Resolving Cache-Load Imbalance Bottleneck of Stochastic Shared-Cache Networks

Adeel Malik, Berksan Serbetci, Petros Elia

Resolving Cache-Load Imbalance Bottleneck of Stochastic Shared-Cache Networks

Adeel Malik, Berksan Serbetci, Petros Elia
Dept. of Communication Systems, EURECOM, Sophia Antipolis, 06410, France
{malik, serbetci, elia}@eurecom.fr

***Abstract*—This work proposes a two-layered coded caching scheme to resolve the cache-load imbalance bottleneck of the coded caching in a stochastic shared-cache network where the association between users and shared caches is random, i.e., for the scenario where each user can appear within the coverage area of – and subsequently is assisted by – a specific cache-enabled helper node based on a uniform probability distribution. To insightfully capture the effectiveness of our scheme in mitigating the adverse effect of randomness in shared-cache networks, we derive the exact scaling laws of the average delivery time. In the scenario of an error-free broadcast channel of bounded capacity per unit of time where the delivery involves K users and Λ cache-enabled helper nodes, we show that empowering users with an additional layer of caching can significantly mitigate, and in certain memory regimes completely nullify the adverse effects of the cache-load imbalance bottleneck.**

***Index Terms*—Coded caching, shared caches, heterogeneous networks, femtocaching.**

I. Introduction

The persistent and exponential growth of mobile data traffic has highlighted the need for the solutions that can complement the serving capacities of limited network bandwidth resources. In this context, cache-enabled wireless networks have emerged as one of the most viable solutions that can transform the storage capabilities of the network nodes into a new and powerful resource. The main idea of cache-enabled wireless networks is to bring content objects closer to the users by proactively storing the content during the off-peak hours at the network edge [1], including at wireless communication stations as well as at end-user devices such that upon the user's request the content is provided locally by neighboring cache-enabled network edge, without or with minimally utilizing the backhaul communication.

Within the framework of cache-enabled wireless networks, an innovative concept of coded caching introduced in [1] has revealed that an unbounded number of cache-aided users can be served with a bounded amount of network resources. The key to this caching approach is a carefully designed cache placement algorithm that enables transmitting independent content to multiple users at a time. This delivery speedup due to multicasting transmissions is referred to as the *coding gain*, and it scales with the total storage capacity of the network. Since then, many extensions of the basic coded caching setting have been studied. This includes the study of coded caching in heterogeneous networks (HetNets) [2], [3], in D2D networks [4], in settings with arbitrary popularity distributions [5], [6], and other settings as well [7]–[13].

The work is supported by the European Research Council under the EU Horizon 2020 research and innovation program / ERC grant agreement no. 725929 (project DUALITY).

A. Cache-Load Imbalance Bottleneck of Shared-Cache Networks

Despite the fact that there exist several studies that prove the massive gains coded caching offers in a wide range of network settings, there is still need of further exploration of its applicability in realistic settings. Among them, the shared-cache setting in which many users have access to the same cache content is of great importance. Such setting can be found in the context of cache-enabled HetNets, where a macrocell (base station) covering a larger area delivers the content to a set of users with the help of cache-enabled femtocells (helper nodes or small base stations) each covering smaller cells. In this scenario, any user that appears in a particular femtocell, can benefit from the cache-content of the helper node covering that femtocell.

The first study of coded caching in a shared-cache setting with uniform user-to-cache association – where each helper node serves an equal number of users – can be found in [2]. The work in [3] generalized this setting to the case where an arbitrary number of users are assisted by each helper node, and characterized the optimal worst-case delivery time for the case when the cache placement is uncoded, and when the cache placement is agnostic to the user-to-cache association. This work proved that, for a setting where a base station serves K users with the help of Λ cache-enabled helper nodes each equipped with a normalized storage capacity of γ, uniform user-to-cache association leads to the minimum worst-case delivery time of $\frac{K(1-\gamma)}{1+\Lambda\gamma}$, consequently achieving a coding gain of $1+\Lambda\gamma$. However, in practice, it is possible that user-to-cache association can be non-uniform.

Recently, the work in [9] studied the cache-load imbalance bottleneck of coded caching in the shared-cache setting by extending the deterministic user-to-cache association setting in [3] to the stochastic network setting, where the cache populations (i.e., number of users associated to a cache) follow a given probability distribution. This work revealed that the cache-load imbalance due to randomness in user-to-cache association can lead to a significant deterioration in

coding gains, especially when cache populations probability distribution is skewed. The very same work also shows that when the cache populations follow a uniform probability distribution, multiplicative deterioration in coding gain due to randomness — as compared to the well-known deterministic uniform user-to-cache association case — can in fact be unbounded, and scales as $\Theta\left(\frac{\log\Lambda}{\log\log\Lambda}\right)$ when $K=\Theta(\Lambda)$, and that this scaling vanishes when $K=\Omega(\Lambda\log\Lambda)$. Moreover, the same work has also revealed that unlike the case of uniform cache population probability distribution – where the deterioration can be avoided as long as $K=\Omega(\Lambda\log\Lambda)$ – the existence of skewness in cache population densities leads to an unbounded deterioration irrespective of the relation between K and Λ. Therefore, it is crucial to look for the solutions that can resolve the cache-load imbalance bottleneck in shared-cache settings. There are a few works that aim to alleviate this adverse effect of cache-load imbalance bottleneck where one approach focuses on *uniformizing* the cache loads by applying load balancing techniques [9], and another one that offers a novel scheme to carefully tune the cache sizes by using optimization tools [14].

In this work, we aim to mitigate the adverse effect of cache-load imbalance bottleneck by adding an additional layer of cache in the shared-cache setting. The main idea is that in addition to the cache-enabled helper nodes, each user is also equipped with its own cache. This will enable the overpopulated femtocells to have higher collective storage capacity compared to the less populated femtocells, and eventually alleviate the detrimental performance deterioration due to randomness.

B. Notations

Throughout the paper, we use the following asymptotic notation: i) $f(x)=O(g(x))$ means that there exist constants a and c such that $f(x)\leq ag(x),\forall x>c$, ii) $f(x)=o(g(x))$ means that $\lim_{x\to\infty}\frac{f(x)}{g(x)}=0$, iii) $f(x)=\Omega(g(x))$ if $g(x)=O(f(x))$, iv) $f(x)=\omega(g(x))$ means that $\lim_{x\to\infty}\frac{g(x)}{f(x)}=0$, v) $f(x)=\Theta(g(x))$ if $f(x)=O(g(x))$ and $f(x)=\Omega(g(x))$. We use the term $\mathrm{polylog}(x)$ to denote the class of functions $\bigcup_{k\geq 1}O((\log x)^k)$ that are polynomial in $\log x$. Unless otherwise stated, logarithms are assumed to have base 2.

II. Network Setting & Problem Statement

We consider a shared-cache network setting which consists of a base station (BS) with a content library of N unit-sized files $\mathcal{F}=[F_1,F_2,\ldots,F_N]$, Λ cache-enabled helper nodes $\mathcal{H}=[1,2,\ldots,\Lambda]$, and K cache-enabled users $\mathcal{U}=[1,2,\ldots,K]$. Each helper node $\lambda\in\mathcal{H}$ is equipped with a normalized storage capacity of $\gamma\triangleq\frac{M_h}{N}\in\left[\frac{1}{\Lambda},\frac{2}{\Lambda},\ldots,1\right]$ (i.e., can store the content equal to the size of M_h files) and each user $k\in\mathcal{U}$ is equipped with a normalized storage capacity of $\gamma_u\triangleq\frac{M_u}{N}\in\left[\frac{1}{\Lambda},\frac{2}{\Lambda},\ldots,1\right]$ (i.e., can store the content equal to the size of M_u files). The BS delivers content via an error-free broadcast link of bounded capacity per unit of time to K users, with the assistance of helper nodes. We assume that in addition to its own cache, each user within the coverage area of any helper node $\lambda\in\mathcal{H}$ can download the content stored in that helper node's cache at zero cost. In this setting, the storage regime of $\gamma_u+\gamma\geq 1$ is trivial as in this case any user could retrieve the complete content that it requests at zero cost. Therefore, in this work, we are only interested in the storage regime of $\gamma_u+\gamma<1$. Henceforth, each helper node will be referred to a cell.

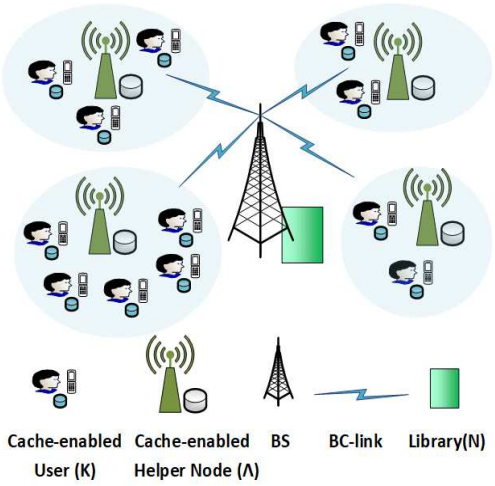

Fig. 1: An instance of a cache-aided heterogeneous network.

The communication process consists of three phases; the *content placement phase*, the *user-to-cell association phase*, and the *content delivery phase*. The first phase involves the placement of the content in the users' and cells' caches. We assume that this phase is oblivious to the outcome of the next two phases. The second phase is the association phase where each user appears within the coverage area of any particular cell $\lambda\in\mathcal{H}$ from which it can download content at zero cost. We assume that this phase is also oblivious to the other two phases. The final phase involves the process of the BS delivering the content to K users, where users simultaneously request files from the library. This phase is aware of the outcome of the first two phases.

Content Placement: We consider a library partition parameter α, where $0\leq\alpha\leq 1$, which divides each file $F_i\in\mathcal{F}$ into two parts F_i^1 and F_i^2, such that $F_i=[F_i^1,F_i^2]$, where $\left|F_i^1\right|=\alpha$ unit-sized, and $\left|F_i^2\right|=(1-\alpha)$ unit-sized. We adopt the uncoded content placement scheme of [1], where the first part of the files $\mathcal{F}^1=[F_1^1,F_2^1,\ldots,F_N^1]$ to be stored in the cells' caches, and the second part of the files $\mathcal{F}^2=[F_1^2,F_2^2,\ldots,F_N^2]$ to be cached in the users' caches.

User-to-cell association: We consider a uniformly random user-to-cell association process, where each user can appear in the coverage area of any particular cell with equal probability. For any given instance, we observe a *cell population vector* $\mathbf{V}=[v_1,v_2,\ldots,v_\Lambda]$, where v_λ denotes the number of users that are within the coverage area of cell $\lambda\in\mathcal{H}$. In addition, we denote the profile vector $\mathbf{L}=[l_1,l_2,\ldots,l_\Lambda]=sort(\mathbf{V})$ as the sorted version of $\mathbf{V}$, where $sort(\mathbf{V})$ denotes the sorting of vector $\mathbf{V}$ in descending order.

Figure 1 depicts an instance of cache-aided heterogeneous network setting for $\mathbf{L} = [5, 4, 3, 2]$.

Content Delivery: The delivery phase begins when the BS receives the requests of all users, where each user $k \in \mathcal{U}$ requests a single file F_{d_k} that is indexed by $d_k \in [1, 2, \ldots, N]$ from the content library $\mathcal{F}$. We denote $\mathbf{d} = [d_1, d_2, \ldots, d_K]$ as the *request vector* of K users. We assume that each user requests a different file, which is a common assumption in coded caching works [1], [3] as it leads to the worst-case delivery time. Once the BS is aware of users' request vector $\mathbf{d}$, it commences delivery over an error-free broadcast link of bounded capacity of one unit-sized file per time slot.

A. Problem Formulation

The stochastic nature of the user-to-cell association leads to randomness in cell population vector $\mathbf{V}$. Thus, our measure of interest is the average delay given by

$$\overline{T}(\gamma, \gamma_u) = \min_{\alpha} E_{\mathbf{V}}[T(\mathbf{V}, \alpha)] = \min_{\alpha} \sum_{\mathbf{V}} P(\mathbf{V}) T(\mathbf{V}, \alpha), \quad (1)$$

where $T(\mathbf{V}, \alpha)$ is the worst-case delivery time needed to complete the delivery of any request vector $\mathbf{d}$ corresponding to a specific cell population vector $\mathbf{V}$, and $P(\mathbf{V})$ is the probability of observing the cell population vector $\mathbf{V}$. The BS delivers the content in two phases. In the first phase, the BS adopts the delivery scheme proposed in [3], where each user $k \in \mathcal{U}$ retrieves the first part of its request $F^1_{d_k}$ using the content received from the BS and the content stored in the cell's cache that it is associated with. We know from [3] that for any $\mathbf{V}$ such that $sort(\mathbf{V}) = \mathbf{L}$, the BS needs to transmit the data equivalent to a total of

$$T_h(\mathbf{L}, \alpha) = \sum_{\lambda=1}^{\Lambda - t} l_\lambda \frac{\binom{\Lambda - \lambda}{t}}{\binom{\Lambda}{t}} \quad (2)$$

partitioned files, where $t = \frac{\Lambda \gamma}{\alpha}$, and the size of each partitioned file is $\left|F^1_i\right| = \alpha$. Then, in the second phase, the BS adopts the delivery scheme proposed in [1], where each user $k \in \mathcal{U}$ retrieves the second part of its request $F^2_{d_k}$ using the content received from the BS and the content stored in its own cache. This phase of delivery is independent of the user-to-cell association. We know from [1] that in this phase, the BS has to transmit the data equivalent to a total of

$$T_u(K, \alpha) = \frac{K(1 - \frac{\gamma_u}{1-\alpha})}{1 + \frac{K\gamma_u}{1-\alpha}} \quad (3)$$

partitioned files, where the size of each partitioned file is $\left|F^2_i\right| = (1 - \alpha)$. Consequently, for a fixed partition parameter α, the worst-case delivery time for any $\mathbf{V}$ such that $sort(\mathbf{V}) = \mathbf{L}$ is given as

$$T(\mathbf{L}, \alpha) = \alpha T_h(\mathbf{L}, \alpha) + (1 - \alpha) T_u(K, \alpha), \quad (4)$$

and the average delay takes the form of

$$\begin{aligned} \overline{T}(\gamma, \gamma_u, \alpha) &= \alpha E_{\mathbf{L}}[T_h(\mathbf{L}, \alpha)] + (1 - \alpha) E_{\mathbf{L}}[T_u(K, \alpha)] \\ &= \alpha \sum_{\mathbf{L} \in \mathcal{L}} P(\mathbf{L}) \sum_{\lambda=1}^{\Lambda - t} l_\lambda \frac{\binom{\Lambda - \lambda}{t}}{\binom{\Lambda}{t}} + (1 - \alpha) \frac{K(1 - \frac{\gamma_u}{1-\alpha})}{1 + \frac{K\gamma_u}{1-\alpha}}, \quad (5) \end{aligned}$$

where $\mathcal{L}$ is the set of all possible profile vectors $\mathbf{L}$ and

$$P(\mathbf{L}) \triangleq \sum_{\mathbf{V}: sort(\mathbf{V}) = \mathbf{L}} P(\mathbf{V}), \quad (6)$$

is simply the cumulative probability over all $\mathbf{V}$ for which $sort(\mathbf{V}) = \mathbf{L}$. Then, our metric of interest takes the form of

$$\overline{T}(\gamma, \gamma_u) = \min_{\alpha} \overline{T}(\gamma, \gamma_u, \alpha). \quad (7)$$

In this work, we focus on the asymptotic analysis of the performance of the proposed two-layered shared-cache setting. In the next section, we provide the scaling laws of the performance as well as the order-optimal partition parameter α which leads to the minimum average delay in a simple and insightful form.

III. Main Results

In this section we present our main results on the performance of our proposed two-layered shared-cache setting. Our first result provides the scaling law of the average delivery time $\overline{T}(\gamma, \gamma_u, \alpha)$ for any given partition parameter α, in the limit of large Λ.

Theorem 1. *In a Λ-cell, K-user shared-cache setting with each user equipped with a normalized storage capacity γ_u and each cell equipped with a normalized storage capacity of γ, the average delay for any given partition parameter α and a random user-to-cell association scales as*

$$\begin{aligned} \overline{T}(\gamma, \gamma_u, \alpha) = \Theta \Bigg(&\frac{\alpha^2 (c\gamma_u + \gamma)}{\gamma \gamma_u} + \\ &\frac{\alpha(-c\gamma\gamma_u + \gamma_u\gamma - 2\gamma) + \gamma - \gamma\gamma_u}{\gamma\gamma_u} \Bigg). \quad (8) \end{aligned}$$

where

$$c = \begin{cases} \frac{\log \Lambda}{log\left(\frac{\Lambda}{K} \log \Lambda\right)} & \text{if } K \in \left[\frac{\Lambda}{\text{polylog}(\Lambda)}, o\left(\Lambda \log \Lambda\right)\right] \\ \frac{K}{\Lambda} & \text{if } K = \Omega\left(\Lambda \log \Lambda\right). \end{cases} \quad (9)$$

Proof. The proof can be found in Appendix A. □

Now, we proceed with the following lemma that provides the order-optimal $\hat{\alpha}$ which leads to minimum average delay $\overline{T}(\gamma, \gamma_u, \alpha)$.

Lemma 1. *The order-optimal partition parameter $\hat{\alpha}$ that minimizes the average delay $\overline{T}(\gamma, \gamma_u, \alpha)$ is given as*

$$\hat{\alpha} = \begin{cases} \gamma & \text{if } \gamma_u \geq \frac{2 - 2\gamma}{1 + c} \\ \frac{c\gamma\gamma_u - \gamma_u\gamma + 2\gamma}{2(c\gamma_u + \gamma)} & \text{otherwise,} \end{cases} \quad (10)$$

where

$$c = \begin{cases} \frac{\log \Lambda}{log\left(\frac{\Lambda}{K} \log \Lambda\right)} & \text{if } K \in \left[\frac{\Lambda}{\text{polylog}(\Lambda)}, o\left(\Lambda \log \Lambda\right)\right] \\ \frac{K}{\Lambda} & \text{if } K = \Omega\left(\Lambda \log \Lambda\right). \end{cases} \quad (11)$$

Proof. The proof can be found in Appendix B. □

With Lemma 1 at hand, we are ready to to present our final result which is the scaling law of our performance metric.

Theorem 2. *In a Λ-cell, K-user shared-cache setting with each user equipped with a normalized storage capacity γ_u and each cell equipped with a normalized storage capacity of*

γ, the minimum average delay corresponding to the order-optimal partition parameter $\hat{\alpha}$ and a random user-to-cell association scales as

$$\overline{T}(\gamma,\gamma_u)=\begin{cases}\Theta\left((1-\gamma)\frac{1-\gamma_u-\gamma}{\gamma_u}\right) & \text{if } \gamma_u \geq \frac{2-2\gamma}{1+c}\\ \Theta\left(\frac{c(1-\gamma-\gamma_u)}{c\gamma_u+\gamma}-\frac{\gamma\gamma_u(c-1)^2}{4c\gamma_u+4\gamma}\right) & \text{otherwise}\end{cases} \tag{12}$$

where

$$c=\begin{cases}\frac{\log\Lambda}{log\left(\frac{\Lambda}{K}\log\Lambda\right)} & \text{if } K\in\left[\frac{\Lambda}{\text{polylog}(\Lambda)},o\left(\Lambda\log\Lambda\right)\right]\\ \frac{K}{\Lambda} & \text{if } K=\Omega\left(\Lambda\log\Lambda\right).\end{cases} \tag{13}$$

Proof. The proof can be found in Appendix C. □

Remark 1. *For the case of $\gamma_u = 0$, our proposed two-layered shared-cache setting is exactly equal to the original stochastic shared-cache setting studied in [9], and the scaling of the average delay in Theorem 2 is exactly equal to the one presented in [9, Theorem 3].*

Theorem 2 provides different memory regimes, and it hints out that adding an additional layer of cache in the shared-cache setting can be useful to resolve the cache-load imbalance bottleneck. In order to better understand the impact of the additional layer, we proceed to present the following lemma which characterizes the performance of our proposed shared-cache setting for the well-known deterministic uniform user-to-cache association, which leads to the minimum worst-case delivery time [9].

Lemma 2. *In a Λ-cell, K-user shared-cache setting with each user equipped with a normalized storage capacity of γ_u and each cell equipped with a normalized storage capacity of γ, the minimum worst-case delivery time corresponding to a uniform user-to-cell association evaluated at corresponding order-optimal partition parameter $\hat{\alpha}$ scales as*

$$T_{min}(\gamma,\gamma_u)=\begin{cases}\Theta\left((1-\gamma)\frac{1-\gamma_u-\gamma}{\gamma_u}\right) & \text{if } \gamma_u \geq \frac{2-2\gamma}{1+c}\\ \Theta\left(\frac{c(1-\gamma-\gamma_u)}{c\gamma_u+\gamma}-\frac{\gamma\gamma_u(c-1)^2}{4c\gamma_u+4\gamma}\right) & \text{otherwise,}\end{cases} \tag{14}$$

where $c=\frac{K}{\Lambda}$.

Proof. The proof can be found in Appendix D. □

In identifying the exact scaling laws of the problem, Theorem 2 nicely captures the following points.

- It identifies the system parameter regime $K = \Omega\left(\Lambda\log\Lambda\right)$ which is consistent with the observation of [9], where there is no performance deterioration due to randomness as the multiplicative gap between the average delivery time and the minimum worst-case delivery time is equal to one i.e., $\frac{\overline{T}(\gamma,\gamma_u)}{T_{min}(\gamma,\gamma_u)}=1$.
- For the system parameter regimes where the performance deterioration due to the cache-load imbalance is unbounded, it identifies a memory regime of $\gamma_u \geq \frac{2-2\gamma}{1+\frac{\log\Lambda}{log\left(\frac{\Lambda}{K}\log\Lambda\right)}}$ for the additional cache layer that can completely nullify the impact of the cache-load imbalance such that multiplicative gap between the average delivery time and the minimum worst-case delivery time is equal to one.

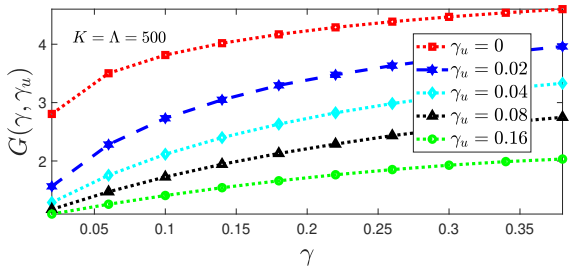

Fig. 2: Multiplicative gap $G(\gamma,\gamma_u)$ between $\overline{T}(\gamma,\gamma_u)$ from (16) and $T_{min}(\gamma,\gamma_u)$ from (17).

IV. Numerical Validation

We know from (5) that it is computationally expensive to numerically evaluate the exact average delay even for small system parameters. Therefore, we proceed to numerically evaluate the effectiveness of our proposed approach in mitigating the effect of the cache-load imbalance bottleneck by using the *sampling-based numerical* (SBN) approximation method, where we generate a sufficiently large set $\mathcal{L}_1$ of randomly generated profile vectors $\mathbf{L}$, and approximate $\overline{T}(\gamma,\gamma_u,\alpha)$ for a fixed α as

$$\overline{T}(\gamma,\gamma_u,\alpha)\approx\frac{1}{|\mathcal{L}_1|}\sum_{\mathbf{L}\in\mathcal{L}_1}T(\mathbf{L},\alpha), \tag{15}$$

where $T(\mathbf{L},\alpha)$ is given in (4). Then, we numerically find the approximate optimal partition parameter $\hat{\alpha}$ that minimizes the $\overline{T}(\gamma,\gamma_u,\alpha)$ in (15) and approximate $\overline{T}(\gamma,\gamma_u)$ as

$$\overline{T}(\gamma,\gamma_u)\approx\frac{1}{|\mathcal{L}_1|}\sum_{\mathbf{L}\in\mathcal{L}_1}T(\mathbf{L},\hat{\alpha}). \tag{16}$$

Following the same approach, for the uniform user-to-cell association, we numerically find the approximate optimal partition parameter $\hat{\alpha}$ that minimizes the $T(\mathbf{L}_{uni},\hat{\alpha})$ in (4), where $\mathbf{L}_{uni}$ is a uniform profile vector, and approximate $T_{min}(\gamma,\gamma_u)$ as.

$$T_{min}(\gamma,\gamma_u)\approx T(\mathbf{L}_{uni},\hat{\alpha}) \tag{17}$$

The corresponding approximate performance gap is then evaluated by dividing $\overline{T}(\gamma,\gamma_u)$ by $T_{min}(\gamma,\gamma_u)$. For $|\mathcal{L}_1|=10000$, and $K=\Lambda=500$, Figure 2 compares the SBN approximation of multiplicative performance gap $G(\gamma,\gamma_u)=\frac{\overline{T}(\gamma,\gamma_u)}{T_{min}(\gamma,\gamma_u)}$ between the average delivery time $\overline{T}(\gamma,\gamma_u)$ for a random user-to-cell association and the minimum worst-case delivery time $T_{min}(\gamma,\gamma_u)$ for a uniform user-to-cell association. This figure highlights the effectiveness of our proposed two-layered shared-cache setting in mitigating the impact of the cache-load imbalance. We see that even for small γ_u, the performance gap $G(\gamma,\gamma_u)$ is significantly less than the case of $\gamma_u=0$ i.e., the original stochastic shared-cache setting [9].

V. Conclusions

The work explored the effectiveness of the two-layered coded caching scheme in resolving the cache-load imbalance bottleneck of the coded caching in a stochastic shared-cache network, where each user can appear within the coverage area of one of Λ cache-enabled cells with equal probability. We identified the exact scaling laws of the average delivery time of our proposed two-layered coded caching scheme, and showed that this scheme significantly mitigates, and in certain memory regimes completely nullifies the detrimental performance deterioration due to randomness. In a nutshell, empowering users with an additional layer –even a modest amount– of caching in stochastic shared-cache networks can be a viable solution to resolve the cache-load imbalance bottleneck.

Appendix

A. Proof of Theorem 1

We know from [9, Theorem 3] that $E_{\mathbf{L}}[T_h(\mathbf{L},\alpha)]$ corresponding to the delivery of the first part of each requested file is given as

$$E_{\mathbf{L}}[T_h(\mathbf{L},\alpha)] = \begin{cases} \Theta\Big(K\frac{1-\frac{\gamma}{\alpha}}{1+\Lambda\frac{\gamma}{\alpha}}\frac{\Lambda\log\Lambda}{K\log\frac{\Lambda\log\Lambda}{K}}\Big) & \text{if } K\in\Big[\frac{\Lambda}{\text{polylog}(\Lambda)}, o\left(\Lambda\log\Lambda\right)\Big] \\ \Theta\left(K\frac{1-\frac{\gamma}{\alpha}}{1+\Lambda\frac{\gamma}{\alpha}}\right) & \text{if } K=\Omega\left(\Lambda\log\Lambda\right). \end{cases} \tag{18}$$

Then, for the case when $K \in \Big[\frac{\Lambda}{\text{polylog}(\Lambda)}, o\left(\Lambda\log\Lambda\right)\Big]$, the average delay $\overline{T}\left(\gamma,\gamma_u,\alpha\right)$ scales as

$$\begin{aligned}\overline{T}\left(\gamma,\gamma_u,\alpha\right) &= \Theta\left(\alpha K\frac{\alpha-\gamma}{\alpha+\Lambda\gamma}\frac{\Lambda}{K}\frac{\log\Lambda}{log\left(\frac{\Lambda}{K}\log\Lambda\right)}\right.\\ &\left.+(1-\alpha)K\frac{1-\alpha-\gamma_u}{1-\alpha+K\gamma_u}\right)\\ &\overset{(a)}{=}\Theta\left(\alpha\frac{\alpha-\gamma}{\gamma}\frac{\log\Lambda}{log\left(\frac{\Lambda}{K}\log\Lambda\right)}+(1-\alpha)\frac{1-\alpha-\gamma_u}{\gamma_u}\right), \end{aligned}\tag{19}$$

where in step (a), we used the fact that $\Lambda\gamma = \omega(\alpha)$ and $K\gamma_u = \omega(1-\alpha)$. Let $c_1 = \frac{\log\Lambda}{log\left(\frac{\Lambda}{K}\log\Lambda\right)}$

$$\begin{aligned}\overline{T}\left(\gamma,\gamma_u,\alpha\right) &= \Theta\left(\frac{c_1\alpha^2-c_1\alpha\gamma}{\gamma}+\frac{\alpha^2+\alpha(\gamma_u-2)+1-\gamma_u}{\gamma_u}\right)\\ &= \Theta\left(\frac{c_1\alpha^2\gamma_u-c_1\alpha\gamma\gamma_u+\alpha^2\gamma+\alpha(\gamma_u-2)\gamma+\gamma-\gamma\gamma_u}{\gamma\gamma_u}\right)\\ &= \Theta\left(\frac{\alpha^2(c_1\gamma_u+\gamma)+\alpha(\gamma_u\gamma-c_1\gamma\gamma_u-2\gamma)+\gamma-\gamma\gamma_u}{\gamma\gamma_u}\right). \end{aligned}\tag{20}$$

Similarly, for the case $K=\Omega\left(\Lambda\log\Lambda\right)$, the average delay $\overline{T}\left(\gamma,\gamma_u,\alpha\right)$ is bounded by

$$\begin{aligned}\overline{T}\left(\gamma,\gamma_u,\alpha\right) &= \Theta\left(\alpha K\frac{\alpha-\gamma}{\alpha+\Lambda\gamma}+(1-\alpha)K\frac{1-\alpha-\gamma_u}{1-\alpha+K\gamma_u}\right)\\ &\overset{(b)}{=}\Theta\left(\alpha\frac{\alpha-\gamma}{\gamma}\frac{K}{\Lambda}+(1-\alpha)\frac{1-\alpha-\gamma_u}{\gamma_u}\right). \end{aligned}\tag{21}$$

where in step (b), we used the fact that $\Lambda\gamma = \omega(\alpha)$ and $K\gamma_u = \omega(1-\alpha)$. Let $c_2 = \frac{K}{\Lambda}$, then we have

$$\begin{aligned}\overline{T}\left(\gamma,\gamma_u,\alpha\right) &= \Theta\left(\frac{\alpha^2(c_2\gamma_u+\gamma)}{\gamma\gamma_u}\right.\\ &\left.+\frac{\alpha(-c_2\gamma\gamma_u+\gamma_u\gamma-2\gamma)+\gamma-\gamma\gamma_u}{\gamma\gamma_u}\right). \end{aligned}\tag{22}$$

This concludes the proof of Theorem 1.

B. Proof of Lemma 1

The order-optimal partition parameter $\hat{\alpha}$ is the solution to the following optimization problem

$$\min_{\alpha}\ \overline{T}\left(\gamma,\gamma_u,\alpha\right) \tag{23a}$$

subject to

$$\gamma\le\alpha\le 1-\gamma_u. \tag{23b}$$

We use the Lagrangian method to solve the problem in (23). We start with the first case when $K\in\Big[\frac{\Lambda}{\text{polylog}(\Lambda)}, o\left(\Lambda\log\Lambda\right)\Big]$, using (20), the corresponding Lagrange function $\mathcal{Y}(\alpha,\delta,\sigma)$ is

$$\begin{aligned}\mathcal{Y}(\alpha,\delta,\sigma) &= \frac{\alpha^2(c_1\gamma_u+\gamma)+\alpha(-c_1\gamma\gamma_u+\gamma_u\gamma-2\gamma)}{\gamma\gamma_u}\\ &+\frac{\gamma-\gamma\gamma_u}{\gamma\gamma_u}+\delta\left(\alpha-1+\gamma_u\right)-\sigma(\alpha-\gamma) \end{aligned}\tag{24}$$

where $\sigma,\delta\in\mathbb{R}$ and $c_1=\frac{\log\Lambda}{log\left(\frac{\Lambda}{K}\log\Lambda\right)}$. The Karush-Kuhn-Tucker (KKT) conditions for (23) are then given by

$$\frac{\partial\mathcal{Y}(\hat{\alpha},\hat{\delta},\hat{\sigma})}{\partial\alpha}=0, \tag{25}$$

$$\hat{\sigma}(\hat{\alpha}-\gamma)=0, \tag{26}$$

$$\hat{\delta}\left(\hat{\alpha}-1+\gamma_u\right)=0, \tag{27}$$

$$\hat{\delta}\ge 0, \tag{28}$$

$$\hat{\sigma}\ge 0, \tag{29}$$

where $\hat{\alpha},\hat{\delta}$, and $\hat{\sigma}$ represent the optimized values. Then from (25), we have

$$\begin{aligned}\frac{\partial\mathcal{Y}(\hat{\alpha},\hat{\delta},\sigma)}{\partial\alpha} &= \frac{2\hat{\alpha}(c_1\gamma_u+\gamma)}{\gamma\gamma_u}\\ &+\frac{(-c_1\gamma\gamma_u+\gamma_u\gamma-2\gamma)}{\gamma\gamma_u}+\hat{\delta}-\hat{\sigma}=0. \end{aligned}\tag{30}$$

We obtain the solution by dividing the operating domain of the partition parameter into the following three regimes:

- Regime I: $\gamma<\hat{\alpha}<1-\gamma_u$
- Regime II: $\hat{\alpha}=1-\gamma_u$
- Regime III: $\hat{\alpha}=\gamma$.

First, for Regime I, when $\gamma<\hat{\alpha}<1-\gamma_u$, we have $\hat{\delta}=\hat{\sigma}=0$ and from (30), we obtain

$$\hat{\alpha}=\frac{(c_1\gamma\gamma_u-\gamma_u\gamma+2\gamma)}{2(c_1\gamma_u+\gamma)} \tag{31}$$

Next, for Regime II, when $\hat{\alpha}=1-\gamma_u$, we have $\hat{\sigma}=0$ and from (30), we obtain

$$\begin{aligned}\hat{\delta} &= -\frac{2(1-\gamma_u)(c_1\gamma_u+\gamma)+(-c_1\gamma\gamma_u+\gamma_u\gamma-2\gamma)}{\gamma\gamma_u}\\ &= -\frac{2c_1\gamma_u+2\gamma-2c_1\gamma_u^2-2\gamma\gamma_u-c_1\gamma\gamma_u+\gamma_u\gamma-2\gamma}{\gamma\gamma_u}\end{aligned}$$

$$= -\frac{2c_1 - 2c_1\gamma_u - \gamma - c_1\gamma}{\gamma}$$
$$= 1 - \frac{c_1(2 - 2\gamma_u - \gamma)}{\gamma}. \quad (32)$$

We know from (28) that $\hat{\delta} \geq 0$, thus $\hat{\alpha} = 1 - \gamma_u$ is a feasible solution only if

$$c_1(2 - 2\gamma_u - \gamma) \leq \gamma$$
$$\gamma_u + \gamma \geq 2 - \frac{\gamma}{c_1} - \gamma_u.$$

However under our assumption that $\gamma_u + \gamma \leq 1$ is not possible as we know that when $K \in \left[\frac{\Lambda}{\text{polylog}(\Lambda)}, o\left(\Lambda \log \Lambda\right)\right]$, $c_1 > 1$. Thus $\hat{\alpha} = 1 - \gamma_u$ is not a feasible solution as it does not satisfy (28).

Finally, for Regime III, when $\hat{\alpha} = \gamma$, we have $\hat{\delta} = 0$ and from (30), we obtain

$$\hat{\sigma} = \frac{2\gamma(c_1\gamma_u + \gamma) - c_1\gamma\gamma_u + \gamma_u\gamma - 2\gamma}{\gamma\gamma_u}$$
$$= \frac{2(c_1\gamma_u + \gamma) - c_1\gamma_u + \gamma_u - 2}{\gamma_u}$$
$$= \frac{c_1\gamma_u + 2\gamma + \gamma_u - 2}{\gamma_u}. \quad (33)$$

We know from (29) that $\hat{\sigma} \geq 0$, thus $\hat{\alpha} = \gamma$ is a feasible solution only if $c_1\gamma_u + 2\gamma + \gamma_u - 2 \geq 0$. Finally, (31) and (33) concludes the proof of (10) for the case of $K \in \left[\frac{\Lambda}{\text{polylog}(\Lambda)}, o\left(\Lambda \log \Lambda\right)\right]$.

Next, we solve the problem (23) for the case of $K = \Omega\left(\Lambda \log \Lambda\right)$. We can see that if we replace c_1 with c_2 in (20), we obtain (22), which is the average delay for the case of $K = \Omega\left(\Lambda \log \Lambda\right)$. Since both c_2 and c_1 are constants and do not depend on α, the optimal solution for the case of $K = \Omega\left(\Lambda \log \Lambda\right)$ is obtained using the same approach by simply replacing c_1 with c_2. This concludes the proof of Lemma 1.

C. Proof of Theorem 2

We begin with the case for $\gamma_u \geq \frac{2-2\gamma}{1+c}$, from (10), we have $\hat{\alpha} = \gamma$. Then, inserting $\hat{\alpha}$ into (8) yields

$$\overline{T}\left(\gamma, \gamma_u\right) = \overline{T}\left(\gamma, \gamma_u, \hat{\alpha}\right)$$
$$= \Theta\left(\frac{\gamma^2(c\gamma_u + \gamma) + \gamma(-c\gamma\gamma_u + \gamma_u\gamma - 2\gamma) + \gamma - \gamma\gamma_u}{\gamma\gamma_u}\right)$$
$$= \Theta\left(\frac{c\gamma_u\gamma + \gamma^2 - c\gamma\gamma_u + \gamma_u\gamma - 2\gamma + 1 - \gamma_u}{\gamma_u}\right)$$
$$= \Theta\left((1-\gamma)\frac{1 - \gamma_u - \gamma}{\gamma_u}\right). \quad (34)$$

Next for the case for $\gamma_u < \frac{2-2\gamma}{1+c}$, from (10), we have $\hat{\alpha} = \frac{c\gamma\gamma_u - \gamma_u\gamma + 2\gamma}{2(c\gamma_u + \gamma)}$. Then, inserting $\hat{\alpha}$ into (8) yields

$$\overline{T}\left(\gamma, \gamma_u\right) = \overline{T}\left(\gamma, \gamma_u, \hat{\alpha}\right)$$
$$= \Theta\left(\frac{\frac{(c\gamma\gamma_u - \gamma_u\gamma + 2\gamma)^2}{4(c\gamma_u + \gamma)} - \frac{(c\gamma\gamma_u - \gamma_u\gamma + 2\gamma)^2}{2(c\gamma_u + \gamma)} + \gamma - \gamma\gamma_u}{\gamma\gamma_u}\right)$$
$$= \Theta\left(-\frac{(c\gamma\gamma_u - \gamma_u\gamma + 2\gamma)^2}{4(c\gamma_u + \gamma)\gamma\gamma_u} + \frac{1 - \gamma_u}{\gamma_u}\right)$$
$$= \Theta\left(\frac{2c\gamma\gamma_u - c^2\gamma_u\gamma - \gamma\gamma_u}{4c\gamma_u + 4\gamma} + \frac{c(1 - \gamma - \gamma_u)}{c\gamma_u + \gamma}\right)$$
$$= \Theta\left(-\frac{\gamma\gamma_u(-2c + c^2 + 1)}{4c\gamma_u + 4\gamma} + \frac{c(1 - \gamma - \gamma_u)}{c\gamma_u + \gamma}\right)$$
$$= \Theta\left(\frac{c(1 - \gamma - \gamma_u)}{c\gamma_u + \gamma} - \frac{\gamma\gamma_u(c-1)^2}{4c\gamma_u + 4\gamma}\right). \quad (35)$$

This concludes the proof of Theorem 2.

D. Proof of Lemma 2

We know from [9, Equation (5), (6)] and (4) that for a fixed partition parameter α, the worst-case delivery time for a uniform user-to-cell association is

$$T_{min}(\gamma, \gamma_u, \alpha) = \Theta\left(\alpha K \frac{1 - \frac{\gamma}{\alpha}}{1 + \Lambda\frac{\gamma}{\alpha}} + (1 - \alpha)\frac{K(1 - \frac{\gamma_u}{1-\alpha})}{1 + \frac{K\gamma_u}{1-\alpha}}\right). \quad (36)$$

From (21), we can see that $T_{min}(\gamma, \gamma_u, \alpha)$ is exactly equal to the worst-case average delivery time for the case of $K = \Omega\left(\Lambda \log \Lambda\right)$. Therefore, the proof of Lemma 2 follows directly from the proof of Lemma 1 and proof of Theorem 2 for the case of $K = \Omega\left(\Lambda \log \Lambda\right)$.

REFERENCES

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[2] J. Hachem, N. Karamchandani, and S. Diggavi, "Coded caching for multi-level popularity and access," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 3108–3141, May 2017.

[3] E. Parrinello, A. Unsal, and P. Elia, "Fundamental limits of coded caching with multiple antennas, shared caches and uncoded prefetching," *IEEE Trans. Inf. Theory*, vol. 66, no. 4, pp. 2252–2268, Apr. 2020.

[4] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless d2d networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb 2016.

[5] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Trans. Inf. Theory*, vol. 63, no. 2, pp. 1146–1158, Feb 2017.

[6] B. Serbetci, E. Lampiris, T. Spyropoulos, and P. Elia, "Augmenting multiple-transmitter coded caching using popularity knowledge at the transmitters," in *Proc. 18th Int. Symp. on Mod. and Opt. in Mob., Ad Hoc, and Wireless Net. (WiOPT)*, Avignon, June 2020, pp. 1–8.

[7] A. Malik, B. Serbetci, and P. Elia, "Coded caching in networks with heterogeneous user activity," 2022. [Online]. Available: https://arxiv.org/pdf/2201.10250.pdf

[8] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1176–1188, Jun. 2018.

[9] A. Malik, B. Serbetci, E. Parrinello, and P. Elia, "Fundamental limits of stochastic shared-cache networks," *IEEE Trans. Commun.*, vol. 69, no. 7, pp. 4433–4447, 2021.

[10] A. Sengupta, R. Tandon, and O. Simeone, "Fog-aided wireless networks for content delivery: Fundamental latency tradeoffs," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6650–6678, Oct. 2017.

[11] A. Malik, B. Serbetci, E. Parrinello, and P. Elia, "Stochastic Analysis of Coded Multicasting for Shared Caches Networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Taipei, Taiwan, Dec. 2020, pp. 1–6.

[12] J. S. P. Roig, D. Gündüz, and F. Tosato, "Interference networks with caches at both ends," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, May 2017, pp. 1–6.

[13] M. Bayat, R. K. Mungara, and G. Caire, "Achieving spatial scalability for coded caching via coded multipoint multicasting," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 227–240, Jan. 2019.

[14] A. Malik, B. Serbetci, and P. Elia, "Stochastic coded caching with optimized shared-cache sizes and reduced subpacketization," 2021. [Online]. Available: https://arxiv.org/pdf/2112.14114.pdf