Optimal Load-Splitting and Distributed-Caching for Dynamic Content

Bahman Abolhassani[1], John Tadrous[2], Atilla Eryilmaz[1]
[1,2,3] Department of Electrical and Computer Engineering
[1] The Ohio State University, Columbus, 43210
[1] Email: abolhassani.2@osu.edu, eryilmaz.2@osu.edu
[2] Gonzaga University, Spokane, WA 99202
[2] Email: tadrous@gonzaga.edu

***Abstract*—In this work, we consider the problem of 'fresh' caching at distributed (front-end) local caches of content that is subject to 'dynamic' updates at the (back-end) database. We first provide new models and analyses of the average operational cost of a network of distributed edge-caches that utilizes wireless multicast to refresh aging content. We attack the problems of what to cache in each edge-cache and how to split the incoming demand amongst them (also called "load-splitting" in the rest of the paper) in order to minimize the operational cost. While the general form of the problem comes with an NP-hard Knapsack structure, we were able to completely solve the problem by judiciously choosing the number of edge-caches to be deployed over the network. Interestingly, our findings reveal that the optimal caching policy necessitates unequal load-splitting over the edge-caches even when all conditions are symmetric. Moreover, we find that edge-caches with higher load will generally cache fewer but relatively more popular content. We further investigate the tradeoffs between cost reduction and cache savings when employing equal and optimal load-splitting solutions for demand with Zipf(z) popularity distribution. Our analysis reveals that equal load-splitting to edge-caches achieves close-to-optimal for less predictable demand ($z < 2$) while also saving in the cache size. On the other hand, for more predictable demand ($z > 2$), optimal load-splitting results in substantial cost gains while decreasing the cache occupancy.**

***Index Terms*—Content Distribution Networks, Caching, Age of Information, Dynamic Content**

I. Introduction

With the emergence of new services and application scenarios, such as Youtube, augmented reality, social networking, and online gaming, which produce dynamically changing data over time, serving the most recent version of data to end-users is becoming the main challenge due to the massive device connectivity. To alleviate the latency of data transmission between the servers and end-users, many applications utilize edge-caches close to the end-users to deliver dynamic contents, reducing the network latency and system congestion during the peak traffic time [1], [2]. Usually, several edge-caches are deployed over the edge networks and the data required by end-users can be cached at one or multiple edge-caches. By caching a large number of dynamic contents in the edge-caches, the average response time can be reduced, benefiting from higher cache hit rates. However higher hit rates come at the expense of less fresh content, resulting in higher overall system cost.

One possible solution for tackling this problem is to cache popular contents at the edge-caches to reduce the total response time to data requests. Content Distribution Networks (CDNs) utilize a large mesh of edge-caches to deliver content from locations closer to the end users [3], [4]. Existing caching strategies rely on the assumption of static (or quasi-static) nature of the stored content and aim to simply maximize the cache hit rate [5]. In many real-world scenarios, such as news updates in social networks and system state updates in cyber-physical networks, the data content is subject to updates at various rates, which render the older versions of the content less useful [6], [7]. Hence, there is a growing need to develop new caching strategies that account for the refresh characteristics and ageing costs of content for efficient dynamic content distribution.

Numerous works study the dynamic content delivery in caching systems and effective strategies have been proposed [8] and [9]. In [10], authors propose two metrics to measure the cached content freshness: age of synchronization (AoS) and age of information (AoI). Most existing research regarding the freshness of the local cache focus on the AoI metric and often the objective is to minimize the average AoI. Kam et al. [9] propose a dynamic model in which the rate of requests depends on the popularity and the freshness of information to minimize the number of missed requests.

While AoI is a meaningful metric for measuring the freshness of content in some systems, there are many real-world scenarios where a content does not lose its value simply because time has passed since it was put into the cache. These types of dynamic contents include news and social network updates where the users prefer to have the most fresh version but so long as there is no new update, that content is considered to be the most fresh version. In this work, we use a new freshness metric called *Age-of-Version* (AoV) which counts the integer difference between the versions at the database and the local cache. We also introduce a new cost function for dynamic content caching which captures both the cost due to the miss event and the cost due to content freshness [11] which grows with the AoV metric. Moreover, our model utilizes the multicasting

This work funded, in part, by the NSF grants: CNS-NeTS-2007231, CNS-SpecEES-1824337, CNS-NeTS-1717045, CNS-NeTS-2106679; and the ONR Grant N00014-19-1-2621.

property of the wireless medium to opportunistically update the cached contents over the edge-caches. Finally, our model extends the traditional caching paradigm to allow for varying *generation dynamics* of content, and calls for new designs that incorporate these dynamics into its decisions.

In particular, we focus on wireless networks that utilize edge-caches to serve dynamic contents to a group of end-users and edge-caches can update their caches content with no additional cost by overhearing that content being served to other edge-caches. we propose a freshness-driven caching model for dynamic content, which accounts for the update rate of data content and provide an analysis of the average operational cost.

This work is related to our earlier work [6], which also considered optimal distributed caching over the wireless edge. However, the setting in [6] is complementary to this one, with each local cache having its separate demand to serve without a possibility of splitting the load. Here, by allowing such a split, the setting as well as the nature of the problem and its solution are completely different. Not only do they lead to new challenges, such as a Knapsack problem appearing within it, but it also results in new insights on how to serve a common edge user population with distributed edge-caches.

By intelligently choosing the number of edge-caches, we propose a policy that jointly optimizes the distributed edge caching and load-splitting between those edge-caches. The proposed optimal policy reveals counter-intuitive insights on the nature of the distributed edge caching for dynamic content. In particular, for the practical case of Zipf popularity, load and cache capacity are generally split unequally between the edge-caches, and edge-caches with higher load will store less items in their cache, however, they are the more popular ones. We aim to reveal the trade-off between our proposed optimal policy and the more practically implemented policy where the load is split equally between the edge-caches. Our contributions, along with the organization of the paper, are as follows.

- In Section II, we present a tractable caching model that utilizes distributed edge-caches for serving dynamic content over wireless broadcast channels in which edge-caches take advantage of the wireless multicasting to keep their cached content fresh.
- In Section III, we provide a full characterization of the optimal caching policy which jointly optimizes the number of edge-caches, load-splitting, and cache placements over the network. The solution is achieved by intelligently manipulating a group of intractable 0-1 Knapsack problems to remove all the inequality constraints that renders such problems NP-hard. The outcome is a policy that achieves the global minimum average cost. Our findings reveals the nature of the unequal load-splitting between the edge-caches and dependence of the content caching on the load allocated to each edge-cache.
- In Section IV, we provide the optimal content placement for the special case of the equal load-splitting. We also characterize the cost-cache trade-off between the optimal policy and the equal load-splitting policy. Our findings reveal that as the number of edge-caches increases, the equal load-splitting cost decreases at the expense of increasing the cache occupancy.
- In Section V, comparing the average cost and the cache occupancy of the proposed optimal policy to the equal load-splitting policy, we investigate trade-off using numerical simulations for the practical case of Zipf popularity and highlight scenarios in which each of these approaches are more cost or cache effective. Our findings reveal that for less predictable demand, i.e., more uncertainty about the demand, equal load-splitting can potentially have significant cache savings while achieving a close-to-optimal cost. On the other hand, as the certainty about the demand increases, the optimal policy can achieve significant gains on the cost without increasing the cache occupancy. Finally, we conclude the work in Section VI.

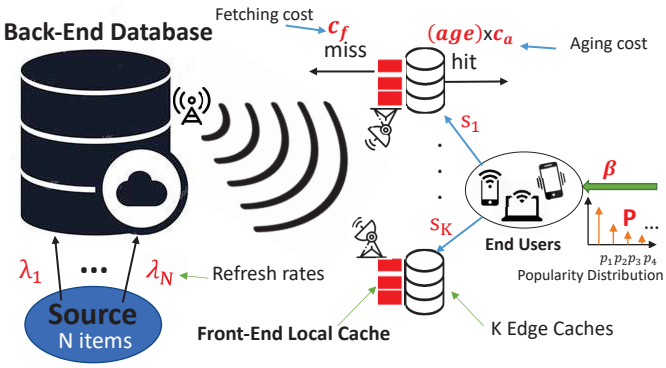

Fig. 1: Setting of *Fresh Caching for Dynamic Content*

II. System Model

We consider the generic hierarchical setting depicted in Fig. 1, whereby: the (limited) local cache serves a user population that generates requests to content according to a popularity distribution; while the back-end database receives updates to refresh the content with different rates. In the following, we will provide the details of this generic model, followed by the goal of our work.

Demand Dynamics: We assume that a set $\mathcal{N}$ of N unit-sized data items (with dynamically changing content) is being served to the user population through a hierarchical caching system as depicted in Fig. 1. In particular, there are K edge-caches that supply local content to the neighboring users. Requests arrive to the local edge-cache k according to a Poisson process[1] with rate $\beta_k \geq 0$, which captures the request intensity of the user population served by the edge-cache k. An incoming request targets data item $n \in \mathcal{N}$ with probability p_n. Accordingly, the probability distribution $\mathbf{p} = (p_n)_{n=1}^{N}$ captures the popularity profile of the data items. Furthermore, denoting the total request arrival rate by β, we define $s_k = \frac{\beta_k}{\beta}, \forall k \in 1, ..., K$ to be the fraction of the total request served by the edge-cache k. Accordingly, the vector $\mathbf{s} = (s_k)_{k=1}^{K}$ captures the load-splitting between the edge-caches.

[1]Accordingly, we assume that the system evolves in continuous time.

Generation Dynamics: At the database, each data item may receive updates at random times to replace its previous content. We assume that data item n receives updates according to a Poisson process with rate $\lambda_n \geq 0$. Note that $\lambda_n = 0$ encapsulates the traditional case of *static* content that never receives an update. We denote the vector $\boldsymbol{\lambda} = (\lambda_n)_{n=1}^N$ as the collection of update rates for the database.

Age Dynamics: Since the data items are subject to updates at the database, the same items in the local caches may be *older versions* of the content. To measure the freshness of local content, we define the *age* $\Delta_n^k(t) \in \{0, 1, \ldots\}$ at time t for item n stored at the edge-cache k as the number of updates that the locally available item n has received in the database since it has been most recently cached. We name this freshness metric as the *Age-of-Version* (AoV), since it counts the integer difference between the versions at the database and the local cache. The incoming request to an item that is stored in the edge-cache k is served from the local cache, but potentially with a positive AoV value $\Delta_n^k(t)$.

Fetching and Ageing Costs: Now that we have the dynamics defined, we can introduce the key operational and performance costs associated with our caching system. On the operational side, we denote the cost of fetching an item from the database to the local cache by $c_f > 0$. On the performance side, we assume that serving an item n from the edge-cache k with age $\Delta_n^k(t)$ incurs a *freshness/age* cost of $c_a \times \Delta_n^k(t)$ for some $c_a \geq 0$, which grows linearly[2] with the AoV metric. This ageing cost measures the growing discontent of the user for receiving an older version of the content she/he demands.

Content Multicasting: We stress that broadcast nature of the wireless medium enables transmission of content made to one edge-cache to be received and used to update content in other edge-caches at no additional cost. This *multicasting property* non-trivially couples the decisions across the distributed cache space for optimal caching solution. Moreover, all replicas of the same content at local caches will have the same age of version thanks to the broadcast nature of wireless communication, i.e., $\Delta_n^k(t) = \Delta_n(t)$ among the edge-caches that hold item n.

Our broad objective in this work is to develop efficient distributed edge caching strategies for the above setting that optimally balance the tradeoff between the cost of serving the fresh item from database and the cost of providing potentially older content to the users from the local cache.

A. Problem Formulation

Let $\mathcal{I}_n \subseteq \mathcal{K}, \forall n \in \mathcal{N}$ be the set of edge-caches that have stored item n and $|\mathcal{K}| = K$ is the total number of edge-caches deployed over the network. Note that due to high refresh rates, edge-caches may not necessarily fill their cache to avoid excessive freshness costs. As such, $\sum_{n=1}^N |\mathcal{I}_n|$ will be always finite for the dynamic content even if there is unlimited cache storage capacity. The total arrival request rate β for items is split between the edge-caches, such that each edge-cache k receives a fraction s_k of the total incoming request. Therefore, $\mathbf{s} = (s_k)_{k=1}^K$ is the vector of load-splitting between the edge-caches where $\sum_{k=1}^K s_k = 1$.

Lemma 1: Let $C^{\mathcal{D}}(\{\mathcal{I}_n\}_n, \mathbf{s})$ be the average caching cost of a system composed of K edge-caches where each item $n \in \mathcal{N}$ is stored in the set $\mathcal{I}_n \subseteq \mathcal{K}$ of edge-caches and each edge-cache $k \in \mathcal{K}$ receives the fraction s_k of the total request. Then:

$$C^{\mathcal{D}}(\{\mathcal{I}_n\}_n, \mathbf{s}) = \beta c_f + \sum_{n=1}^N \left(\left(\sum_{k \in \mathcal{I}_n} s_k \right) \left(\frac{c_a \lambda_n}{1 - \sum_{k \in \mathcal{I}_n} s_k} - \beta p_n c_f \right) \right) \tag{1}$$

Proof. Here we only give the outline of the proof. For the full proof refer to [12]. Let $\{\Pi_{\mathcal{I}_n}^n(t), t \geq 0\}$, $\forall n \in \mathcal{N}$ be the Markov process describing the freshness age of cached item n at time t under the cached set $\mathcal{I}_n$. The evolution of this process is shown in Fig. 2.

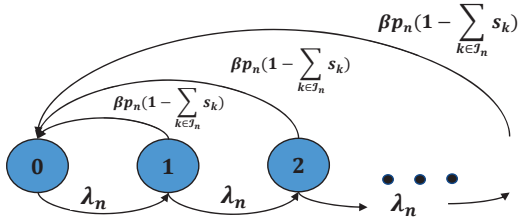

Fig. 2: Markov chain diagram for freshness $\{\Pi_{\mathcal{I}_n}^n(t), t \geq 0\}$ under the cached set $\mathcal{I}_n$.

Since $\Pi_{\mathcal{I}_n}^n(t) \xrightarrow[t \to \infty]{d} \bar{\Pi}_{\mathcal{I}_n}^n$, the average age of item n is given by:

$$\mathbb{E}[\bar{\Pi}_{\mathcal{I}_n}^n] = \frac{\lambda_n}{\beta p_n (1 - \sum_{k \in \mathcal{I}_n} s_k)}. \tag{2}$$

The average system cost in the distributed edge caching where each item $n \in \mathcal{N}$ is stored in the set $\mathcal{I}_n \subseteq \mathcal{K}$ of edge-caches and the load is split between the K edge-caches according to the vector $\mathbf{s} = (s_1, ..., s_K)$, comprises two main terms and is given by:

$$\begin{aligned} C^{\mathcal{D}}(\{\mathcal{I}_n\}_n, \mathbf{s}) = \beta c_f \sum_{n=1}^N p_n (1 - \sum_{k \in \mathcal{I}_n} s_k) \\ + \beta c_a \sum_{n=1}^N p_n (\sum_{k \in \mathcal{I}_n} s_k) \mathbb{E}[\bar{\Pi}_{\mathcal{I}_n}^n]. \end{aligned} \tag{3}$$

The first term shows the average *fetching cost* due to the miss events. The second term shows the average *freshness cost* due to the hit events incurred by serving potentially aged content from the local cache. Substituting Equation (2) in Equation (3) gives the average cost of the system. ■

The cost minimization problem for such system would thus be:

$$\begin{aligned} \min_{(s_k)_k, \{\mathcal{I}_n\}_n, K} \; & C^{\mathcal{D}}(\{\mathcal{I}_n\}_n, \mathbf{s}), \\ s.t. \quad & 0 \leq s_k \leq 1, \sum_k s_k = 1, \\ & \mathcal{I}_n \subseteq \mathcal{K}, \\ & K \geq 0, \end{aligned} \tag{4}$$

[2]While this linearity assumption is meaningful as a first-order approximation to ageing cost and facilitates simpler expressions in the analysis, it can also be generalized to convex forms to extend this basic framework.

Minimizing the average caching cost requires finding the optimal value for the number of edge-caches, how to split the load between those edge-caches and which items should be stored at each edge-cache.

In the following sections, we use the caching cost defined in Equation (1) and propose an optimal caching strategy that jointly optimizes distributed edge caching and load-splitting.

III. Jointly Optimal Distributed Caching and load-splitting of Dynamic Content

In this section we tackle the general problem formulated in (4). The characterization of the optimal caching strategy under this setting will not only yield interesting insights about the impact of generation dynamics, but we will also provide an upper bound on the cache occupancy of the proposed optimal caching strategy.

First, in order to gain an insight into the optimal caching policy, we tackle the problem in a simplified version by assuming that the number of edge-caches K and the vector of load-splitting $\mathbf{s} = (s_1, ..., s_K)$ are given and $\mathbf{s}$ is not necessarily uniform, i.e., unequal load-splitting between the edge-caches. Our objective is thus for the given load-splitting vector $\mathbf{s}$ to choose the cached sets $\mathcal{I}_n(\mathbf{s}) \subseteq \mathcal{K}, \forall n \in \mathcal{N}$ to be stored at the K edge-caches in order to minimize the average cost of the system.

$$\min_{\{\mathcal{I}_n\}_n \in \mathcal{K}^N} C^{\mathcal{D}}(\{\mathcal{I}_n\}_n, \mathbf{s}). \tag{5}$$

Proposition 1: The policy $\{\mathcal{I}_n^*(\mathbf{s})\}_n \in \mathcal{K}^N$ that solves (5) is given by:

$$\mathcal{I}_n^*(\mathbf{s}) = \begin{cases} \mathcal{I}_n'(\mathbf{s}), & s_{k''} \geq (1 - \sum_{k \in \mathcal{I}_n'(\mathbf{s})} s_k) \\ & - \frac{c_a \lambda_n}{\beta p_n c_f} \frac{1}{1 - \sum_{k \in \mathcal{I}_n'(\mathbf{s})} s_k}, \\ \mathcal{I}_n'(\mathbf{s}) \cup \{k''\}, & oth, \end{cases}$$

where $\mathcal{I}_n'(\mathbf{s})$ has the form of $0-1$ knapsack problem given by:

$$\begin{aligned} \mathcal{I}_n'(\mathbf{s}) &= \arg\max_{\mathcal{I}_n \in \mathcal{K}} \sum_{k \in \mathcal{I}_n} s_k \\ s.t. \quad & \sum_{k \in \mathcal{I}_n} s_k \leq \max\left(0, 1 - \sqrt{\frac{c_a \lambda_n}{\beta c_f p_n}}\right) \end{aligned} \tag{6}$$

and $k'' = \arg\min_{k \in \mathcal{K} \setminus \mathcal{I}_n'(\mathbf{s})} s_k$.

Proof. To prove this, we define $\delta_n^{\mathcal{D}}(\mathcal{I}_n, \{k'\})$ to be the marginal cost of adding item n already stored in the set $\mathcal{I}_n \subset \mathcal{K}$ of edge-caches to the new edge-cache $k' \notin \mathcal{I}_n$ that does not have item n in its cache. In other words:

$$\delta_n^{\mathcal{D}}(\mathcal{I}_n, \{k'\}) := C^{\mathcal{D}}(\{\mathcal{I}_n'\}_n)|_{\mathcal{I}_n' = \mathcal{I}_n \cup \{k'\}} - C^{\mathcal{D}}(\{\mathcal{I}_n'\}_n)|_{\mathcal{I}_n' = \mathcal{I}_n}$$

Using the average caching cost in Equation (1), we have:

$$\delta_n^{\mathcal{D}}(\mathcal{I}_n, \{k'\}) = s_{k'} \left(\frac{c_a \lambda_n}{\left(1 - \sum_{k \in \mathcal{I}_n} s_k\right)\left(1 - \sum_{k \in \mathcal{I}_n} s_k - s_{k'}\right)} - \beta p_n c_f \right).$$

In the case of $\delta_n^{\mathcal{D}}(\mathcal{I}_n, \{k'\}) < 0$ for a given cached set $\mathcal{I}_n \subset \mathcal{K}$, adding item n to the edge-cache k' will reduce the average caching cost. On the other hand, items with positive $\delta_n^{\mathcal{D}}(\mathcal{I}_n, \{k'\})$ can only increase the cost if added to the edge-cache k'. Therefore, the sufficient condition for the optimality of set $\mathcal{I}_n(\mathbf{s})$ of the edge-caches to store item n is given by:

$$\delta_n^{\mathcal{D}}(\mathcal{I}_n, \{k'\}) > 0, \quad \forall k' \subset \mathcal{K} \setminus \mathcal{I}_n \tag{7}$$

Using the definition of $\delta_n^{\mathcal{D}}(\mathcal{I}_n, \{k'\})$, for this to not hold we should have:

$$\frac{c_a \lambda_n}{\left(1 - \sum_{k \in \mathcal{I}_n} s_k\right)\left(1 - \sum_{k \in \mathcal{I}_n} s_k - s_{k'}\right)} - \beta p_n c_f \leq 0,$$

for some $k' \subset \mathcal{K} \setminus \mathcal{I}_n$. Since $s_k \geq 0, \forall k \in \mathcal{K}$, we can rewrite this as:

$$\frac{c_a \lambda_n}{\beta p_n c_f} \leq (1 - \sum_{k \in \mathcal{I}_n} s_k)(1 - \sum_{k \in \mathcal{I}_n} s_k - s_{k'}) \leq (1 - \sum_{k \in \mathcal{I}_n} s_k)^2,$$

which gives the condition as:

$$\sum_{k \in \mathcal{I}_n} s_k \leq \max\left(0, 1 - \sqrt{\frac{c_a \lambda_n}{\beta c_f p_n}}\right).$$

We define the set $\mathcal{I}_n'(\mathbf{s})$ as in Equation (6) such that it maximizes $\sum_{k \in \mathcal{I}_n} s_k$ while also satisfying the above condition. For $\mathcal{I}_n'(\mathbf{s})$ to be optimal, the sufficient condition for optimality given in Equation (7) should hold. In other words:

$$s_{k'} \geq (1 - \sum_{k \in \mathcal{I}_n'(\mathbf{s})} s_k) - \frac{c_a \lambda_n}{\beta p_n c_f} \frac{1}{1 - \sum_{k \in \mathcal{I}_n'(\mathbf{s})} s_k}, \forall k' \subset \mathcal{K} \setminus \mathcal{I}_n' \tag{8}$$

Defining $k'' = \arg\min_{k \in \mathcal{K} \setminus \mathcal{I}_n'(\mathbf{s})} s_k$, if Equation (8) holds for k'', then it will hold for $\forall k' \subset \mathcal{K} \setminus \mathcal{I}_n'$ and the set $\mathcal{I}_n^*(\mathbf{s}) = \mathcal{I}_n'(\mathbf{s})$ satisfies the sufficient condition for optimality and therefore is the optimal set of edge-caches to store item n.

On the other hand, if Equation (8) does not hold for k'', it means that adding item n to the edge-cache k'' will reduce the average cost. In this case we prove that the set $\mathcal{I}_n'(\mathbf{s}) \cup \{k''\}$ satisfies the sufficient condition for optimality. According to the definition of $\mathcal{I}_n'(\mathbf{s})$ given in Equation (6), and assuming that $s_{k''} > 0$, we will have that:

$$\sum_{k \in \mathcal{I}_n' \cup \{k''\}} s_k > \max\left(0, 1 - \sqrt{\frac{c_a \lambda_n}{\beta c_f p_n}}\right). \tag{9}$$

The sufficient condition for optimality would thus be:

$$s_{k'} \geq (1 - \sum_{k \in \mathcal{I}_n' \cup \{k''\}} s_k) - \frac{c_a \lambda_n}{\beta p_n c_f} \frac{1}{1 - \sum_{k \in \mathcal{I}_n' \cup \{k''\}} s_k},$$

for $\forall k' \subset \mathcal{K} \setminus \{\mathcal{I}_n' \cup \{k''\}\}$. Because of Equation (9), the right hand side is always negative and the sufficient condition for optimality holds. Therefore, the set $\mathcal{I}_n^*(\mathbf{s}) = \mathcal{I}_n'(\mathbf{s}) \cup \{k''\}$ is the optimal set of edge-caches to store item n. ■

The 0-1 knapsack problem in (6) is known to be NP-hard and is generally intractable [13] due to the nature of the inequality constraint. In the rest of this section we focus on solving the generally intractable optimization problem (6) by intelligently choosing the number of edge-caches K and vector $\mathbf{s} = (s_k)_{k=1}^K$ such that the inequality constraints for all $n \in \mathcal{N}$ becomes equality constraints. Doing so will

remove the complexity that arises by knapsack problems in their general form. Our analysis shows that by intelligently choosing the number of edge-caches and the fraction of the load directed to each edge-cache, we can achieve the global minimum average system cost by our proposed caching strategy.

The following theorem provides the optimal caching strategy for the general problem formulated in (4).

Theorem 1: In a system composed of a data set $\mathcal{N}$ of N items with popularity distribution $\mathbf{p} = (p_n)_{n=1}^N$ and update rates $\boldsymbol{\lambda} = (\lambda_n)_{n=1}^N$, assume without loss of generality that items are ordered such that $y_1^* \geq y_2^* \geq ... \geq y_N^*$ where y_n^* is defined as $y_n^* = \max\left(0, 1 - \sqrt{\frac{c_a \lambda_n}{\beta c_f p_n}}\right) \leq 1$. Let $Q = \max(n : y_n^* > 0) \leq N$, then the following caching strategy where $K^* = Q + 1$ optimally solves (4).

$$s_k^* = \begin{cases} y_k^* - y_{k+1}^*, & k \in \{1, 2, \ldots, K^* - 1\}, \\ 1 - y_1^*, & k = K^*, \end{cases} \tag{10}$$

$$\mathcal{I}_n^* = \begin{cases} \{n, \ldots, K^* - 1\}, & n \in \{1, 2, \ldots, K^* - 1\}, \\ \{\}, & n \in \{K^*, \ldots, N\}, \end{cases} \tag{11}$$

where s_k^* and $\mathcal{I}_n^*$ are the fraction of allocated load to the edge-cache k and the set of edge-caches that have stored item n respectively. Under such policy, the optimal caching cost C^* and the upper bound on the cache occupancy B^* are given by:

$$C^*(\lambda, p) = \beta c_f - \sum_{n=1}^{Q} \left(\sqrt{c_a \lambda_n} - \sqrt{\beta c_f p_n}\right)^2, \tag{12}$$

$$B^*(\lambda, p) \leq \frac{1}{2} Q(Q+1). \tag{13}$$

Proof. We start the proof by defining the variable $y_n = \sum_{k \in \mathcal{I}_n} s_k \in [0, 1], \forall n \in \mathcal{N}$ and rewriting the average cost defined in (1) as:

$$C^{\mathcal{D}}((y_n)_n) = \beta c_f + \sum_{n=1}^{N} \left(y_n \left(\frac{c_a \lambda_n}{1 - y_n} - \beta p_n c_f\right)\right).$$

Next, by relaxing the equality constraint $y_n = \sum_{k \in \mathcal{I}_n} s_k$ and letting y_n taken on any arbitrary value in $[0, 1]$, we can write the cost minimization problem as:

$$\begin{aligned} &\min_{(y_n)_n} C^{\mathcal{D}}((y_n)_n), \\ &s.t. \quad 0 \leq y_n \leq 1. \end{aligned} \tag{14}$$

This is a convex optimization problem whose solution is given as:

$$y_n^* = \max\left(0, 1 - \sqrt{\frac{c_a \lambda_n}{\beta c_f p_n}}\right) \leq 1, \quad \forall n \in \mathcal{N}. \tag{15}$$

Using this solution, we intelligently assign $(s_k)_k$, $\{\mathcal{I}_n\}_n$ and K such that:

$$\sum_{k \in \mathcal{I}_n} s_k = y_n^*, \quad \forall n \in \mathcal{N}. \tag{16}$$

Doing so will render all the inequalities in the 0-1 knapsack problems given in Equation (6) to equality constraints. Therefore, Proposition 1 in this case reduces to:

$$\sum_{k \in \mathcal{I}_n^*(\mathbf{s})} s_k = y_n^*, \quad \mathcal{I}_n^*(\mathbf{s}) = \mathcal{I}_n'(\mathbf{s}), \qquad \forall n \in \mathcal{N}.$$

To guarantee that Equation (16) holds $\forall n \in \mathcal{N}$, we define $Q = \max(n : y_n^* > 0) \leq N$ and choose the number of edge-caches as $K^* = Q + 1$ which is the upper bound on the number of different values that y_n^* can take. Then, by hypothesis, since $y_1^* \geq y_2^* \geq ... \geq y_N^*$, we choose $s_k^*, \forall 1 \leq k < K^*$ such that:

$$\begin{aligned} &y_{K^*-1}^* = s_{K^*-1}^*, \\ &y_{K^*-2}^* = s_{K^*-1}^* + s_{K^*-2}^*, \\ &.... \\ &y_1^* = s_{K^*-1}^* + s_{K^*-2}^* + ... + s_1^*. \end{aligned}$$

Comparing this with Equation (16), where $\sum_{k \in \mathcal{I}_n^*} s_k^* = y_n^*$, will give $\mathcal{I}_n^*, 1 \leq n < K^*$ as in (11). Since $\sum_{k \in \mathcal{K}} s_k^* = 1$, then $s_{K^*}^* = 1 - y_1^*$ and no item will be stored in this edge-cache. Replacing the results in the average cost given in Equation (1) yields the optimal cost $C^*(\lambda, p)$ as in Equation (12). Finally, the upper bound on cache occupancy of the optimal policy is given by:

$$B^*(\lambda, p) = \sum_{n=1}^{N} |\mathcal{I}_n^*(\mathbf{s}^*)| \leq Q + (Q-1) + ... + 1 = \frac{1}{2} Q(Q+1), \tag{17}$$

and the inequality in the equation is due to the fact that s_k^* can be zero for some $k \in \mathcal{K}$. In that case, the edge-cache with no load will store no item in its cache. This completes the proof. ■

Remark 1: The extra edge-cache $k = K^*$ that does not cache any items, contributes to enhancing content freshness since all the load directed to this edge-cache is served fresh from the database. Due to the multicasting, this acts as a freshness mechanism to keep the content in other edge-caches from getting obsolete.

In the following we investigate some special cases.

Proposition 2: In the special case of $\frac{\lambda_n}{p_n} = \frac{\lambda}{p}, \forall n \in \mathcal{N}$, the optimal caching strategy is to have two edge-caches, i.e., $K^* = 2$ where the load is split according to $s_1^* = \max\left(0, 1 - \sqrt{\frac{c_a \lambda}{\beta c_f p}}\right)$ and $s_2^* = 1 - s_1^*$. In the case of $s_1^* > 0$, the first edge-cache will store All the N items in its cache, i.e., $\mathcal{I}_n^* = \{1\}, \forall n \in \mathcal{N}$, otherwise if $s_1^* = 0$, the first edge-cache will store no items, i.e., $\mathcal{I}_n^* = \{\varnothing\}, \forall n \in \mathcal{N}$. The second edge-cache will never store any items. The purpose of the second edge-cache is to utilize multicasting as a freshness mechanism to keep the cached content of the first edge-cache from getting obsolete.

Proof. Since $\frac{\lambda_n}{p_n} = \frac{\lambda}{p}, \forall n \in \mathcal{N}$, according to Equation (15) all $y_n^*, \forall n \in \mathcal{N}$ will be identical.

$$y_1^* = y_2^* = ... = y_N^* = \max\left(0, 1 - \sqrt{\frac{c_a \lambda}{\beta c_f p}}\right).$$

Now we show how Equation (16) holds $\forall n \in \mathcal{N}$. In case of $y_1^* = 0$, no caching will be employed, i.e., $\mathcal{I}_n^* = \{\varnothing\}, \forall n \in \mathcal{N}$. But if $y_1^* > 0$, then since all y_n^* have same value, we can guarantee that Equation (16) holds for $\forall n \in \mathcal{N}$ by choosing an edge-cache that receives the fraction $s_1^* = y_1^*$ and then placing all the items in this edge-cache, i.e., $\mathcal{I}_n^* = \{1\}, \forall n \in \mathcal{N}$. The other edge-cache will receive the remaining load and will have no items stored in its cache. ■

The case of uniform popularity with constant refresh rates is a special case of this. According to Proposition 2, the optimal policy in this case will deploy two edge-caches over the network and split the load and cache space unequally between those edge-caches, even though the popularity and refresh rates are uniform. This reveals the counter-intuitive nature of the optimal policy that benefits by splitting the load and cache capacity unequally between the edge-caches to fully leverage the wireless broadcast as a free cache update mechanism.

Proposition 3: In the case of item popularity distributed according to Zipf with parameter z and constant update rates, i.e., $p_n = \frac{p_0}{n^z}$ and $\lambda_n = \lambda, \forall n \in \mathcal{N}$ with $p_0 = \frac{1}{\sum_{n=1}^{N} \frac{1}{n^z}} < 1$, the proposed optimal caching strategy of Theorem 1 reduces to:

$$s_k^* = \begin{cases} \sqrt{\frac{c_a\lambda}{\beta c_f p_0}}(\sqrt{(k+1)^z} - \sqrt{k^z}), k \in \{1, \dots, K^* - 2\}, \\ 1 - \sqrt{\frac{c_a\lambda}{\beta c_f p_0}}\sqrt{k^z}, \qquad k = K^* - 1, \\ \sqrt{\frac{c_a\lambda}{\beta c_f p_0}} \qquad k = K^*, \end{cases} \tag{18}$$

where $K^* = \min\left(\lfloor \sqrt[z]{\frac{\beta c_f p_0}{c_a \lambda}} \rfloor, N\right) + 1$,

Proof. For the case of Zipf popularity distribution with constant update rate, $p_1 \geq p_2 \geq ... \geq p_N$, then y_n^* given in (15) can be written as:

$$y_n^* = \max\left(0, 1 - \sqrt{\frac{c_a\lambda}{\beta c_f p_0}}\sqrt{n^z}\right) \leq 1, \quad \forall n \in \mathcal{N}, \tag{19}$$

which results in Q as:

$$Q = \max\left(n : y_n^* > 0\right) = \min\left(\lfloor \sqrt[z]{\frac{\beta c_f p_0}{c_a \lambda}} \rfloor, N\right)$$

which gives $K^* = Q + 1$. Replacing y_n^* in Equation (10) will give the s_k^* as in (18). ■

This reveals very interesting insights on the nature of the proposed optimal policy. The optimal policy will split the load unequally between the edge-caches and will completely discard the less popular items, i.e., less popular items will not be stored in any of the edge-caches. More interestingly, edge-caches with higher load will generally store less items in their cache, however, they are the more popular ones. This is counter-intuitive, because one may guess that putting more items on the edge-caches with higher load will result in cost reduction over the network. However, the optimal policy which aims to minimize the cost by balancing the freshness and fetching cost, not only avoids to fill up the edge-caches with higher load, but it puts less items into edge-caches as their load increases. Yet, by intelligently deciding to put the most popular items into edge-caches with higher load while keeping the cache small, the optimal policy achieves the optimal cost over the network.

Remark 2: In the special case of Zipf popularity distribution with parameter $z = 2$, the optimal caching strategy is to have $K^* = \lfloor \sqrt{\frac{\beta c_f p_0}{c_a \lambda}} \rfloor + 1$ and then split the load equally between the edge-caches such that each edge-cache receives the fraction $s_k = \sqrt{\frac{c_a\lambda}{\beta c_f p_0}} \approx \frac{1}{K^*}$ of the total load.

The results of Remark 2 motivates us to investigate the performance of *equal* load-splitting more deeply. It may not always be possible to split the load unequally between the edge-caches due to complexity of the implementation. In the next section, we propose an optimal policy for the special case of the equal load-splitting and investigate under what conditions such a policy can be beneficial.

IV. Optimal Distributed Caching for Equal Load-Splitting of Dynamic Content

In this section, we attack the problem in (4) for the special case when the total number of edge-caches $|\mathcal{K}| = K$ is given and the load is split equally between the edge-caches. In this case, $s_k = \frac{1}{K}, \forall k \in \mathcal{K}$. This equal load-splitting is simple to implement and yields interesting insights on the cost and cache occupancy trade-offs. We first characterize the optimal caching strategy and then provide insights on the cache occupancy of the proposed strategy.

For K edge-caches, each receiving a fraction $s_k = \frac{1}{K}, k \in \{1, 2, ..., K\}$ of the total load, we define $r_n = |\mathcal{I}_n|$ to be the number of edge-caches that have stored item n and let $\mathbf{r} = (r_1, ..., r_N)$ be the vector of replication.

Define the feasible set of solutions as:

$$\mathcal{F}_K = \{\mathbf{r} = (r_1, \dots, r_N) \,|\quad r_n \in \{0, 1, \dots, K\}\},$$

where each item can be stored at most once in each edge-cache.

Lemma 2: Let $C^{\mathcal{S}}(K, \mathbf{r})$ be the average expected system cost in the equal load-splitting scenario with K edge-caches and vector of replication $\mathbf{r} \in \mathcal{F}_K$. Then:

$$C^{\mathcal{S}}(K, \mathbf{r}) = \beta c_f + \sum_{n=1}^{N} r_n \left(\frac{c_a \lambda_n}{K - r_n} - \frac{\beta p_n c_f}{K}\right). \tag{20}$$

Proof. Since the number of replica r_n is defined to be $r_n = |\mathcal{I}_n|$, and in the load is split equally between the K edge-caches, we have: $\sum_{k \in \mathcal{I}_n} s_k = \frac{r_n}{K}$.

Replacing this in the general cost defined in Lemma 1 gives the average cost of the caching system with K edge-caches and under vector of replication $\mathbf{r}$ as $C^{\mathcal{S}}(K, \mathbf{r})$. ■

Our objective is thus to choose the content to be stored at the K edge-caches in order to minimize the average cost of the system, that is:

$$\min_{\mathbf{r} \in \mathcal{F}_K} C^{\mathcal{S}}(K, \mathbf{r}). \tag{21}$$

Proposition 4: The policy $\mathbf{r}^* = (r_n^*)_n \in \mathcal{F}_K$ that solves (21) is given by:

$$r_n^* = \lfloor K + \frac{1}{2} - \sqrt{\frac{1}{4} + K^2 \frac{c_a \lambda_n}{\beta c_f p_n}} \rfloor^+, \quad \forall n \in \mathcal{N}, \tag{22}$$

where $\lfloor x \rfloor^+ = \max(0, \lfloor x \rfloor)$, and $\lfloor x \rfloor$ is the greatest integer less than or equal to x.

Proof. We define $\delta_n^{\mathcal{S}}(l)$ to be the marginal cost of adding item n to the caches given that l of the edge-caches have already cached item n. In other words:

$$\delta_n^{\mathcal{S}}(l) := C^{\mathcal{S}}(K, \mathbf{r})|_{r_n = l+1} - C^{\mathcal{S}}(K, \mathbf{r})|_{r_n = l}$$

Therefore, we have:

$$\delta_n^{\mathcal{S}}(l) = \frac{Kc_a\lambda_n}{(K-l)(K-l-1)} - \frac{\beta p_n c_f}{K} \quad \forall n \in \mathcal{N}. \quad (23)$$

In the case of $\delta_n^{\mathcal{S}}(l) < 0$ for a given integer l, adding item n to one more edge-cache will decrease the average cost. On the other hand, items with positive $\delta_n^{\mathcal{S}}(l)$ can only increase the average cost if cached. Therefore, we can add item n to the edge-caches, as long as $\delta_n^{\mathcal{S}}(l)$ is negative. Such $\delta_n^{\mathcal{S}}(l)$ reveals the effect of refresh rate alongside the popularity on gains that can be achieved by caching an item. The optimal caching strategy will keep filling the cache for each item n until $\delta_n^{\mathcal{S}}(l)$ turns positive. Therefore, the optimal number of replica for item n would be:

$$r_n^* = 1 + \max\{l \in \{0, 1, ...\} : \delta_n^{\mathcal{S}}(l) < 0\}, \forall n \in \mathcal{N}.$$

Using $\delta_n^{\mathcal{S}}(l)$ defined in (23) yields r_n^* as (22). ■

Next we study the trade-off between the average system cost and cache occupancy of the optimal policy for the equal load-splitting compared to the optimal caching policy for the general case when the load is allowed to be split unequally between the edge-caches.

Proposition 5: In a system composed of a data set $\mathcal{N}$ of N items with popularity distribution $\mathbf{p} = (p_n)_{n=1}^N$ and update rates $\boldsymbol{\lambda} = (\lambda_n)_{n=1}^N$, assume [3] $y_1^* \geq y_2^* \geq ... \geq y_N^*$ where $y_n^*, \forall n \in \mathcal{N}$ is defined in Equation (15). Let $Q = \max(n : y_n^* > 0) \leq N$ which is independent of the number of edge-caches K, then we have:

$$C^{\mathcal{S}}(K, \mathbf{r}^*) - C^* \leq \left(2\beta c_f \sqrt{\frac{\beta c_f}{c_a}} \sum_{n=1}^{Q} \sqrt{\frac{p_n^3}{\lambda_n}}\right) \frac{1}{K^2}, \quad (24)$$

where $C^{\mathcal{S}}(K, \mathbf{r}^*)$ is the optimal cost in the equal load-splitting and C^* is the minimum achievable cost in Theorem (1). Also we have:

$$B^{\mathcal{S}}(K, \mathbf{r}^*) - B^*(\lambda, p) \geq \sum_{n=1}^{Q} \lfloor K + \frac{1}{2} - \sqrt{\frac{1}{4} + K^2 \frac{c_a\lambda_n}{\beta c_f p_n}} \rfloor^+ - \frac{1}{2}Q(Q+1), \quad (25)$$

where $B^{\mathcal{S}}(K, \mathbf{r}^*)$ is the cache occupancy under the optimal policy in the equal load-splitting and $B^*(\lambda, p)$ is the cache occupancy of the proposed optimal policy in Theorem (1).

Proof. To prove Equation (24), we use the following Taylor approximation.

$$C^{\mathcal{S}}(K, \mathbf{r}^*) \leq C^* + \beta c_f \sum_{n=1}^{N} p_n |\frac{r_n^*}{K} - y_n^*| (1 - \frac{1}{1 + |\frac{r_n^*}{K} - y_n^*| \sqrt{\frac{\beta c_f p_n}{c_a \lambda_n}}})$$
$$\leq C^* + 2\beta c_f \sqrt{\frac{\beta c_f}{c_a}} \sum_{n=1}^{N} (\sqrt{\frac{p_n^3}{\lambda_n}} |\frac{r_n^*}{K} - y_n^*|^2)$$

where $y_n^* = \max(0, 1 - \sqrt{\frac{c_a\lambda_n}{\beta c_f p_n}})$. In the case when $y_n^* = 0$, we have $\frac{c_a\lambda_n}{\beta c_f p_n} \geq 0$ and according to Equation (22), $r_n^* = 0$, which gives $|\frac{r_n^*}{K} - y_n^*| = 0$. In the case when $y_n^* > 0$, we can show that $|Ky_n^* - r_n^*| < 1$ which gives $|\frac{r_n^*}{K} - y_n^*| < \frac{1}{K}$. Since, by hypothesis, $y_1^* \geq y_2^* \geq ... \geq y_N^*$, we can write:

$$C^{\mathcal{S}}(K, \mathbf{r}^*) - C^* \leq \left(2\beta c_f \sqrt{\frac{\beta c_f}{c_a}} \sum_{n=1}^{Q} \sqrt{\frac{p_n^3}{\lambda_n}}\right) \frac{1}{K^2},$$

where the terms inside the parentheses are independent of K and this shows a cost reduction with the rate $\frac{1}{K^2}$.

Also, since the cache occupancy of equal load-splitting is equal to $B^{\mathcal{S}}(K, \mathbf{r}^*) = \sum_{n=1}^{N} r_n^*$, where r_n^* is given in Equation (22), and as we showed that in the case of $y_n^* = 0$, we have $r_n^* = 0$, therefore, using the definition of Q and the lower bound on the cache occupancy of our proposed optimal policy in (13), we can write the lower bound on the cache saving of our proposed policy compared to the equal load-splitting as in Equation (25). ■

This shows that as the number of edge-caches K increases, the cost of the equal load-splitting converges to the optimal cost with rate $\frac{1}{K^2}$ but its cache occupancy increases with the rate of up to K^2.

V. Numerical Results: Performance Comparison

In this section we compare the performance of the equal load-splitting to the optimal case of general load-splitting between the edge-caches using numerical simulations. We consider the simulation parameters to be $\beta = 5$ for the average total request rate and the normalized fetching and aging costs to be $c_f = 1$ and $c_a = 0.01$ respectively. We assume that the database consists of $N = 10^6$ items.

We compare the average cost achieved by the optimal caching policy and the average cost of the equal load-splitting policy under the number of edge-caches $K = K^*$ and the same system variables declared above. We adopt the percentage cost gain of the optimal caching to the equal load-splitting strategy's cost as our performance metric. Such a metric is defined as:

$$\text{Cost Gain}(\%) = 100 \times \frac{C^{\mathcal{S}}(K^*, \mathbf{r}^*) - C^*}{C^*}.$$

The percentage cost gain is depicted in Fig. 3. The figure shows that gains are negligible for small Zipf parameters. In other words, if the item demand is less predictable, i.e., more uncertainty about the demand, the equal load-splitting policy performs almost as good as the optimal caching and the difference vanishes as the refresh rate decreases and items become less dynamic. But as the Zipf parameter increases and the certainty about the demand increases, the cost reduction gain increases. It also reveals that the gain becomes more substantial as the refresh rate of items decreases. The figure reveals a dip in the cost reduction gain at the Zipf parameter $z = 2$, which agrees with the results of the Remark 2.

Next we compare the cache occupancy of the optimal caching policy and the equal load-splitting policy under the number of edge-caches $K = K^*$ and the same system variables declared above. We adopt the percentage cache loss of the optimal caching to the equal load-splitting strategy's

[3] This already holds without assumption in Zipf with constant refresh rates.

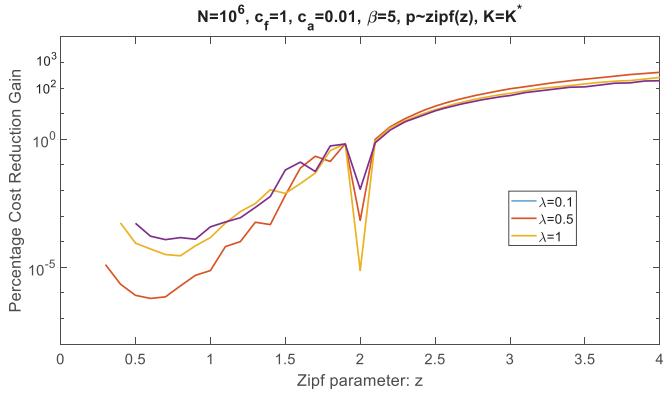

Fig. 3: Percentage cost gain of the optimal caching policy

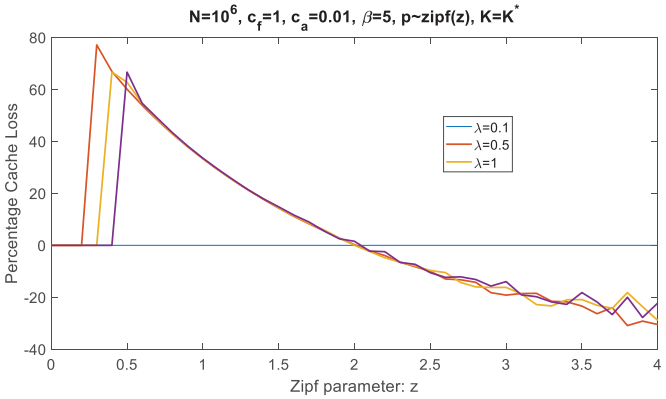

Fig. 4: Percentage cache loss of the optimal caching policy

cache occupancy as our performance metric. Such a metric is defined as:

$$\text{Cache Loss } (\%) = 100 \times \frac{B^* - B^{\mathcal{S}}(K^*, \mathbf{r}^*)}{B^*}.$$

The percentage cache loss is depicted in Fig. 4. The figure shows that for small Zipf parameters, more uncertainty about the demand, equal load-splitting policy occupies significantly less cache space compared to the optimal caching policy and the differences increases as the refresh rate decreases and items become less dynamic. If parameter z approaches zero, content popularity becomes almost uniform and both policies may decide not to cache any items at all. On the other hand, as the Zipf parameter increases and certainty about demand increases, the percentage cache loss of the optimal policy decreases and the optimal policy which achieves the global minimum cost, will occupy less cache space compared to the equal load-splitting policy. The figure reveals that both optimal and equal load-splitting policies achieve almost same cost and same cache sizes at the Zipf parameter $z = 2$, which agrees with Remark 1.

According to Fig. 3 and 4, when item demand is less predictable, i.e., $z < 2$, equal load-splitting policy achieves almost the same average cost of the optimal caching policy while potentially saving in the cache occupancy. On the other hand, as item demand becomes more predictable, i.e., $z > 2$, optimal caching policy results in substantial gains in the caching cost while simultaneously reducing the cache occupancy.

Notice that in Figs. 3 and 4, we have assumed $K = K^*$ both for the optimal and equal load-splitting policies. According to Proposition 5, increasing K for the equal load-splitting policy, such that $K > K^*$, the resulting average cost approaches the optimal cost but this is achieved at the expense of increasing the cache occupancy.

VI. Conclusion

In this work, we have proposed and investigated an increasingly important caching scenario for serving dynamically changing content. We introduced the *age-of-version* metric to capture the served content's freshness and track the number of stale versions per content. We have addressed the problem of developing optimal caching strategies for minimizing the system's cost which is shaped by a combination of the service cost of fetching fresh content directly from a back-end database and the aging cost of cached, potentially older, content from a front-end cache. By utilizing the broadcast nature of the wireless medium, our model reveals the benefits of the multicasting property as a mechanism to update the cached content. We have characterized the optimal caching policy both in the general case and also in the special case of the equal load-splitting. Moreover, we have explored the trade-off between the cost minimization and cache savings gain of these two policies. Our results demonstrate that for more predictable demand, splitting the cache and load unequally between the edge-caches results in significant cost gains without increasing the total cache occupancy. On the other hand, for less predictable demand, equal load-splitting achieves a close-to-optimal cost while saving in cache occupancy.

References

[1] B. Abolhassani, J. Tadrous, and A. Eryilmaz, "Delay gain analysis of wireless multicasting for content distribution," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 529–542, 2020.

[2] ——, "Wireless multicasting for content distribution: Stability and delay gain analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1–9.

[3] J. Zhang, "A literature survey of cooperative caching in content distribution networks," *arXiv preprint arXiv:1210.0071*, 2012.

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

[5] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1382–1393, 2016.

[6] B. Abolhassani, J. Tadrous, and A. Eryilmaz, "Achieving freshness in single/multi-user caching of dynamic content over the wireless edge," in *IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2020.

[7] B. Abolhassani, J. Tadrous, A. Eryilmaz, and E. Yeh, "Fresh caching for dynamic content," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[8] E. Najm and R. Nasser, "Age of information: The gamma awakening," in *2016 IEEE International Symposium on Information Theory (ISIT)*. Ieee, 2016, pp. 2574–2578.

[9] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Information freshness and popularity in mobile caching," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 136–140.

[10] J. Zhong, R. D. Yates, and E. Soljanin, "Two freshness metrics for local cache refresh," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1924–1928.

[11] D. Wessels, *Web caching*. " O'Reilly Media, Inc.", 2001.

[12] [Online]. Available: http://www2.ece.ohio-state.edu/~eryilmaz/papers/FreshCachingReport2021

[13] S. Sahni, "Approximate algorithms for the 0/1 knapsack problem," *Journal of the ACM (JACM)*, vol. 22, no. 1, pp. 115–124, 1975.