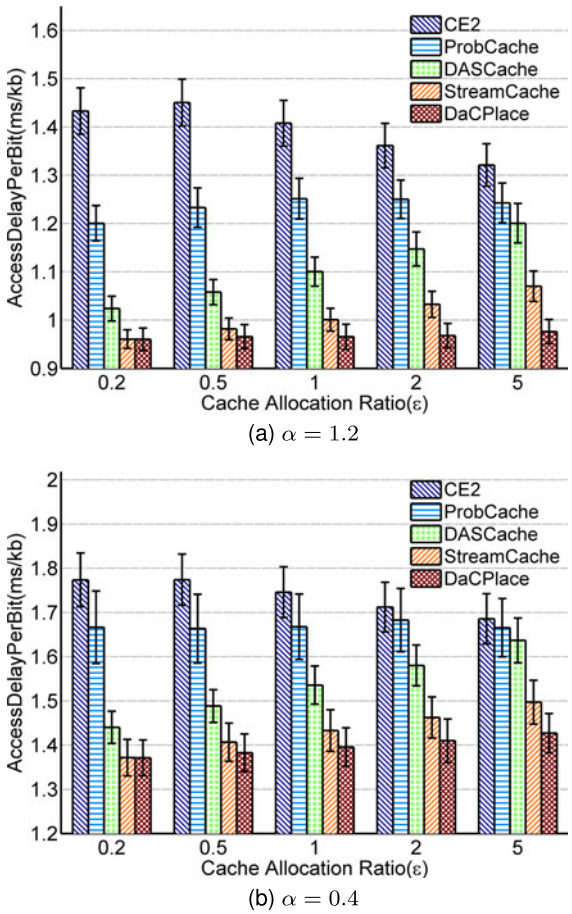


Fig. 6. Video access delay per bit across cache allocation ratios.

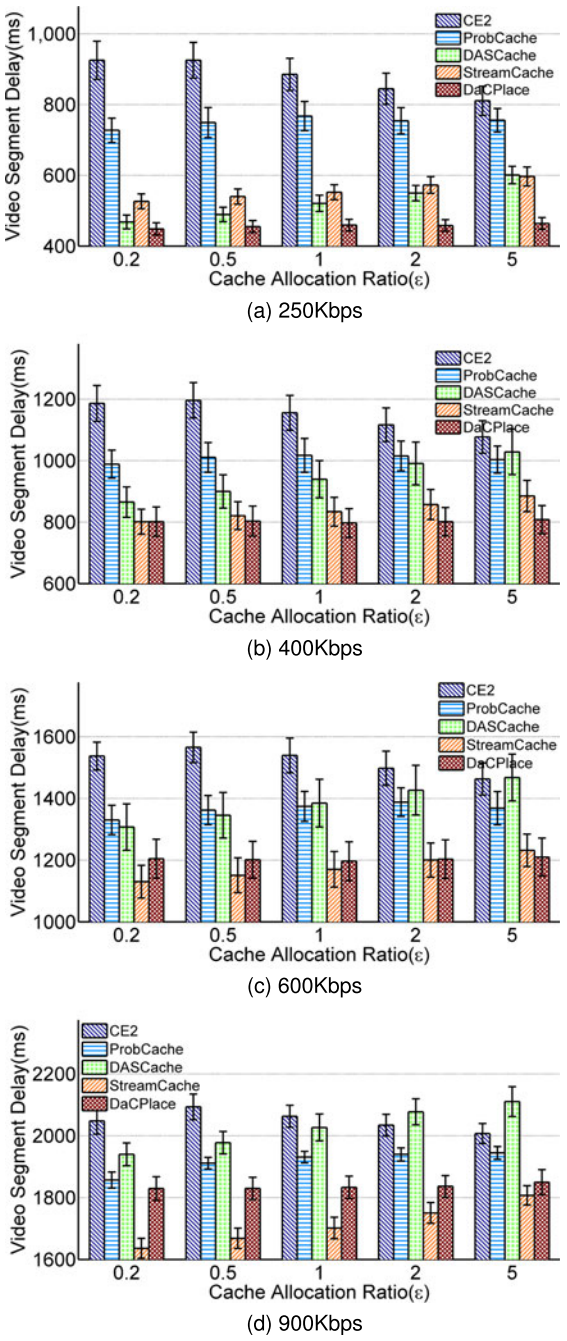


Fig. 7. Average video segment delay ($\alpha = 1.2$).

performance gain is coupled with the expense of less cache hits caused by worse cache utilization. In fact, this constitutes a core challenge in the design of any caching scheme, whereby striking the balance between efficient cache utilization and fast system response. For example, as shown in Fig. 6, with increased ϵ, the performance of *DASCache* degrades and *CE2* improves. These are two typical cases where cache redundancy and system response play as the dominating factor. However, as to *DaCPlace*, the performance difference is statistically insignificant. We then focus on the scenario ($\alpha = 1.2$) to present more features of our scheme. Fig. 7 presents the average delay grouped by bit rates. *DaCPlace* results in fastest download speed for video chunks encoded with 250 Kbps and 400Kbps. For example, when $\epsilon = 1$, *DaCPlace* is 92.8% and 45.0% faster than *CE2*. It is important to note that the access delay of *DaCPlace* is longer than *StreamCache* for higher bit rates (600 Kbps and 900 Kbps). Nevertheless, the overall access delay per bit (shown in Fig. 6a) of *DaCPlace* is still lower than *StreamCache*. This reveals that *DaCPlace* caters to the requests for low bit rates, which is significant for adaptive video streaming: users who request low bit rates should be bound to benefit from caching the most. This is because the link bandwidth of users requesting low bit rates is relatively low, hence cached content would tremendously improve QoS.

As shown in Fig. 6a, the performance of *DASCache* degrades as the size of caching storage on edge routers increases. For example, as ϵ changes from 0.2 to 5, the access delay per bit of *DASCache* correspondingly increases by 17.3%. This impact is further investigated over the amount of cache hits. Fig. 8 shows that cache his of *DASCache* decrease by 48.0% as ϵ changes from 0.2 to 5, but cache hits of *DaCPlace* are relatively unchanged across all tested cache allocation ratios. The superior performance is attributed to *DaCPlace* utilizing *off-path* and *on-path* caching in an optimized and cooperative manner. When $\epsilon = 5$, the amount of cache hits of *DaCPlace* increases by 55.7% over *DASCache*, which demonstrates the significant improvement caused by *off-path* caching.

We also evaluated performance over different cache allocation ratios, under balanced and unbalanced topologies, and found similar performance trends. This result demonstrates that our caching performance is resilient to topology variations.

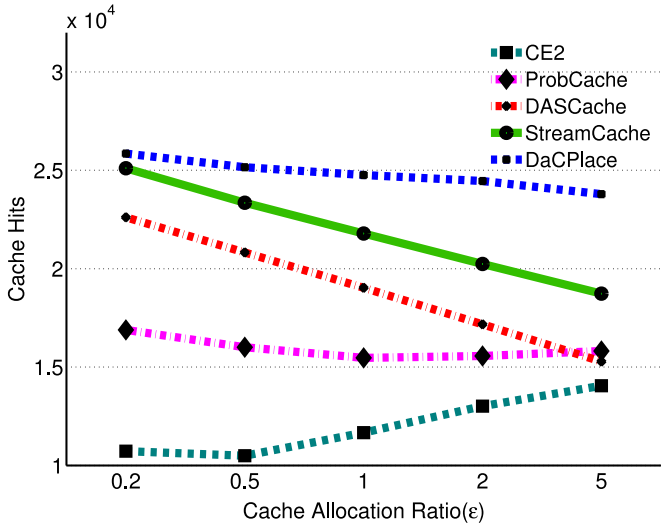


Fig. 8. Total cache hits ($\alpha = 1.2$) across cache allocation ratios.

7.3.3 The Impact of Content Popularity

The popularity of video content is modeled in two levels, as explained in Section 3.2. We control the skewness parameter α in the Zipf distribution, and p in our Geometric distribution model, to vary content popularity on file and chunk levels; respectively.

Fig. 9a shows the results across different skewness parameter values. For example, at $\alpha = 0.8$, *DaCPlace* improves by 37.0% and 26.9% over *CE2* and *ProbCache* respectively. When α changes from 0.4 to 1.2, all tested caching schemes lead to less average access delay. This is because users' requests concentrate on a smaller set of popular content with larger α, which thereby increases the chance of cache hits. *StreamCache* captures popularity variance with different skewness values, and achieves equivalent performance (around 3.2% difference across all cases) compared with the optimal *DaCPlace*.

To capture the impact of the Geometric distribution modeling of video requests, we present a comparative experiment in Fig. 9b that depicts the access delay across different geometric parameter values. As p increases, more users are likely to finish viewing the entire video, which results in increased delay per bit for all caching schemes. The change in p would inherently influence the popularity per video chunk. Even though a certain video file is popular, the last few chunks could witness infrequent requests. Since the first several chunks of a popular video file are highly likely to be cached, as users continue to watch the video, chunks at the middle and (more towards) the end of that file may have to be retrieved directly from the server.

Therefore, the longer the time a user watches a video, the less likely a cache hit occurs. This explains why as p increases, the average access delay grows as well. As both our optimal *DaCPlace* and heuristic *StreamCache* schemes have considered such request patterns, they outperform other caching schemes in capturing this variance over the extended duration of the video. Moreover, it is worthwhile to note that it is possible for users to experience bit rate switches while watching the video in the middle because the throughput of video chunks witnessing cache hit or miss are quite different. Our *DaCPlace* placement scheme is not designed for a particular user but the rate adaptation algorithm should be further considered for smooth playback.

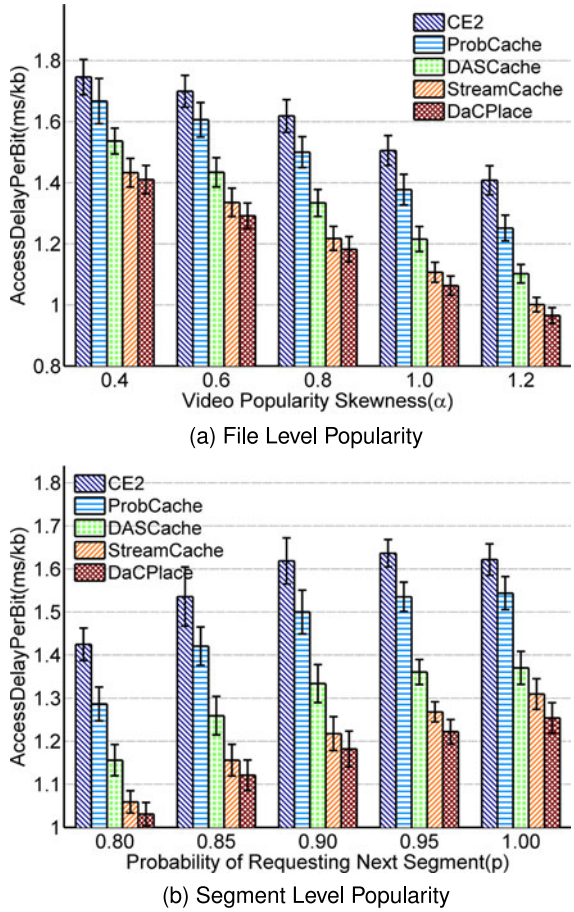


Fig. 9. Video access delay per bit under popularity settings.

7.3.4 The Impact of Transcoding

In addition to cache placement for individual bit rates, another approach which caters to adaptive streaming is transcoding. As explained in Section 2, *DASH-INC* is a typical framework which caches only the highest bit rate of video content to avoid redundancy. Although it is questionable that a single router in ICN has the same processing capability as a backend server or Media Cloud (as required in [24]) to handle simultaneous video demand, we assume an infinite processing speed at ICN routers with no transcoding delay, and compare this upper bound performance of transcoding over *DASH-INC*, with our proposed cache placement schemes.

In this scenario, we apply the default parameters as detailed in Table 2, and evaluate transcoding across different cache sizes. We choose *ProbCache* as the caching scheme used in *DASH-INC* since *ProbCache* is a general method, without specifying the cached bit rate, and it outperforms *CE2* in previous simulations. We also present the performance of *ProbCache* and our schemes *StreamCache*, *DaCPlace*, without transcoding for comparison.

Fig. 10 shows the video access delay across different cache capacities. We observe that transcoding can effectively improve the cache utilization and results in less video delay. For example, at $\omega = 15\%$, the upper bound performance of *ProbCache* with transcoding is 19.6% better than

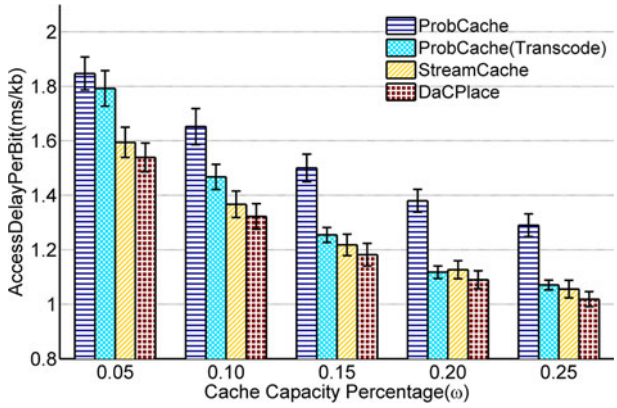


Fig. 10. Video access delay per bit including the test on transcoding.

ProbCache without transcoding. However, as shown in Fig. 10, there is no significant difference between transcoding upper bound and our schemes *StreamCache* and *DaCPlace* at large cache capacity. This result demonstrates that cache placement of selective bit rates, which is optimized for adaptive streaming traffic, can outperform transcoding in the real life with nonzero transcoding delay.

Although transcoding reduces cache redundancy, this method has two disadvantages: First, as only the highest bit rate of video content are candidates of caching, transcoding disregards the streaming traffic of low bit rates, which skews the request patterns for popularity-based schemes (such as *ProbCache*); Second, in some circumstances (such as poor link conditions or mobility), users' requests will typically be at low bit rates. Even though we may ignore the transcoding delay (in this simulation), caching only the highest bit rate for transcoding consumes the cache capacity at a faster pace, hindering the ability of in-network caching to serve more content, compared with the bit-rate-selective caching as we propose.

8 Conclusions and Future Work

We address the premise of dynamic adaptive streaming of video content, with the aim of minimizing the average access time per bit and improving QoS under varying network conditions. The future of video delivery is coupled with adaptive streaming, and schemes that address heterogeneous users cannot ignore tailored video delivery. This factor has played a major role in improving video delivery services under the current Internet and is projected to be even more influential as we steer to ICNs with dominating video traffic.

At the core of this work, we argue for the importance of 1) capturing the characteristics of bit rate selection over varying user demands, which we modeled using a discrete-time Markov chain, 2) establishing a solid queueing & service delay analysis to project link utilization and network variability, which we presented over a multi-class M/G/1 Queueing Model, 3) catering for interest aggregation in video demand, as it significantly reduces network overhead in handling equivalent content requests, which we analyzed and presented as thinning in a Poisson process, 4) presenting *DaCPlace*, as a benchmark solution for variable bit rate caching over ICNs, which incorporates content popularity as a core factor in optimizing caching performance, 5) designing a heuristic scheme, *StreamCache*, which significantly decreases the computational complexity and achieves near-optimal performance to *DaCPlace*, and 6) enabling future improvements on our model and potential benchmarking by developing an NS-3 based ndnSIM simulation environment for *StreamCache* and *DaCPlace*.

We conclude that gauging popularity, on both the chunk and file levels, is critical to optimal cache placement. As adaptive video streaming yields requests for different bit rates, it is crucial to evaluate their respective effects on throughput to better optimize cache utilization. Moreover, utilizing ubiquitous in-network caching improves video delivery delay, under popularity-based schemes, in comparison to edge-based caching. *StreamCache* and *DaCPlace* reduce cache redundancy by capitalizing on off-path caching, further building on ubiquitous in-network caching. Overall, *StreamCache* and *DaCPlace* outperform existing caching schemes under varying cache sizes and content popularities, while managing topology variations.

In future work, novel models are needed to capture user request patterns, ones which are specific to the growing 'Prosumer' body of users, not on the current host-centric Internet architecture that impacts patterns of requests/popularity/variability across users. This is especially evident in the lack of solid statistics/traces on ICN-based user behaviour. Moreover, now that *StreamCache* and *DaCPlace* address challenges with caching multiple bit-rates, and contrast performance gains against transcoding frameworks, we need to further investigate the premise of hybrid approaches that could, for example, transcode less popular bit-rates.

Future research on finding more adaptive popularity schemes, to augment the currently dominant Zipf distribution, are much needed. In extending *DaCPlace* and *StreamCache*, our future work will incorporate rate adaptation, whereby we consider the impact of rate adaptation on the user end, and how caching schemes cater to current and future replacement strategies. Also, although throughput is a fundamental metric, we will extend our work to improve QoS under other factors, such as playback freezing, bit rate switch frequency.

Appendix A
Formulation Transformation

Big-M Transformation. The optimization objective (Equation (5)) is expanded using Equations (6), (7), and (8), where we designate $\mathbb{E}[RTT_j(f,k,b)]$ as a continuous variable. Then, we need extra constraints on this $\mathbb{E}[RTT_j(f,k,b)]$ in order to make it equal to the minimal round trip delay among available forwarding paths based on Definition 4.4. However, Definition 4.4 is not linear. The task in this step is then transforming the *min* operator, using the "**big-**$\mathbb{M}$" method (where $\mathbb{M}$ denotes a large positive constant number). We also define an artificial binary variable, $\beta_j^m(f,k,b)$ which indicates whether choosing the forwarding path m would result in minimal delay or not. For simplicity, we denote $\beta_j^m(f,k,b)$ with β_j^m. For each $j \in N$ where $m = 1 \cdots |U_j|$, we need to add $|U_j|$ constraints with different m

$$\mathbb{E}[RTT_j(f,k,b)] \geq \mathbb{E}[RTT_j^m(f,k,b)] - \beta_j^m \mathbb{M}, \tag{22}$$

and one equality constraint

$$\sum_{m=1}^{|U_j|} \beta_j^m = |U_j| - 1, \tag{23}$$

where Constraints (22) and (23) ensure that $\mathbb{E}[RTT_j(f,k,b)]$ would be assigned the round trip time on a forwarding path with the smallest delay.

Forwarding Path Contraints. The expression of Definition 4.3 is now contained in Constraint (22) by replacing $\mathbb{E}[RTT_j^m(f,k,b)]$, where $d_j(m)$ can be specified based on the type of end points using the corresponding linear expression of Definition 4.2 before solving the optimization problem. However, the value of $\mathbb{I}(j,m,n)$ cannot be predetermined based on Definition 4.1 since it relies on the variable of cache placement indicator x. Thus, in this step, we transform Definition 4.1 into a linear expression by designating $\mathbb{I}(j,m,n)$ as a binary variable and adding the following *Forwarding Path Contraints* for each $\mathbb{I}(j,m,n)$

$$\mathbb{I}(j,m,n) \geq \mathbb{I}(j,m,n-1), \tag{24}$$

$$\mathbb{I}(j,m,n) \geq x_{L_{j,m}(n)}(f,k,b), \tag{25}$$

$$\mathbb{I}(j,m,n) \leq \mathbb{I}(j,m,n-1) + x_{L_{j,m}(n)}(f,k,b), \tag{26}$$

where (24) and (25) give the lower bound of $\mathbb{I}(j,m,n)$, which denote that the binary indicator $\mathbb{I}(j,m,n)$ should be greater or equal to the value of its previous hop indicator, $\mathbb{I}(j,m,n-1)$, and the cache decision x of current router, $x_{L_{j,m}(n)}(f,k,b)$. Constraint (26) gives the upper bound. If both $\mathbb{I}(j,m,n-1)$ and $x_{L_{j,m}(n)}(f,k,b)$ are **0**, $\mathbb{I}(j,m,n)$ must be **0**. Thus, Constraints (24), (25) and (26) together are equivalent to Definition 4.1 on $\mathbb{I}(j,m,n)$.

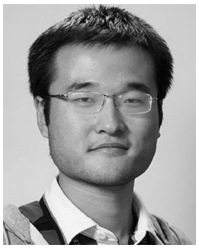
ACKNOWLEDGMENTS

This research is supported by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Ontario Ministry of Economic Development and Innovation under the Ontario Research Fund-Research Excellence (ORF-RE) program.

REFERENCES

[1] Cisco, "Cisco visual networking index: Forecast and methodology, 2015–2020," 2015.
[2] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, Apr. 2011.
[3] G. Xylomenos, et al., "A survey of information-centric networking research," *IEEE Commun. Surveys Tut.*, vol. 16, no. 2, pp. 1024–1049, Apr.–Jun. 2014.
[4] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, "Adaptive multimedia streaming in information-centric networks," *IEEE Netw.*, vol. 28, no. 6, pp. 91–96, Nov./Dec. 2014.
[5] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Commun. ACM*, vol. 49, no. 1, pp. 101–106, 2006.
[6] A. Passarella, "A survey on content-centric technologies for the current internet: CDN and P2P solutions," *Comput. Commun.*, vol. 35, no. 1, pp. 1–32, 2012.
[7] Y. Liu, et al., "Dynamic adaptive streaming over CCN: A caching and overhead analysis," in *Proc. IEEE Int. Conf. Commun.*, 2013, pp. 3629–3633.
[8] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," *IEEE Commun. Surveys Tut.*, vol. 17, no. 3, pp. 1473–1499, Jul.–Sep. 2015.
[9] Y. Wang, M. Xu, and Z. Feng, "Hop-based probabilistic caching for information-centric networks," in *Proc. IEEE Global Commun. Conf.*, 2013, pp. 2102–2107.
[10] Y. Xu, Y. Li, T. Lin, G. Zhang, Z. Wang, and S. Ci, "A dominating-set-based collaborative caching with request routing in content centric networking," in *Proc. IEEE Int. Conf. Commun.*, 2013, pp. 3624–3628.
[11] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking," in *Proc. ACM SIGCOMM Workshop Inf.-Centric Netw.*, 2013, pp. 27–32.
[12] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. ACM SIGCOMM Workshop Inf.-Centric Netw.*, 2012, pp. 55–60.
[13] T. Koponen, et al., "A data-oriented (and beyond) network architecture," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2007, pp. 181–192.
[14] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing information networking further: From PSIRP to PURSUIT," in *Proc. Int. Conf. Broadband Commun. Netw. Syst.*, 2010, pp. 1–13.
[15] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. ACM Int. Conf. Emerging Netw. Experiments Technol.*, 2009, pp. 1–12.
[16] J. Li, et al., "Popularity-driven coordinated caching in named data networking," in *Proc. ACM/IEEE Symp. Architectures Netw. Commun. Syst.*, 2012, pp. 15–26.
[17] S. K. Fayazbakhsh, et al., "Less pain, most of the gain: Incrementally deployable ICN," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2013, pp. 147–158.
[18] A. Dabirmoghaddam, M. M. Barijough, and J. Garcia-Luna-Aceves , "Understanding optimal caching and opportunistic caching at the edge of information-centric networks," in *Proc. 1st ACM Int. Conf. Inf.-Centric Netw.*, 2014, pp. 47–56 .
[19] D. Lee, C. Dovrolis, and A. Begen, "Caching in http adaptive streaming: Friend or foe?" in *Proc. ACM Netw. Operating Syst. Support Digital Audio Video Workshop*, 2014, pp. 31–36.
[20] R. Jia, Z. Liu, X. Wang, X. Gan, X. Wang, and J. J. Xu, "Modeling dynamic adaptive streaming over information-centric networking," *IEEE Access*, vol. 4, pp. 8362–8374, 2016.
[21] J. Lee, K. Lim, and C. Yoo, "Cache replacement strategies for scalable video streaming in CCN," in *Proc. IEEE Asia-Pacific Conf. Commun.*, 2013, pp. 184–189.
[22] C. Kreuzberger, B. Rainer, and H. Hellwagner, "Modelling the impact of caching and popularity on concurrent adaptive multimedia streams in information-centric networks," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, 2015, pp. 1–6.
[23] R. Grandl, K. Su, and C. Westphal, "On the interaction of adaptive video streaming with content-centric networking," in *Proc. IEEE Packet Video Workshop*, 2013, pp. 1–8.
[24] Y. Jin, Y. Wen, and C. Westphal, "Optimal transcoding and caching for adaptive streaming in media cloud: An analytical approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1914–1925, Dec. 2015.
[25] W. Li, S. Oteafy, and H. S. Hassanein, "Dynamic adaptive streaming over popularity-driven caching in information-centric networks," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 5747–5752.
[26] L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang, "OSPFN: An OSPF based routing protocol for named data networking," Univ. Memphis and Univ. Arizona, Tech. Rep. NDN-2012–13, 2012.
[27] G. Rossini and D. Rossi, "Evaluating ccn multi-path interest forwarding strategies," *Comput. Commun.*, vol. 36, no. 7, pp. 771–778, 2013.
[28] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Design and evaluation of the optimal cache allocation for content-centric networking," *IEEE Trans. Comput.*, vol. 65, no. 1, pp. 95–107, Jan. 2016.
[29] D. M. Lucantoni, "New results on the single server queue with a batch markovian arrival process," *Commun. Statistics Stochastic Models*, vol. 7, no. 1, pp. 1–46, 1991.
[30] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Modeling data transfer in content-centric networking," in *Proc. 23rd Int. Teletraffic Congress*, 2011, pp. 111–118.
[31] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data Networks*, vol. 2. Englewood Cliffs, NJ, USA: Prentice-Hall, 1987.
[32] W. Li, S. M. Oteafy, and H. S. Hassanein, "StreamCache: Popularity-based caching for adaptive streaming over information-centric networks," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–6.
[33] A. Afanasyev, et al., "ndnSIM: NDN simulator for ns-3," Univ. California, Los Angeles, Tech. Rep. NDN-0005, 2012.
[34] Gurobi, "Gurobi optimizer reference manual." [Online]. Avialable: http://www.gurobi.com/documentation/.
[35] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over http dataset," in *Proc. 3rd Multimedia Syst. Conf.*, 2012, pp. 89–94.

[36] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks," in *Proc. Int. Conf. Res. Netw.*, 2009.
[37] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proc. 18th Annu. Joint Conf. IEEE Comput. Commun. Societies*, vol. 1, 1999, pp. 126–134.



Wenjie Li (S'14) received the BEng degree in computer science and engineering from Southeast University, Nanjing, Jiangsu, China, in 2013 and the MSc degree in computer science from Queen's University, in 2015. He is working toward the PhD degree in the School of Computing, Queen's University, Canada, and a research assistant in the Queen's Telecommunications Research Lab. His research interests include routing and ubiquitous caching in information-centric networks and efficient content delivery over ICN for dynamic adaptive streaming applications. He is a member of the IEEE.

Sharief M.A. Oteafy (S'08-M'13) received the PhD degree in 2013, focusing on adaptive resource management in Next Generation Sensing Networks, introducing Organic WSNs that adapt to their environment and scale with resource augmentation. He is an adjunct assistant professor in the School of Computing, Queen's University. His current research focuses on dynamic architectures for enabling large scale synergy with the Internet of Things and managing the proliferation of big sensed data. He is the IEEE AHSN Standards Liaison and on the ComSoc Tactile Internet standards WG, and has co-authored a book on *Dynamic Wireless Sensor Networks*, and more than 40 peer-refereed publications in sensing systems and IoT. He co-chaired a number of IEEE workshops, in conjunction with IEEE ICC, and IEEE LCN conferences, and served on the TPC of numerous IEEE and ACM symposia. He is a member of the IEEE.

Hossam S. Hassanein (S'86-M'90-SM'06-F'17) received the PhD degree in computing science from the University of Alberta, Canada, in 1990. He is a leading authority in the areas of broadband, wireless and mobile networks architecture, protocols, control, and performance evaluation. His record spans more than 500 publications in journals, conferences, and book chapters, in addition to numerous keynotes and plenary talks at flagship venues. He is also the founder and director of the Telecommunications Research Lab, School of Computing, Queen's University, Kingston, ON, Canada, with extensive international academic and industrial collaborations. He is an IEEE Communications Society Distinguished Speaker and a past chair of the IEEE Communication Society Technical Committee on AHSN. He is a fellow of the IEEE and has received several recognitions and best papers awards.