Optimal Load-Splitting and Distributed-Caching for Dynamic Content over the Wireless Edge

Bahman Abolhassani, John Tadrous, Atilla Eryilmaz

***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 This reduces the complex problem to a solvable special case. 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]. 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 [2], [3]. 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 [4], [5]. Content Distribution Networks (CDNs) utilize a large mesh of edge-caches to deliver content from locations closer to the end users [6], [7]. 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 [8], [9]. These works are based on the promise that the content stored in the cache will ultimately be used (see [10], [11] and [12]). An important factor that may greatly affect the caching decision is the content generation dynamics. 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 [13]. In these types of dynamic contents, users prefer to have the most fresh version of the content while also making sure that the total cost of the network remains low [14], [15]. 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 such as [13], [16]–[31] and effective strategies have been proposed. In particular, authors in [26] 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 [32]. As such, these works focus mainly on minimizing the miss rate [21] or minimizing the average age of the cached content (see [33], [16]). Kam et al. [21] 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 [34]–[37], there are many real-world scenarios where a content does not lose its

Manuscript received November 15, 2021; revised August 29, 2022, and accepted January 11, 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Longbo Huang. Date of publication –, 2023; date of current version January 11, 2023. This work is supported in part by the NSF grants: NSF AI EDGE Institute grant 2112471, CNS-NeTS-2106679, CNS-NeTS-2007231, CNS-SpecEES-1824337; and the ONR Grant N00014-19-1-2621;

B. Abolhassani and A. Eryilmaz are with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail:abolhassani.2@osu.edu; eryilmaz.2@osu.edu).

J. Tadrous is with the Department of Electrical and Computer Engineering, Gonzaga University, Spokane, WA 99202 (e-mail:tadrous@gonzaga.edu).

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. Kompella et al. [38] study the information freshness at the monitoring station for applications that rely on information freshness such as networked monitoring and automated control (e.g., tactical networks, sensor networks, airplane/vehicular control, and Cyber-Physical Systems) applications. They conclude that while AoI is indeed a good metric for freshness, optimizing the AoI metric does not always guarantee optimal signal reconstruction at the monitoring station [39].

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 [40] which grows with the AoV metric. Moreover, our model utilizes the multicasting 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 [13], which also considered optimal distributed caching over the wireless edge. However, the setting in [13] 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. A special case of the problem comprising only one user group was investigated in [41] and an optimal caching policy was presented for this special case. In the current version we extend the previous model by allowing multiple user groups, each with a different popularity profile, request dynamic content from a shared database. We also add channel failure to incorporate multicast outage for a more realistic system model [42], [43].

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. Such cache update mechanism is subject to wireless channel failure and is not always successful.
- 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. We show that the general case of multiple user groups can be simplified to a single user group with the aggregated demand profile, letting us to focus on the single user group in the rest of paper. Also, 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.

II. System Model

We consider the generic hierarchical setting depicted in Fig. 1, whereby: the (limited) local cache serves multiple groups of user population where each user group generates

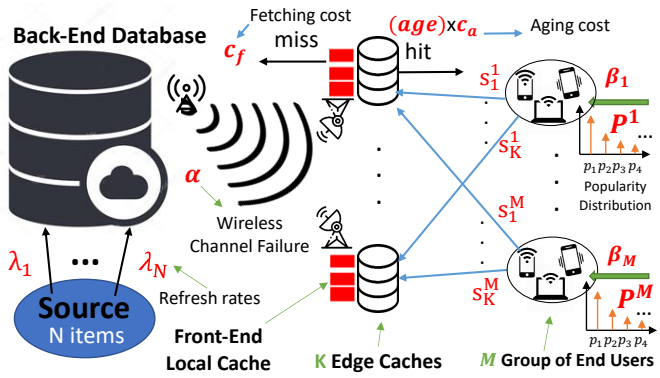

Fig. 1: Setting of *Fresh Caching for Dynamic Content*

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) are being served to the user population through a hierarchical caching system as depicted in Fig. 1. In particular, a set $\mathcal{K}$ of K edge-caches are distributed over the network that supply local content to the neighboring users. Users are divided into different groups based on their interest, e.g., different age segments, cultural background, etc. The set $\mathcal{M}$ of M user groups with different popularity profile request content from edge caches. Each user group $m \in \mathcal{M}$ generates requests according to a Poisson process[1] with rate $\beta_m \geq 0$, which captures the request intensity of that user population served by the edge-caches. An incoming request from user group $m \in \mathcal{M}$ targets data item $n \in \mathcal{N}$ with probability p_n^m. Accordingly, the probability distribution $\mathbf{p}^m = (p_n^m)_{n=1}^N$ captures the popularity profile of the data items for user group m. Furthermore, for each user group $m \in \mathcal{M}$, we define s_k^m to be the fraction of the user group m's requests served by the edge cache $k \in \mathcal{K}$. Accordingly, the vector $\mathbf{s}^m = (s_k)_{k=1}^K, \forall m \in \mathcal{M}$ captures the load-splitting between the edge-caches for the user group m.

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 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, due to the wireless channel imperfections, such cache update mechanism is subject to failure. We assume α is the probability that each edge cache can successfully update its cache when the content is being served through multicast to another edge cache. These successful update events are independent over the edge caches. As such, the age of item n at the edge cache k, i.e., $\Delta_n^k(t)$, would be different among the edge-caches that hold item n. Furthermore, we assume that upon each failed fetching attempt, the database will reattempt fetching until the content is successfully delivered to the user requesting that content, not disregarding any user request.

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. For each user group $m \in \mathcal{M}$, the arrival request rate β_m is split between the edge-caches, such that each edge-cache k receives a fraction s_k^m of the total incoming requests. Therefore, $\mathbf{s}^m = (s_k^m)_{k=1}^K, \forall m \in \mathcal{M}$ is the vector of load-splitting between the edge-caches for user group m where $\sum_{k=1}^K s_k^m = 1, \forall m \in \mathcal{M}$.

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

[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.

Lemma 1: Let $C^{\mathcal{D}}(\{\mathcal{I}_n\}_n, \{\mathbf{s}^m\}_m)$ 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 user group m sends the fraction s_k^m of its request to the edge-cache $k \in \mathcal{K}$. Then:

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

Proof. Let $\{\Pi_{\mathcal{I}_n}^{n,k}(t), t \geq 0\}$, $\forall n \in \mathcal{N}$ be the Markov process describing the freshness age of cached item n stored at the edge cache $k \in \mathcal{K}$ at time t under the cached set $\mathcal{I}_n$. The evolution of this process is shown in Fig. 2 where $\mu_n = \sum_{\mathrm{m}=1}^{M}\left(\beta_m p_n^m\left(1-\sum_{k\in\mathcal{I}_n} s_k^m\right)\right)$. As

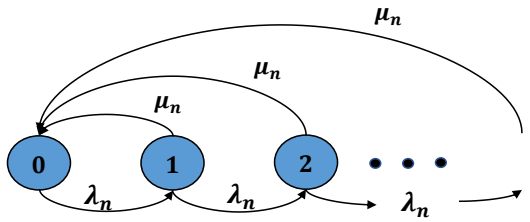

Fig. 2: Markov chain diagram for freshness $\{\Pi_{\mathcal{I}_n}^{n,k}(t), t \geq 0\}$ of item n at edge cache $k \in \mathcal{I}_n$ under the cached set $\mathcal{I}_n$.

discussed earlier, in the distributed edge caching scenario, the broadcast capability of wireless service acts as a natural update mechanism. In other words, edge-caches update their cached content for free by overhearing that content while being fetched to the users requesting that content at the time of miss event. For any item n in the cache, since $\mathcal{I}_n$ is the set of edge-caches that have stored item n, the fraction $\sum_{k\in\mathcal{I}_n} s_k^m, \forall m \in \mathcal{M}$ of the total request for item n generated by user group m is served from the edge-caches. Thus, $1-\sum_{k\in\mathcal{I}_n} s_k^m, \forall m \in \mathcal{M}$ is the miss probability for item n due to the incoming request from user group m for that item to those edge-caches that have not stored item n. Each miss event for item n triggers a fetching from the database which is successfully delivered to the user with probability α. Upon each failed fetching attempt, we will reattempt the fetching until the content is successfully delivered to the requesting user. Thus, resulting in a service rate of $\frac{\beta_m p_n^m}{\alpha}(1-\sum_{k\in\mathcal{I}_n} s_k^m), \forall m \in \mathcal{M}$ for that item from user group m. As it can be seen in Fig. 2, every service of item n acts as an update mechanism for the edge-caches that hold item n in their cache and upon occurrence, the multicast service of item n from the database, with probability α will move the system back to state zero, the most fresh version. This is due to the fact that each multicast service is successfully received by edge caches with probability α. Thus, the effective cache update rate resulted by user group m is $\alpha\frac{\beta_m p_n^m}{\alpha}(1-\sum_{k\in\mathcal{I}_n} s_k^m) = \beta_m p_n^m(1-\sum_{k\in\mathcal{I}_n} s_k^m)$. Adding the cache update rate over all the users, give the total rate $\mu_n = \sum_{\mathrm{m}=1}^{M}\sum_{n=1}^{N}\left(\frac{\beta_m p_n^m}{\alpha}\left(1-\sum_{k\in\mathcal{I}_n} s_k^m\right)\right)$ to state zero. Every arriving content update to the item n in the database that occurs with rate λ_n increments the age of that item in the cache by one.

Since $\Pi_{\mathcal{I}_n}^{n.k}(t) \xrightarrow[t\to\infty]{d} \bar{\Pi}_{\mathcal{I}_n}^{n}, \forall k \in \mathcal{I}_n$, and using the steady state distribution of $\Pi_{\mathcal{I}_n}^{n,k}(t)$, define $\pi_i^n(\mathcal{I}_n) = P(\bar{\Pi}_{\mathcal{I}_n}^{n} = i), i \in \{0,1,2,\ldots\}$ to be the probability of item n having the age of i under the cached set $\mathcal{I}_n$. Then the average age of item n is given by:

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

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 of each user group m is split between the K edge-caches according to the vector $\mathbf{s}^m = (s_1^m, ..., s_K^m)$, $\forall m \in \mathcal{M}$, comprises two main terms and is given by:

$$C^{\mathcal{D}}(\{\mathcal{I}_n\}_n, \{\mathbf{s}^m\}_m) = c_f \sum_{\mathrm{n}=1}^{N}\sum_{m=1}^{M}\left(\frac{\beta_m p_n^m}{\alpha}\left(1-\sum_{k\in\mathcal{I}_n} s_k^m\right)\right) + c_a \sum_{\mathrm{n}=1}^{N}\mathbb{E}[\bar{\Pi}_{\mathcal{I}_n}^{n}]\sum_{\mathrm{m}=1}^{M}\left(\beta_m p_n^m\left(\sum_{k\in\mathcal{I}_n} s_k^m\right)\right). \tag{3}$$

The first term in Equation (3), shows the average *fetching cost* under any cached set $\{\mathcal{I}_n\}_n$ as a function of the total miss rate $\sum_{\mathrm{m}=1}^{M}\sum_{n=1}^{N}\left(\beta_m p_n^m\left(1-\sum_{k\in\mathcal{I}_n} s_k^m\right)\right)$. For any of the K edge-caches, if a requested item is in the edge-cache of the user requesting that item, it will be immediately served from the cache with the freshness cost, otherwise it will be fetched from the database and the urgent fetching cost c_f is incurred. Each fetching attempt is successful with probability α and we will keep attempting to fetch until the content is successfully delivered to the user requesting that content.

The second term in Equation (3) shows the average *freshness cost* under any cached set $\{\mathcal{I}_n\}_n$. For each item n, the total arrival request rate is $\sum_{m=1}^{M}\beta_m p_n^m$ and the fraction $\sum_{\mathrm{m}=1}^{M}\left(\beta_m p_n^m\left(\sum_{k\in\mathcal{I}_n} s_k^m\right)\right)$ will be served from the local cache with a freshness cost. since the item with age i incurs the cost of $i \cdot c_a$, the average freshness cost for item n will be $c_a\mathbb{E}[\bar{\Pi}_{\mathcal{I}_n}^{n}]$. Thus, the total freshness cost incurred by item n is given by $c_a\mathbb{E}[\bar{\Pi}_{\mathcal{I}_n}^{n}]\left(\sum_{\mathrm{m}=1}^{M}\left(\beta_m p_n^m\left(\sum_{k\in\mathcal{I}_n} s_k^m\right)\right)\right)$. Summing over all the items gives the total freshness cost of the system. Substituting Equation (2) in Equation (3) and changing the order of summation, gives the average cost of the system. ■

The cost minimization problem for such system would thus be:

$$\min_{\{\mathbf{s}^m\}_m,\{\mathcal{I}_n\}_n,K} C^{\mathcal{D}}(\{\mathcal{I}_n\}_n,\{\mathbf{s}^m\}_m),$$
$$s.t. \quad 0 \leq s_k^m \leq 1, \forall\, m \in \mathcal{M}, k \in \mathcal{K},$$
$$\sum_{k=1}^{K} s_k^m = 1,\ \forall m \in \mathcal{M}, \tag{4}$$
$$\mathcal{I}_n \subseteq \mathcal{K},$$
$$K \geq 0.$$

Minimizing the average caching cost requires finding the optimal value for the number of edge-caches, how to split the load of each user group 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 consider the case when there is only one user group with the vector of load-splitting $\mathbf{s} = (s_1, ..., s_K)$. Moreover, we tackle the problem in a special case by assuming that the number of edge-caches K and the vector of load-splitting $\mathbf{s} = (s_1, ..., s_K)$ is 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:

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

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

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

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:

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

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:

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

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:

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

which gives the condition as:

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

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:

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

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:

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

The sufficient condition for optimality would thus be:

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

for $\forall k' \subset \mathcal{K} \backslash \{\mathcal{I}'_n \cup \{k''\}\}$. 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 [44] due to the nature of the inequality constraint. In the rest of this section, we consider the general case of the M user groups and focus on solving the typically intractable optimization problem (6) by intelligently choosing the number of edge-caches K and vectors $\mathbf{s}^m = (s^m_k)_{k=1}^K, \forall m \in \mathcal{M}$ 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 sufficient condition for a caching strategy to optimally solve the general problem formulated in (4).

Theorem 1: In a system composed of a data set $\mathcal{N}$ of N items with update rates $\boldsymbol{\lambda} = (\lambda_n)_{n=1}^N$ and a set $\mathcal{M}$ of M user groups with popularity distribution $\mathbf{p}^m = (p^m_n)_{n=1}^N, \forall m \in \mathcal{M}$, 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 $y^*_n = \left(\sum_{m=1}^M \beta_m p^m_n\right) \max\left(0, 1 - \sqrt{\frac{\alpha c_a \lambda_n}{c_f \sum_{m=1}^M \beta_m p^m_n}}\right)$. Let $Q = \max(n : y^*_n > 0) \leq N$, then any caching strategy that satisfies the following, where $K^* = Q + 1$, optimally solves (4).

$$\sum_{m=1}^M \beta_m p^m_n (s^m_k)^* = \begin{cases} y^*_k - y^*_{k+1}, & k \in \{1, \ldots, K^* - 1\} \\ \sum_{m=1}^M \beta_m p^m_n - y^*_1, & k = K^*, \end{cases} \tag{10}$$

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

where $(s^m_k)^*$ and $\mathcal{I}^*_n$ are the fraction of allocated user group m's 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:

$$C^*(\lambda, p) = \frac{c_f}{\alpha} \sum_{m=1}^M \beta_m - \sum_{n=1}^Q \left(\sqrt{c_a \lambda_n} - \sqrt{\frac{c_f}{\alpha} \sum_{m=1}^M \beta_m p^m_n}\right)^2, \tag{12}$$

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

Proof.

We start the proof by defining the variable $y_n = \sum_{k \in \mathcal{I}_n} \sum_{m=1}^M \beta_m p^m_n s^m_k, \forall n \in \mathcal{N}$, where it captures the fraction of total requests for item n that are served from the edge caches. Next, we rewrite the average cost defined in (1) as:

$$C^{\mathcal{D}}((y_n)_n) = \sum_{n=1}^N \left(y_n \left(\frac{c_a \lambda_n}{\sum_{m=1}^M \beta_m p^m_n - y_n} - \frac{c_f}{\alpha}\right)\right) + \frac{c_f}{\alpha} \sum_{m=1}^M \beta_m.$$

Due to the dynamic nature of the content, items that are placed in the cache must frequently receive updates to prevent them from getting obsolete. In our model, such cache updates occur when there is a miss event, i.e., a request arrives to an edge cache for an item that is not in the cache. Therefore, there must be a balance between serving requests from the cache and fetching them from the database due to miss events. Accordingly, y_n, which captures the fraction of total requests for item n that are served from the edge caches, is the parameter that should be optimized. Optimizing the cost over y_n for all data items, can be expressed as:

$$\begin{aligned} &\min_{(y_n)_n} C^{\mathcal{D}}((y_n)_n), \\ &s.t. \quad 0 \leq y_n \leq \sum_{m=1}^M \beta_m p^m_n. \end{aligned} \tag{14}$$

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

$$y^*_n = \left(\sum_{m=1}^M \beta_m p^m_n\right) \max\left(0, 1 - \sqrt{\frac{\alpha c_a \lambda_n}{c_f \sum_{m=1}^M \beta_m p^m_n}}\right). \tag{15}$$

On the other hand, y_n is a function of s^m_k where s^m_k is the fraction of the user group m's requests served by the edge cache k. Therefore, to achieve the minimum cost, we must choose $s^m_k, \forall k, m$ such that total requests for each item n that are served from the edge caches is y^*_n. Taking that into account, we intelligently assign $\{\mathbf{s}^m\}_m$, $\{\mathcal{I}_n\}_n$ and K to guarantee that the following holds for all data items.

$$\sum_{k \in \mathcal{I}_n} \sum_{m=1}^M \beta_m p^m_n s^m_k = y^*_n, \quad \forall n \in \mathcal{N}. \tag{16}$$

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

$$\sum_{k \in \mathcal{I}^*_n(\mathbf{s})} \sum_{m=1}^M \beta_m p^m_n s^m_k = y^*_n, \quad \mathcal{I}^*_n(\mathbf{s}) = \mathcal{I}'_n(\mathbf{s}), \quad \forall n \in \mathcal{N}.$$

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 $(s_k^m)^*, \forall 1 \leq k < K^*, \forall m \in \mathcal{M}$ such that:

$$y_{K^*-1}^* = \sum_{m=1}^{M} \beta_m p_n^m \left(s_{K^*-1}^m\right)^*,$$

$$y_{K^*-2}^* = \sum_{m=1}^{M} \beta_m p_n^m \left(\left(s_{K^*-1}^m\right)^* + \left(s_{K^*-2}^m\right)^*\right),$$

$$\vdots$$

$$y_1^* = \sum_{m=1}^{M} \beta_m p_n^m \left(\left(s_{K^*-1}^m\right)^* + \left(s_{K^*-2}^m\right)^* + ... + \left(s_1^m\right)^*\right).$$

Comparing this with Equation (16), where $\sum_{k\in\mathcal{I}_n}\sum_{m=1}^{M} \beta_m p_n^m \left(s_k^m\right)^* = y_n^*$, will give $\mathcal{I}_n^*, 1 \leq n < K^*$ as in (11). Since $\sum_{k\in\mathcal{K}} \left(s_k^m\right)^* = 1, \forall m \in \mathcal{M}$, then $\sum_{k\in\mathcal{K}}\sum_{m=1}^{M} \beta_m p_n^m \left(s_k^m\right)^* = \sum_{m=1}^{M} \beta_m p_n^m$. Thus, $\sum_{m=1}^{M} \beta_m p_n^m \left(s_{K^*}^m\right)^* = \sum_{m=1}^{M} \beta_m p_n^m - y_1^*$ 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 $\sum_{m=1}^{M} \left(s_k^m\right)^*$ can be zero for some $k \in \mathcal{K}$, meaning the edge cache k will receive no traffic from the user groups. 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 broadcast property of wireless channel, this acts as a freshness mechanism to keep the content in other edge-caches from getting obsolete.

In the following we propose an easy to implement caching strategy that satisfies the necessary condition for optimality given in Theorem 1.

Proposition 2: In a system composed of a data set $\mathcal{N}$ of N items with update rates $\boldsymbol{\lambda} = (\lambda_n)_{n=1}^N$ and a set $\mathcal{M}$ of M user groups with popularity distribution $\mathbf{p}^m = (p_n^m)_{n=1}^N, \forall m \in \mathcal{M}$, assume without loss of generality that items are ordered such that $q_1^* \geq q_2^* \geq ... \geq q_N^*$ where q_n^* is defined as $q_n^* = \max\left(0, 1 - \sqrt{\frac{\alpha c_a \lambda_n}{c_f \sum_{m=1}^{M} \beta_m p_n^m}}\right)$. Let $Q = \max\left(n : q_n^* > 0\right) \leq N$, then the following caching strategy where $K^* = Q + 1$ and $\left(s_k^m\right)^* = s_k^*, \forall m \in \mathcal{M}$ optimally solves (4).

$$s_k^* = \begin{cases} q_k^* - q_{k+1}^*, & k \in \{1, \ldots, K^* - 1\}, \\ 1 - q_1^*, & k = K^*, \end{cases} \tag{18}$$

$$\mathcal{I}_n^* = \begin{cases} \{n, \ldots, K^* - 1\}, & n \in \{1, 2, \ldots, K^* - 1\}, \\ \{\}, & n \in \{K^*, \ldots, N\}. \end{cases} \tag{19}$$

Such caching policy attains the minimum achievable caching cost given in (12).

Proof. We just need to show that the proposed caching policy satisfies the optimality condition given in Theorem 1. First, we define q_n^* as:

$$q_n^* = \max\left(0, 1 - \sqrt{\frac{\alpha c_a \lambda_n}{c_f \sum_{m=1}^{M} \beta_m p_n^m}}\right). \tag{20}$$

Note that q_n^* is defined such that $y_n^* = q_n^* \sum_{m=1}^{M} \beta_m p_n^m$. Therefore, $\max\left(n : q_n^* > 0\right) = \max\left(n : y_n^* > 0\right)$ and all we need to show is that the proposed load-splitting strategy given in (18) satisfies the condition given in equation (10). We let each edge cache receive the same fraction of the load from all the user groups, i.e., $\left(s_k^m\right)^* = s_k^*, \forall m \in \mathcal{M}$. Putting this in Equation (10) and dividing both sides of the equation by $\sum_{m=1}^{M} \beta_m p_n^m$ gives the proposed load-splitting of Equation (18) for all the user groups. ■

This reveals a key insight on the optimal caching policy when there are multiple user groups requesting dynamic content from a shared database but each with a different popularity profile. According to Proposition 2, the optimal caching policy in such a case can be achieved by first aggregating the load of all the user groups, and then applying the proposed caching policy as if there is only one user group with demand profile $\left(\sum_{m=1}^{M} \beta_m p_n^m\right)_n$ requesting dynamic content from database with refresh rates $(\lambda_n)_n$. As such, the case of multiple user groups can be simplified to only one user group with aggregated demand profile.

In the following we consider the case of only one user group and omit the upper indices m. Applying the proposed algorithm to some special cases gives us very interesting results.

Proposition 3: 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 $s_1^* = \max\left(0, 1 - \sqrt{\frac{\alpha\, c_a \lambda}{\beta c_f p}}\right)$ 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 (20) all $q_n^*, \forall n \in \mathcal{N}$ will be identical.

$$q_1^* = q_2^* = ... = q_N^* = \max\left(0, 1 - \sqrt{\frac{\alpha\, c_a \lambda}{\beta c_f p}}\right).$$

Now we show how Equation (16) holds $\forall n \in \mathcal{N}$. In case of $q_1^* = 0$, no caching will be employed, i.e., $\mathcal{I}_n^* = \{\varnothing\}, \forall n \in \mathcal{N}$. But if $q_1^* > 0$, then since all q_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 $s_1^* = q_1^*$ 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 3, 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 4: 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:

$$s_k^* = \begin{cases} \sqrt{\frac{\alpha\, c_a \lambda}{\beta c_f p_0}}(\sqrt{(k+1)^z} - \sqrt{k^z}), k \in \{1, \dots, K^* - 2\}, \\ 1 - \sqrt{\frac{\alpha\, c_a \lambda}{\beta c_f p_0}}\sqrt{k^z}, \qquad k = K^* - 1, \\ \sqrt{\frac{\alpha\, c_a \lambda}{\beta c_f p_0}} \qquad k = K^*, \end{cases} \tag{21}$$

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

Proof. For the case of Zipf popularity distribution with constant update rate, $p_1 \geq p_2 \geq ... \geq p_N$, then q_n^* given in (20) can be written as:

$$q_n^* = \max\left(0, 1 - \sqrt{\frac{\alpha\, c_a \lambda}{\beta c_f p_0}}\sqrt{n^z}\right) \leq 1, \quad \forall n \in \mathcal{N}, \tag{22}$$

which results in Q as:

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

which gives $K^* = Q + 1$. Replacing q_n^* in Equation (18) will give the s_k^* as in (21). ■

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 $K^* = \lfloor \sqrt{\frac{\beta c_f p_0}{\alpha\, c_a \lambda}} \rfloor + 1$ and then split the load equally between the edge-caches such that each edge-cache receives the fraction $s_k = \sqrt{\frac{\alpha\, c_a \lambda}{\beta c_f p_0}} \approx \frac{1}{K^*}$ 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 there is only one user group and its 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:

$$C^{\mathcal{S}}(K, \mathbf{r}) = \frac{\beta c_f}{\alpha} + \sum_{n=1}^{N} r_n \left(\frac{c_a \lambda_n}{K - r_n} - \frac{\beta p_n c_f}{\alpha\, K}\right). \tag{23}$$

Proof. Since the number of replica r_n is defined to be $r_n = |\mathcal{I}_n|$, and 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:

$$\min_{\mathbf{r} \in \mathcal{F}_K} C^{\mathcal{S}}(K, \mathbf{r}). \tag{24}$$

Proposition 5: The policy $\mathbf{r}^* = (r_n^*)_n \in \mathcal{F}_K$ that solves (24) is given by:

$$r_n^* = \lfloor K + \frac{1}{2} - \sqrt{\frac{1}{4} + K^2 \frac{\alpha\, c_a \lambda_n}{\beta c_f p_n}}\ \rfloor^+, \quad \forall n \in \mathcal{N}, \tag{25}$$

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:

$$\delta_n^{\mathcal{S}}(l) = \frac{K c_a \lambda_n}{(K-l)(K-l-1)} - \frac{\beta p_n c_f}{\alpha K} \quad \forall n \in \mathcal{N}. \quad (26)$$

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 (26) yields r_n^* as (25). ■

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 6: 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] $q_1^* \geq q_2^* \geq ... \geq q_N^*$ where $q_n^*, \forall n \in \mathcal{N}$ is defined in Equation (20). Let $Q = \max(n : q_n^* > 0) \leq N$ which is independent of the number of edge-caches K, then we have:

$$C^{\mathcal{S}}(K, \mathbf{r}^*) - C^* \leq \left(2\frac{\beta c_f}{\alpha} \sqrt{\frac{\beta c_f}{\alpha\, c_a}} \sum_{n=1}^{Q} \sqrt{\frac{p_n^3}{\lambda_n}} \right) \frac{1}{K^2}, \quad (27)$$

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:

$$B^{\mathcal{S}}(K, \mathbf{r}^*) - B^*(\lambda, p) \geq \sum_{n=1}^{Q} \lfloor K + \frac{1}{2} - \sqrt{\frac{1}{4} + K^2 \frac{\alpha\, c_a \lambda_n}{\beta c_f p_n}} \rfloor^+ - \frac{1}{2} Q(Q+1), \quad (28)$$

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).

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

Proof. To prove Equation (27), we use the following Taylor approximation.

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

where $q_n^* = \max(0, 1 - \sqrt{\frac{\alpha\, c_a \lambda_n}{\beta c_f p_n}})$. In the case when $q_n^* = 0$, we have $\frac{\alpha\, c_a \lambda_n}{\beta c_f p_n} \geq 0$ and according to Equation (25), $r_n^* = 0$, which gives $|\frac{r_n^*}{K} - q_n^*| = 0$. In the case when $q_n^* > 0$, we can show that $|K q_n^* - r_n^*| < 1$ which gives $|\frac{r_n^*}{K} - q_n^*| < \frac{1}{K}$. Since, by hypothesis, $q_1^* \geq q_2^* \geq ... \geq q_N^*$, we can write:

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

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 (25), and as we showed that in the case of $q_n^* = 0$, 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 (28). ■

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, $\alpha = .9$ for the channel reliability, 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 and there is only one user group requesting items from the database. We use Zipf distribution to capture the popularity of items, i.e., users send their request to the database according to a Zipf distribution with parameter z.

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:

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

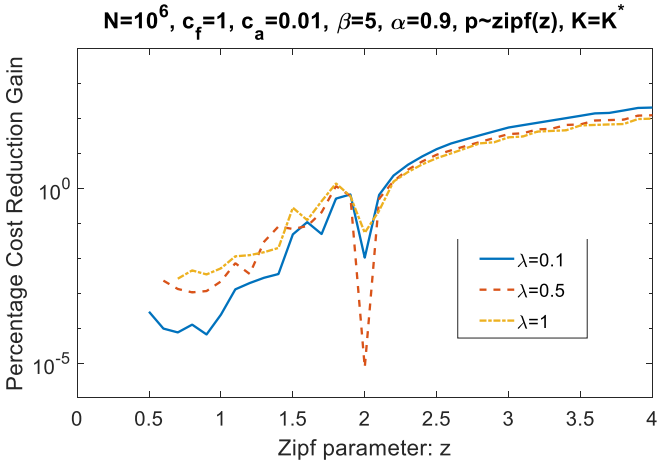

Fig. 3: Percentage cost gain of the optimal caching policy

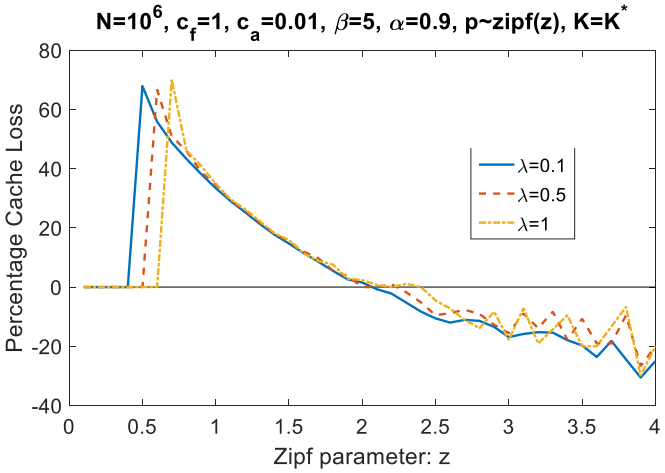

Fig. 4: Percentage cache loss of the optimal caching policy

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

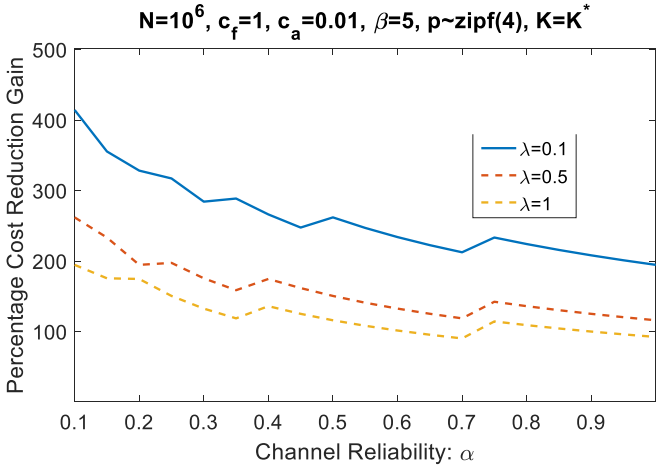

Fig. 5: Percentage cost gain of the optimal caching policy

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

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 6, 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.

Fig. 5 shows the percentage cost gain as the function of the channel reliability α for the Zipf(4) popularity distribution. The figure shows that the gain increases as the wireless channel becomes less reliable. In other words, as the reliability of the wireless channel decreases, the cost optimal caching policy which splits the load unequally between

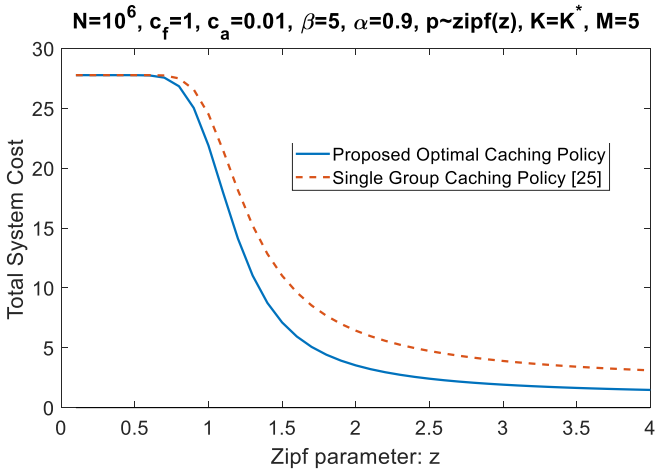

Fig. 6: Total cost of a system with $M = 5$ user groups

the edge caches, suffers less compared to the equal load-splitting policy. This is yet another reason to show the superiority of the optimal unequal load splitting policy for more predictable item demand.

Finally, Fig. 6 shows the total system cost when applying the single group caching strategy separately to each user group [41], and then adding the costs versus applying the proposed optimal caching strategy given in Proposition 2 to the system. Here we assume there are $M = 5$ user groups requesting items from a shared database according to Zipf distribution with parameter z. First, we treat each user group independently and apply the single group caching strategy separately to them and then add the costs to obtain the total system cost. Then we consider the same system consisting of five user groups and apply the proposed optimal caching policy given in Proposition 2 to obtain the total system cost. As we can see in Fig. 6, the proposed optimal caching policy outperforms the single group caching policy applied separately to user groups. According to the figure, under uniform popularity profile, both caching strategies obtain the same system cost. Moreover, as the demand profile becomes more predictable, the total system cost under both of these caching policies decreases. However, the proposed caching policy that achieves the optimal cost outperforms the single group caching policy. The gain is shown only for $M = 5$ user groups and as the number of user groups increases, the gain will also increase. This shows that in a system including multiple user groups, it is better to make the caching decision based on the whole system rather than making the caching decisions for each user group separately.

Next, we compare the cost reduction gains of the proposed policy compared to a well-known caching policy, namely cache-the-most-popular strategy. The objective of the cache-the-most-popular strategy is to try to minimize the cost by caching only the items that have the highest probability of being requested by the users. Such a policy is defined in [13] and [45] as following.

Definition 1 (Cache-The-Most-Popular): Define the $\mathcal{I}_B^p \subseteq \mathcal{N}$ to be the set of $B \leq N$ most popular items. Then cache-the-most-popular strategy for the K edge caches and total cache capacity B, is given by:

$$r_n^p := \begin{cases} 1, & n \in \mathcal{I}_B^p, \\ 0, & n \in \mathcal{N} \setminus \mathcal{I}_B^p, \end{cases}$$

with $\mathbf{r}^p := (r_1^p, ..., r_N^p)$ being the vector of replication and $C^{\mathcal{P}}(K, B)$ is the cost under the cache-the-most-popular strategy when there are K edge caches and the total cache capacity is B.

Such strategy does not consider the freshness of items, yet by caching only the most popular items, it focuses on minimizing the number of cache misses.

We compare the average cost achieved by the proposed caching policy and the average cost of the cache-the-most-popular strategy under the number of edge-caches $K = K^*$ and the total cache capacity $B = B^*$ defined in Theorem 1. We assume the same system variables declared above and adopt the percentage cost gain of the optimal caching to the cache-the-most-popular strategy's cost as our performance metric. Such a metric is defined as:

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

The percentage cost gain is depicted in Fig. 7 as the function of the Zipf parameter z. 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 cache-the-most-popular strategy, which focuses on minimizing the cache misses, performs almost as well as the optimal caching. Moreover, the difference vanishes as the refresh rate increases and items become more dynamic. But as the Zipf parameter increases and the certainty about the demand increases, the cost reduction gain increases. The figure reveals that if the demand profile is predictable enough, i.e., $z > 1.5$, the gain is almost 100 percent. In other words, the proposed policy almost achieves half of the cost incurred from cache-the-most-popular strategy.

Fig. 8 shows the percentage cost gain as the function of the channel reliability α for the Zipf(2) popularity distribution. The figure shows that the gain increases as the wireless channel becomes less reliable. In other words, as the reliability of the wireless channel decreases, the proposed caching policy, which splits the load unequally between the edge caches, suffers less compared to the cache-the-most-popular strategy. This is yet another reason to show the superiority of the proposed policy for more predictable item demand in the presence of wireless channel imperfections.

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

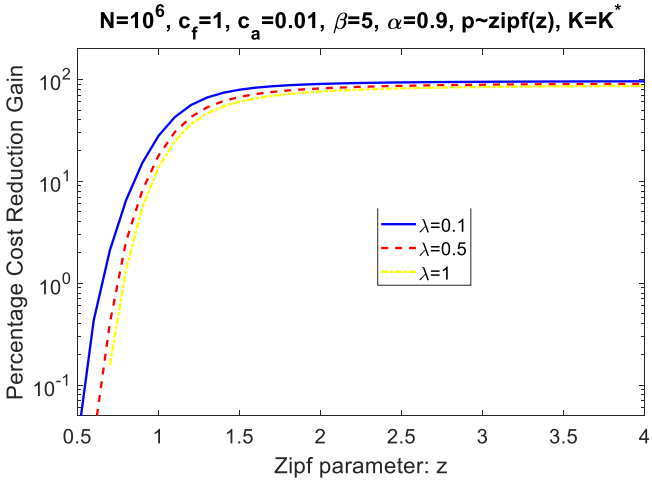

Fig. 7: Percentage cost gain of the proposed caching policy compared to the cache-the-most-popular strategy.

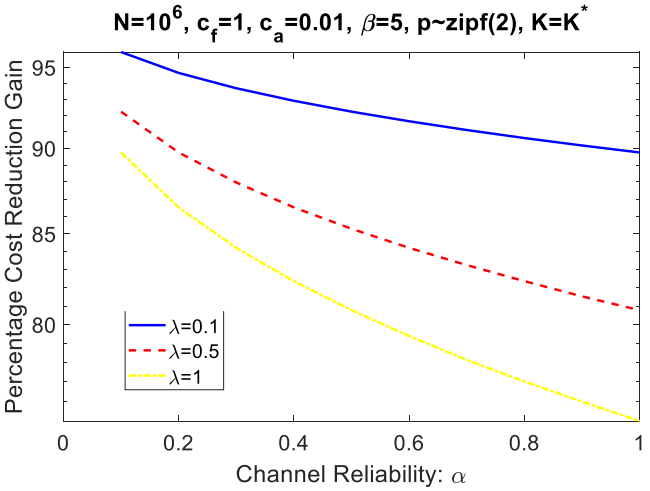

Fig. 8: Percentage cost gain of the proposed caching policy compared to the cache-the-most-popular strategy.

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] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5g systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[2] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[3] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas, "Distributed cache management in information-centric networks," *IEEE Transactions on Network and Service Management*, vol. 10, no. 3, pp. 286–299, 2013.

[4] W. Meizhen, S. Yanlei, and T. Yue, "The design and implementation of lru-based web cache," in *2013 8th International Conference on Communications and Networking in China (CHINACOM)*. IEEE, 2013, pp. 400–404.

[5] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1111–1125, 2018.

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

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

[8] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[9] 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.

[10] Z. Zheng and Z. Zheng, "Towards an improved heuristic genetic algorithm for static content delivery in cloud storage," *Computers & Electrical Engineering*, vol. 69, pp. 422–434, 2018.

[11] M. Rabinovich and O. Spatscheck, *Web caching and replication*. Addison-Wesley Boston, USA, 2002, vol. 67.

[12] T. Tang, "Multicast enabled caching service," Oct. 10 2002, uS Patent App. 09/934,013.

[13] 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.

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

[15] ——, "Fresh caching of dynamic content over the wireless edge," *IEEE/ACM Transactions on Networking*, 2022.

[16] 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.

[17] K. S. Candan, W.-S. Li, Q. Luo, W.-P. Hsiung, and D. Agrawal, "Enabling dynamic content caching for database-driven web sites," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001, pp. 532–543.

[18] W. Shi, R. Wright, E. Collins, and V. Karamcheti, "Workload characterization of a personalized web site and its implications for dynamic content caching," in *Proceedings of the Seventh International Workshop on Web Caching and Content Distribution (WCW'02)*. Citeseer, 2002, pp. 1–16.

[19] W.-S. Li, O. Po, W.-P. Hsiung, K. S. Candan, and D. Agrawal, "Freshness-driven adaptive caching for dynamic content web sites," *Data & Knowledge Engineering*, vol. 47, no. 2, pp. 269–296, 2003.

[20] P. Wu, J. Li, L. Shi, M. Ding, K. Cai, and F. Yang, "Dynamic content update for wireless edge caching via deep reinforcement learning," *IEEE Communications Letters*, vol. 23, no. 10, pp. 1773–1777, 2019.

[21] 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.

[22] X. Xu, C. Feng, S. Shan, T. Zhang, and J. Loo, "Proactive edge caching in content-centric networks with massive dynamic content requests," *IEEE Access*, vol. 8, pp. 59 906–59 921, 2020.

[23] J. Gao, S. Zhang, L. Zhao, and X. Shen, "The design of dynamic probabilistic caching with time-varying content popularity," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1672–1684, 2020.

[24] S. Mehrizi, A. Tsakmalis, S. ShahbazPanahi, S. Chatzinotas, and B. Ottersten, "Popularity tracking for proactive content caching with dynamic factor analysis," in *2019 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2019, pp. 875–880.

[25] S. Kumar and R. Tiwari, "Optimized content centric networking for future internet: dynamic popularity window based caching scheme," *Computer Networks*, vol. 179, p. 107434, 2020.

[26] 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.

[27] X. Zhang, G. Zheng, S. Lambotharan, M. R. Nakhai, and K.-K. Wong, "A reinforcement learning-based user-assisted caching strategy for dynamic content library in small cell networks," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3627–3639, 2020.

[28] A. Masood, D. S. Lakew, and S. Cho, "Learning based content caching for wireless networks," in *2020 International Conference on Information Networking (ICOIN)*. IEEE, 2020, pp. 74–75.

[29] S. M. Azimi, O. Simeone, A. Sengupta, and R. Tandon, "Online edge caching and wireless delivery in fog-aided networks with dynamic content popularity," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1189–1202, 2018.

[30] X. Zhang, G. Zheng, S. Lambotharan, M. R. Nakhai, and K.-K. Wong, "A learning approach to edge caching with dynamic content library in wireless networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[31] M. S. H. Abad, E. Ozfatura, O. Ercetin, and D. Gündüz, "Dynamic content updates in heterogeneous wireless networks," in *2019 15th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*. IEEE, 2019, pp. 107–110.

[32] L. Huang and E. Modiano, "Optimizing age-of-information in a multi-class queueing system," in *2015 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2015, pp. 1681–1685.

[33] C. Kam, S. Kompella, and A. Ephremides, "Age of information under random updates," in *2013 IEEE International Symposium on Information Theory*. IEEE, 2013, pp. 66–70.

[34] A. Maatouk, M. Assaad, and A. Ephremides, "On the age of information in a csma environment," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 818–831, 2020.

[35] X. Chen, C. Wu, T. Chen, H. Zhang, Z. Liu, Y. Zhang, and M. Bennis, "Age of information aware radio resource management in vehicular networks: A proactive deep reinforcement learning perspective," *IEEE Transactions on wireless communications*, vol. 19, no. 4, pp. 2268–2281, 2020.

[36] M. Bastopcu and S. Ulukus, "Age of information for updates with distortion," in *2019 IEEE Information Theory Workshop (ITW)*. IEEE, 2019, pp. 1–5.

[37] H. H. Yang, A. Arafa, T. Q. Quek, and H. V. Poor, "Age of information in random access networks: A spatiotemporal study," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[38] S. Kompella and C. Kam, "Special issue on age of information," *Journal of Communications and Networks*, vol. 21, no. 3, pp. 201–203, 2019.

[39] ——, "Age of information: Control and estimation," *NAVAL RESEARCH LAB WASHINGTON DC WASHINGTON United States*, 2020.

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

[41] B. Abolhassani, J. Tadrous, and A. Eryilmaz, "Optimal load-splitting and distributed-caching for dynamic content," in *IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2021.

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

[43] ——, "Delay gain analysis of wireless multicasting for content distribution," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 529–542, 2020.

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

[45] B. Abolhassani, J. Tadrous, and A. Eryilmaz, "Single vs distributed edge caching for dynamic content," *IEEE/ACM Transactions on Networking*, vol. 30, no. 2, pp. 669–682, 2021.

Bahman Abolhassani received the B.Sc. and M.Sc. degrees in electrical engineering from Sharif University of Technology (SUT), Tehran, Iran, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, The Ohio State University, Columbus, OH, USA. Between 2015 and 2017, he was a researcher at the Optical Networks Research Laboratory, SUT. His research interests include communication networks, optimization theory, caching and algorithm design.

John Tadrous is an associate professor of electrical and computer engineering at Gonzaga University. He received his Ph.D. degree in electrical engineering from the ECE Department at The Ohio State University in 2014, MSc degree in wireless communications from the Center of Information Technology at Nile University in 2010, and BSc degree from the EE Department at Cairo University in 2008. Between 2016 and 2021 he served as an assistant professor of electrical and computer engineering at Gonzaga University. From May 2014 to August 2016, he was a post-doctoral research associate with the ECE Department at Rice University. In 2020, Dr. Tadrous was elevated to a Senior Member of the IEEE. In addition, he received the Gonzaga University's Faculty Award for Professional Contributions. His research interests include modeling and analysis of human behavior's impact on data networks in various timescales from seconds to hours, and how to harness that behavior for improved network resource management. Dr. Tadrous' served a technical program committee member for several conferences such as Mobihoc, COMSNETS, and WiOpt.

Atilla Eryilmaz (S'00 / M'06 / SM'17) received his M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 2001 and 2005, respectively. Between 2005 and 2007, he worked as a Postdoctoral Associate at the Laboratory for Information and Decision Systems at the Massachusetts Institute of Technology. Since 2007, he has been at The Ohio State University, where he is currently a Professor and the Graduate Studies Chair of the Electrical and Computer Engineering Department. Dr. Eryilmaz's research interests span optimal control of stochastic networks, machine learning, optimization, and information theory. He received the NSF-CAREER Award in 2010 and two Lumley Research Awards for Research Excellence in 2010 and 2015. He is a co-author of the 2012 IEEE WiOpt Conference Best Student Paper, subsequently received the 2016 IEEE Infocom, 2017 IEEE WiOpt, 2018 IEEE WiOpt, and 2019 IEEE Infocom Best Paper Awards. He has served as: a TPC co-chair of IEEE WiOpt in 2014, ACM Mobihoc in 2017, and IEEE Infocom in 2022; an Associate Editor (AE) of IEEE/ACM Transactions on Networking between 2015 and 2019; an AE of IEEE Transactions on Network Science and Engineering between 2017-2022; and is currently an AE of the IEEE Transactions on Information Theory since 2022.