Fundamental Limits of Stochastic Shared-Cache Networks

Adeel Malik, Berksan Serbetci, Emanuele Parrinello, Petros Elia

***Abstract*—The work establishes the exact performance limits of stochastic coded caching when users share a bounded number of cache states, and when the association between users and caches, is random. Under the premise that more balanced user-to-cache associations perform better than unbalanced ones, our work provides a statistical analysis of the average performance of such networks, identifying in closed form, the exact optimal average delivery time. To insightfully capture this delay, we derive easy-to-compute closed-form analytical bounds that prove tight in the limit of a large number Λ of cache states. In the scenario where delivery involves K users, we conclude that the multiplicative performance deterioration due to randomness — as compared to the well-known deterministic uniform case — can be unbounded and can scale as $\Theta\left(\frac{\log\Lambda}{\log\log\Lambda}\right)$ at $K=\Theta(\Lambda)$, and that this scaling vanishes when $K=\Omega(\Lambda\log\Lambda)$. To alleviate this adverse effect of cache-load imbalance, we consider various load-balancing methods, and show that employing proximity-bounded load balancing with an ability to choose from h neighboring caches, the aforementioned scaling reduces to $\Theta\left(\frac{\log(\Lambda/h)}{\log\log(\Lambda/h)}\right)$, while when the proximity constraint is removed, the scaling is of a much slower order $\Theta(\log\log\Lambda)$. The above analysis is extensively validated numerically.**

***Index Terms*—Coded caching, shared-cache, load balancing, heterogeneous networks, femtocaching.**

I. Introduction

EVER-INCREASING volumes of mobile data traffic, have brought to the fore the need for new solutions that can serve a continuously increasing number of users, and do so with a limited amount of network bandwidth resources. In this context, cache-enabled wireless networks have emerged as a promising solution that can transform the storage capability of the nodes into a new and powerful network resource.

The potential of such cache-enabled wireless networks has been dramatically elevated following the seminal publication in [2] which introduced the concept of *coded caching*, and which revealed that — in theory — an unbounded number of users can be served even with a bounded amount of network resources. This was a consequence of a novel cache placement algorithm that enabled the delivery of independent content to many users at a time. Since then, several extensions of the basic coded caching setting have been studied. Such works include the study of coded caching for arbitrary file popularity distributions [3]–[5], various optimality results in [6]–[8], results for various topology models [9]–[11], for MIMO broadcast channels [12], [13], for PHY-based coded caching [11]–[19], for a variety of heterogeneous networks (HetNets) [20], [21], D2D networks [22], and other settings as well [23]–[27].

The authors are with the Communication Systems Department at EURECOM, Sophia Antipolis, 06410, France (email: malik@eurecom.fr, serbetci@eurecom.fr, parrinel@eurecom.fr, elia@eurecom.fr). 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). This paper was presented in part at the 2020 IEEE Global Communications (GLOBECOM) Conference [1].

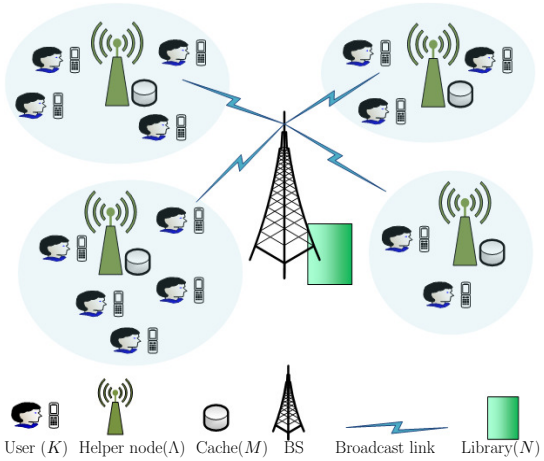

Fig. 1. An instance of a cache-aided heterogeneous network.

A. Coded caching networks with shared caches

Pivotal to the development of larger, realistic coded caching networks is the so-called *shared-cache setting*, where different users are forced to benefit from the same cache content. This setting is of great importance because it reflects promising scenarios as well as unavoidable constraints.

Such a promising scenario can be found in the context of cache-enabled heterogeneous networks, where a central transmitter (a base station) delivers content to a set of interfering users, with the assistance of cache-enabled helper nodes that serve as caches to the users. An instance of such a network is illustrated in Figure 1. Such networks capture modern trends that envision a central base-station covering a larger area, in tandem with a multitude of smaller helper nodes each covering smaller cells. In this scenario, any user that appears in a particular small cell, can benefit from the cache-contents of the single helper node covering that cell.

In the context of coded caching, an early work on this scenario can be found in [20], which employed the uniform user-to-cache association assumption where each helper node is associated to an equal number of users. This assumption was removed in [21], which — under the assumption that content cache placement is uncoded as well as agnostic to the user-to-cache association — identified the exact optimal worst-case delivery time (i.e, the case of users' demand vector which requires the longest delivery time), as a function of the user-to-cache association profile that describes the number of users served by each cache. A similar setting was studied in [28] for the case of non-distinct requests, as well as in [29]–[31] for

the topology-aware (non-agnostic) scenario, where the user-to-cache association is known during cache placement. In this context, the work in [29] proposed a novel coded placement that exploits knowledge of the user-to-cache association, while the work in [30] used this same knowledge, to modulate cache-sizes across the different helper nodes as a function of how many users they serve. Similarly, the work in [31] optimized over the cache sizes, again as a function of the load of each cache, and then proceeded to design a novel coded caching scheme which substantially outperforms the optimal scheme in [21]; the latter designed for the scenario where the cache placement is oblivious to the user-to-cache association phase. It is interesting to note that to a certain extent, this same shared-cache setting also applies to the scenario where each user requests multiple files (see for example [32]).

Very importantly, this same shared-cache setting is directly related to the unavoidable subpacketization bottleneck because this bottleneck can force the use of a reduced number of distinct cache states that must be inevitably shared among the many users[1]. This number of distinct cache states, henceforth denoted as Λ, will be forced under most realistic assumptions, to be substantially less than the total number of users, simply because most known coded caching techniques require file sizes that scale exponentially with Λ (see [12], [33]–[36]). In a hypothetical scenario where coded caching is applied across a city of, let's say, one million mobile users, one would have to assign each user with one of Λ cache states (Λ independent caches), where Λ would probably be forced to be in the double or perhaps triple digits [33].

As one can see, both of the above isomorphic settings imply that during the content delivery that follows the allocation of cache-states to each user, different broadcast sessions would experience user populations that differently span the spectrum of cache states. In the most fortunate of scenarios, a transmitter would have to deliver to a set of K users that uniformly span the Λ states (such that each cache state is found in exactly K/Λ users), while in the most unfortunate of scenarios, a transmitter would encounter K users that happen to have an identical cache state. Both cases are rare instances of a stochastic process, which we explore here in order to identify the exact optimal performance of such systems.

Most of our results apply both to the heterogeneous network scenario as well as the aforementioned related subpacketization-constrained setting which was nicely studied in [37]. This interesting work in [37] introduced the main problem, and after providing upper bounds[2], introduced the challenge of identifying the fundamental limits of this same problem. This is indeed the challenge that is resolved here, where we are able to derive these fundamental limits of performance, in their exact form. The problem for which we are deriving these fundamental limits, is itself a crucial problem in the context of coded caching. Based on our analysis, and our new insight that indeed we can have an unbounded "damage" from the stochastic nature of the user-to-cache association, we proceed to provide simple and realistic load-balancing techniques. Our analysis captures the benefits of the techniques, in their exact form.

For ease of exposition, we will focus the wording of our description to the first scenario corresponding to a heterogeneous network where Λ plays the role of the number of helper nodes. All the results though of Section II certainly apply to the latter setting as well.

B. Load balancing in mobile networks

As mentioned above, we will be focusing on heterogeneous networks and will explore the statistical properties brought about by the introduction of cache-aided helper nodes. We have also suggested that performance generally suffers when the different helper nodes are unevenly loaded. For this, it is only natural that we look at basic load-balancing approaches, which have long played a pivotal role in improving the statistical behavior of wireless networks. This role was highlighted in the survey found in [38], which discussed why long-standing assumptions about cellular networks need to be rethought in the context of load balanced heterogeneous networks, and showed that natural user association metrics like signal-to-interference-plus-noise ratio (SINR) or received signal strength indication (RSSI) can lead to a major imbalance. This work gathered together the earlier works on load balancing in HetNets and compared the primary technical approaches – such as optimization, game theory and Markov decision processes – to HetNet load balancing. In the same context, various algorithms have also been proposed to optimize the traffic load by analyzing user association to servers for cellular networks [39], by providing a distributed $\alpha-$optimal user association and cell load-balancing algorithm for wireless networks [40], by developing SINR-based flexible cell association policies in heterogeneous networks [41], and even investigating traffic load balancing in backhaul-constrained cache-enabled small cell networks powered by hybrid energy sources [42].

In this paper, we build a bridge between load balancing and coded caching, with the aim of improving the network performance by balancing the user load placed on each cache. We will show that the effect of load balancing can in fact be unbounded in the limit of many caches.

C. Shared-cache setting & problem statement

We consider the shared-cache coded-caching setting where a transmitter having access to a library of N equisized files, delivers content via a broadcast link to K receiving users, with the assistance of Λ cache-enabled helper nodes. Each helper node $\lambda \in [1, 2, \ldots, \Lambda]$ is equipped with a cache of storage capacity equal to the size of M files, thus being able to store a fraction $\gamma \triangleq \frac{M}{N} \in \left[\frac{1}{\Lambda}, \frac{2}{\Lambda}, \ldots, 1\right]$ of the library. Each such helper node, which will be henceforth referred to as a 'cache', can assist in the delivery of content to any number of receiving users.

The communication process consists of three phases; the *content placement phase*, the *user-to-cache association phase*, and the *delivery phase*. The first phase involves the placement of library-content in the caches, and it is oblivious to the outcome of the next two phases. The second phase is when each user is assigned – independently of the placement phase

[1]When adopting the cache placement strategy in [2] for Λ caches, we refer the content to be placed in a single cache as a cache state.

[2]It is worth noting that the provided upper bounds in [37] can have a large gap from the here-derived optimal.

– to exactly one cache from which it can download content at zero cost. This second phase is also oblivious of the other two phases[3]. The final phase begins with users simultaneously requesting one file each, and continues with the transmitter delivering this content to the receivers. Naturally this phase is aware of the content of the caches, as well as aware of which cache assists each user.

***User-to-cache association*:** For any cache $\lambda \in [1,\dots,\Lambda]$, we denote by v_λ the number of users that are assisted by it, and we consider the *cache population vector* $\mathbf{V} = [v_1,\dots,v_\Lambda]$. Additionally we consider the sorted version $\mathbf{L} = [l_1,\dots,l_\Lambda] = sort(\mathbf{V})$, where $sort(\mathbf{V})$ denotes the sorting of vector $\mathbf{V}$ in descending order. We refer to $\mathbf{L}$ as a *profile vector*, and we note that each entry l_λ is simply the number of users assisted by the λ-th most populous (most heavily loaded) cache. Figure 1 depicts an instance of our shared-cache setting where $\mathbf{L} = [5,4,3,2]$.

***Delivery phase*:** The delivery phase commences with each user $k \in [1,\dots,K]$ requesting a single library file that is indexed by $d_k \in [1,\dots,N]$. As is common in coded caching works, we assume that each user requests a different file. Once the transmitter is notified of the *request vector* $\mathbf{d} = [d_1,\dots,d_K]$, it commences delivery over an error-free broadcast link of bounded capacity per unit of time.

D. Metric of interest

As one can imagine, any given instance of the problem, experiences a different user-to-cache association, and thus[4] a different $\mathbf{V}$. Our measure of interest is thus the average delay

$$\overline{T}(\gamma) \triangleq E_{\mathbf{V}}[T(\mathbf{V})] = \sum_{\mathbf{V}} P(\mathbf{V})T(\mathbf{V}), \tag{1}$$

where $T(\mathbf{V})$ is the worst-case delivery time[5] corresponding to any cache population vector $\mathbf{V}$, and where $P(\mathbf{V})$ is the probability that the user-to-cache association corresponds to vector $\mathbf{V}$.

More precisely, we use $T(\mathbf{V},\mathbf{d},\mathcal{X})$ to define the delivery time required by some generic caching-and-delivery scheme $\mathcal{X}$ to satisfy request vector $\mathbf{d}$ when the user-to-cache association is described by the vector $\mathbf{V}$. Our aim here is to characterize the optimal average delay

$$\begin{aligned}\overline{T}^*(\gamma) &= \min_{\mathcal{X}} E_{\mathbf{V}}\left[\max_{\mathbf{d}} T(\mathbf{V},\mathbf{d},\mathcal{X})\right] \\ &= \min_{\mathcal{X}} E_{\mathbf{L}}\left[E_{\mathbf{V}_\mathbf{L}}\left[\max_{\mathbf{d}} T(\mathbf{V},\mathbf{d},\mathcal{X})\right]\right], \end{aligned} \tag{2}$$

where the minimization is over all possible caching and delivery schemes $\mathcal{X}$, and where $E_{\mathbf{V}_\mathbf{L}}$ denotes the expectation over all vectors $\mathbf{V}$ whose sorted version is equal to some fixed $sort(\mathbf{V}) = \mathbf{L}$. Consequently the metric of interest takes the form

$$\overline{T}(\gamma) = E_{\mathbf{L}}[T(\mathbf{L})] = \sum_{\mathbf{L}} P(\mathbf{L})T(\mathbf{L}), \tag{3}$$

where $T(\mathbf{L}) \triangleq E_{\mathbf{V}_\mathbf{L}}\left[\max_{\mathbf{d}} T(\mathbf{V},\mathbf{d})\right]$, and where

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

is simply the cumulative probability over all $\mathbf{V}$ for which $sort(\mathbf{V}) = \mathbf{L}$.

We will consider here the uncoded cache placement scheme in [2], and the delivery scheme in [21], [37], which will prove to be optimal for our setting under the common assumption of uncoded cache placement. This multi-round delivery scheme introduces — for any $\mathbf{V}$ such that $sort(\mathbf{V}) = \mathbf{L}$ — a worst-case delivery time of

$$T(\mathbf{L}) = \sum_{\lambda=1}^{\Lambda-t} l_\lambda \frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}}, \tag{4}$$

where $t = \Lambda\gamma$.

From equation (4) we can see that the minimum delay corresponds to the case when $\mathbf{L}$ is uniform. When Λ divides K, this minimum (uniform) delay takes the well-known form

$$T_{min} = \frac{K(1-\gamma)}{1+\Lambda\gamma}, \tag{5}$$

while for general K,Λ, it takes the form[6]

$$T_{min} = \frac{\Lambda - t}{1+t}\left(\left\lfloor \frac{K}{\Lambda} \right\rfloor + 1 - f(\hat{K})\right), \tag{6}$$

where $\hat{K} = K - \left\lfloor \frac{K}{\Lambda} \right\rfloor \Lambda$, $f(\hat{K}) = 1$ when $\hat{K} = 0$, $f(\hat{K}) = 0$ when $\hat{K} \geq \Lambda - t$, and $f(\hat{K}) = \frac{\prod_{i=t+1}^{\hat{K}+t}(\Lambda-i)}{\prod_{j=0}^{\hat{K}-1}(\Lambda-j)}$ when $\hat{K} < \Lambda - t$. The proof of this is straightforward, but for completeness it can be found in the longer version of this work [43, Appendix F]. The above T_{min} is optimal under the assumption of uncoded placement (cf. [21]).

On the other hand, for any other (now non-uniform) $\mathbf{L}$, the associated delay $T(\mathbf{L})$ will exceed T_{min} (see [21] for the proof, and see Figure 2 for a few characteristic examples), and thus so will the average delay

$$\begin{aligned} E_{\mathbf{L}}[T(\mathbf{L})] &= \sum_{\mathbf{L}\in\mathcal{L}} P(\mathbf{L})T(\mathbf{L}) \\ &= \sum_{\lambda=1}^{\Lambda-t}\sum_{\mathbf{L}\in\mathcal{L}} P(\mathbf{L}) l_\lambda \frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}} = \sum_{\lambda=1}^{\Lambda-t} E[l_\lambda]\frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}}, \end{aligned} \tag{7}$$

where $\mathcal{L}$ describes the set of all possible profile vectors $\mathbf{L}$ (where naturally $\sum_{\lambda=1}^{\Lambda} l_\lambda = K$), and where $E[l_\lambda]$ is the expected number of users in the λ-th most populous cache[7].

E. Our contribution

In this work we assume that each user can appear in the coverage area of any particular cache-enabled cell (i.e., can

[3]This assumption is directly motivated by the time-scales of the problem, as well as by the fact that in the heterogeneous setting, the user-to-cache association is a function of the geographical location of the user. Note that users can only be associated to caches when users are within the coverage of caches, and a dynamic user-to-cache association that requires continuous communication between the users and the server may not be desirable as one seeks to minimize the network load overhead and avoid the handover.

[4]We briefly note that focusing on $\mathbf{V}$ rather than the sets of users connected to each cache, maintains all the pertinent information, as what matters for the analysis is the number of users connected to each cache and not the index (identity) of the users connected to that cache.

[5]This delay corresponds to the time needed to complete the delivery of any file-request vector $\mathbf{d}$, where the time scale is normalized such that a unit of time corresponds to the optimal amount of time needed to send a single file from the transmitter to the receiver, had there been no caching and no interference.

[6]When $K/\Lambda \notin \mathbb{Z}^+$, the best-case delay corresponds to having $l_\lambda = \lfloor K/\Lambda \rfloor + 1$ for $\lambda \in \left[1,2,\cdots,\hat{K}\right]$ and $l_\lambda = \lfloor K/\Lambda \rfloor$ for $\lambda \in \left[\hat{K}+1,\hat{K}+2,\cdots,\Lambda\right]$, where $\hat{K} = K - \lfloor K/\Lambda \rfloor \Lambda$.

[7]It is straightforward to see that $\sum_{\mathbf{L}\in\mathcal{L}} l_\lambda P(\mathbf{L})$ is equivalent to $\sum_{j=0}^{K} jP(l_\lambda = j) = E[l_\lambda]$, where $P(l_\lambda = j) = \sum_{\mathbf{L}\in\mathcal{L}:\mathbf{L}(\lambda)=j} P(\mathbf{L})$.

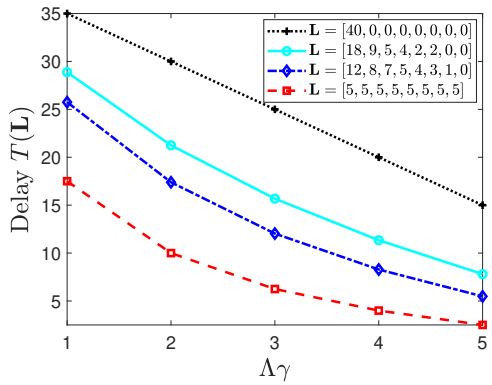

Fig. 2. Delay $T(\mathbf{L})$ for different profile vectors $\mathbf{L}$, for K=40 and Λ=8.

be associated to any particular cache) with equal probability. We will identify the optimal average delay $\overline{T}^*(\gamma)$ and the corresponding (multiplicative) performance deterioration

$$G(\gamma) = \frac{\overline{T}^*(\gamma)}{T_{min}} \tag{8}$$

experienced in this random setting. Our aim is to additionally provide expressions that can either be evaluated in a numerically tractable way, or that can be asymptotically approximated in order to yield clear insight. The following are our contributions, step by step.

- In Section II-A, we characterize in closed form the exact optimal average delay $\overline{T}^*(\gamma)$, optimized over all placement and delivery schemes under the assumption of uncoded cache placement and under the assumption that each user can appear in the coverage area of any particular cache-enabled cell with equal probability.
- To simplify the numerical interpretation of the above expression, we propose in Section II-B analytical bounds that can be calculated efficiently.
- In Section II-C, we characterize the exact scaling laws of performance. It is interesting to see that the aforementioned multiplicative deterioration $G(\gamma) = \frac{\overline{T}^*(\gamma)}{T_{min}}$ can in fact be unbounded, as Λ increases. For example, when $K = \Theta(\Lambda)$ (i.e., when K matches the order of Λ), the performance deterioration scales exactly as $\Theta\left(\frac{\log \Lambda}{\log \log \Lambda}\right)$, whereas when K increases, this deterioration gradually reduces, and ceases to scale when $K = \Omega(\Lambda \log \Lambda)$.
- In Section III, we use two load-balancing approaches to alleviate the effect of randomness. In the practical scenario where we are given a choice to associate a user to the least loaded cache from a randomly chosen group of h *neighboring* helper nodes, the performance deterioration stops scaling as early as $K = \Omega\left(\frac{\Lambda}{h} \log \frac{\Lambda}{h}\right)$. An even more dramatic improvement can be seen when the aforementioned neighboring/proximity constraint is lifted. The above reveals that load balancing, when applicable, can play a crucial role in significantly reducing the performance deterioration due to random user-to-cache association.
- In Section IV, we perform extensive numerical evaluations that validate our analysis.
- In Section V, we extend our analysis to the scenario where cache population intensities (i.e, probability that a user can appear in the coverage area of any particular cache-enabled cell) are following a non-uniform distribution.

F. Notations

Throughout this paper, we use the notation $[x] \triangleq [1, 2, \ldots, x]$, and we use $\mathbf{A}/\mathbf{B}$ to denote the difference set that consists of all the elements of set $\mathbf{A}$ not in set $\mathbf{B}$. Unless otherwise stated, logarithms are assumed to have base 2. 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 $\log x$.

II. Main Results

In this section we present our main results on the performance of the K-user broadcast channel with Λ caches, each of normalized size γ, and a uniformly random user-to-cache association process. As noted, the analysis applies both to the Λ-cell heterogeneous network, as well as to the isomorphic subpacketization-constrained setting.

A. Exact characterization of the optimal average delay

We proceed to characterize the exact optimal average delay $\overline{T}^*(\gamma)$. Crucial in this characterization will be the vector $\mathbf{B_L} = [b_1, b_2, \ldots, b_{|B_\mathbf{L}|}]$, where each element $b_j \in \mathbf{B_L}$ indicates the number of caches in a distinct group of caches in which each cache has the same load[8]. Under the assumption that each user can be associated to any particular cache with equal probability, the optimal average delay $\overline{T}^*(\gamma)$ — optimized over all coded caching strategies with uncoded placement — is given by the following theorem.

Theorem 1. *In the K-user, Λ-caches setting with normalized cache size γ and a random user-to-cache association, the average delay*

$$\overline{T}^*(\gamma) = \sum_{\lambda=1}^{\Lambda-t} \sum_{\mathbf{L}\in\mathcal{L}} \frac{K!\ t!\ (\Lambda-t)!\ l_\lambda \binom{\Lambda-\lambda}{t}}{\Lambda^K \prod_{i=1}^{\Lambda} l_i!\ \prod_{j=1}^{|\mathbf{B_L}|} b_j!} \tag{9}$$

is exactly optimal under the assumption of uncoded placement.

Proof: The proof can be found in Appendix A. ∎

One can now easily see that when $\frac{K}{\Lambda} \in \mathbb{Z}^+$, the optimal multiplicative deterioration $G(\gamma) = \frac{\overline{T}^*(\gamma)}{T_{min}}$ takes the form

$$G(\gamma) = \sum_{\lambda=1}^{\Lambda-t} \sum_{\mathbf{L}\in\mathcal{L}} \frac{(K-1)!\ (\Lambda-t-1)!\ (t+1)!\ l_\lambda \binom{\Lambda-\lambda}{t}}{\Lambda^{K-1}\ \prod_{i=1}^{\Lambda} l_i!\ \prod_{j=1}^{|\mathbf{B_L}|} b_j!}. \tag{10}$$

Remark 1. *Theorem 1 provides the exact optimal performance in the random association setting, as well as a more*

[8]For example, for a profile vector $\mathbf{L} = [5, 5, 3, 3, 3, 2, 1, 0, 0]$, there are five distinct groups in terms of having the same load, then the corresponding vector $\mathbf{B_L} = [2, 3, 1, 1, 2]$, because two caches have a similar load of five users, three caches have a similar load of three users, two caches have a similar load of zero and all other caches have distinct number of users.

	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{V}\rvert$
$K=10$	42	92378
$K=20$	530	10015005
$K=30$	3590	211915132
$K=40$	16928	2.054455634×10^{9}
$K=50$	62740	$1.2565671261\times10^{10}$

TABLE I
SIZE OF $\mathcal{L}$ AND $\mathcal{V}$ ($\Lambda=10$)

efficient way to evaluate this performance compared to the state of the art (SoA) (cf. [37, Theorem 1]). The worst-case computational time complexity for calculating the exact optimal average delay $\overline{T}^*(\gamma)$ is $O\left(\max(K,|\mathcal{L}|\,\Lambda)\right)$ as compared to the $O\left(\max(K,|\mathcal{V}|\,\Lambda\log\Lambda)\right)$ for the case of [37, Theorem 1]. This speedup is due to the averaging being over the much smaller set $\mathcal{L}$ of all $\mathbf{L}$, rather than over the set $\mathcal{V}$ of all $\mathbf{V}$ (see Table I for a brief comparison). The time complexities mentioned above do not include the cost of creating the sets $\mathcal{L}$ and $\mathcal{V}$. However, we note that the creation of $\mathcal{V}$ is a so-called weak composition problem, whereas the creation of $\mathcal{L}$ is an integer partition problem [44]. It is easy to verify that the complexities of the algorithms for the integer partition problem are significantly lower than the ones for the weak composition problem [45]–[48].

Despite the aforementioned speedup, exact evaluation of (9) can still be computationally expensive for large parameters. This motivates our derivation of much-faster to evaluate analytical bounds on $\overline{T}^*(\gamma)$, which we provide next.

B. Computationally efficient bounds on the optimal performance

The following theorem bounds the optimal average delay $\overline{T}^*(\gamma)$.

Theorem 2. *In the K-user, Λ-cache setting with normalized cache size γ and a random user-to-cache association, the optimal average delay $\overline{T}^*(\gamma)$ is bounded by*

$$\overline{T}^*(\gamma)\leq K\frac{\Lambda-t}{t+1}-\sum_{\lambda=1}^{\Lambda-t}\frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}}\sum_{j=0}^{K-1}\max\left(1-\frac{\Lambda}{\lambda}(1-P_j),0\right), \tag{11}$$

and

$$\overline{T}^*(\gamma)\geq\frac{\Lambda-t}{1+t}\left(\frac{K}{\Lambda}\frac{\Lambda-t-1}{\Lambda-1}+\frac{t}{\Lambda-1}\left(K-\sum_{j=\lceil\frac{K}{\Lambda}\rceil}^{K-1}P_j\right)\right), \tag{12}$$

where

$$P_j=\sum_{i=0}^{j}\binom{K}{i}\left(\frac{1}{\Lambda}\right)^{i}\left(1-\frac{1}{\Lambda}\right)^{K-i}. \tag{13}$$

Proof: The proof is deferred to Appendix B. ∎

Remark 2. *The worst-case computational time complexity for calculating the analytical bounds on the optimal performance $\overline{T}^*(\gamma)$ based on Theorem 2 is $O\left(\max(K\log K,K\Lambda)\right)$. This is significantly better compared to the the complexity of $O\left(\max(K,|\mathcal{L}|\,\Lambda)\right)$ for the exact calculation (cf. Theorem 1) The above bound is computationally efficient due to its dependence only on the P_j (cf.* (13)*), which is the cumulative distribution function (cdf) of a random variable that follows the binomial distribution with K independent trials and $\frac{1}{\Lambda}$ success probability. To compute bounds, the value of P_j needs to be calculated for all values of $j\in[0,K-1]$, which can be computationally expensive (i.e., $O\left(K\log K\right)$). However, as is known, there exists a $\tilde{j}\in[0,K-1]$, where $P_j\approx1$. Since the cdf is a non-decreasing function in j, it is clear[9] that $P_j\approx1$ for $j>\tilde{j}$. An illustration for $K=10^6$, and $\Lambda=10^3$ is shown in Figure 3, where it is evident that $\tilde{j}<<K$.*

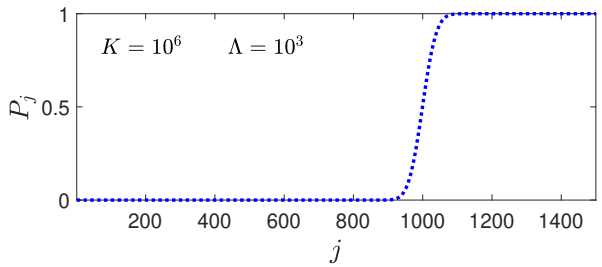

Fig. 3. Behavior of P_j for $K=10^6$ and $\Lambda=10^3$.

Directly from Theorem 2 and equation (5), we can conclude that for $\frac{K}{\Lambda}\in\mathbb{Z}^+$, the performance deterioration $G(\gamma)$ as compared to the deterministic uniform case, is bounded as

$$G(\gamma)\leq\Lambda-\frac{t+1}{K-K\gamma}\sum_{\lambda=1}^{\Lambda-t}\frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}}\sum_{j=0}^{K-1}\max\left(1-\frac{\Lambda}{\lambda}(1-P_j),0\right), \tag{14}$$

and

$$G(\gamma)\geq\frac{\Lambda-t-1}{\Lambda-1}+\frac{\Lambda}{K}\frac{t}{\Lambda-1}\left(K-\sum_{j=\lceil\frac{K}{\Lambda}\rceil}^{K-1}P_j\right), \tag{15}$$

where P_j is given in Theorem 2.

We now proceed to provide the exact scaling laws of the fundamental limits of the performance in a simple and insightful form.

C. Scaling laws of coded caching with random association

The following theorem provides the asymptotic analysis of the optimal $\overline{T}^*(\gamma)$, in the limit of large Λ.

Theorem 3. *In the K-user, Λ-caches setting with normalized cache size γ and random user-to-cache association, the optimal delay scales as*

$$\overline{T}^*(\gamma)=\begin{cases}\Theta\left(\frac{T_{min}\Lambda\log\Lambda}{K\log\frac{\Lambda\log\Lambda}{K}}\right) & \text{if } K\in\left[\frac{\Lambda}{\text{polylog}(\Lambda)},o\left(\Lambda\log\Lambda\right)\right]\\ \Theta\left(T_{min}\right) & \text{if } K=\Omega\left(\Lambda\log\Lambda\right).\end{cases} \tag{16}$$

Proof: Deferred to Appendix C. ∎

Directly from the above, we now know that the performance deterioration due to user-to-cache association randomness,

[9]The well-known De Moivre-Laplace Theorem can help us gain some intuition as to why the above method is computationally efficient and precise. In our case here, our binomial distribution — which according to the aforementioned theorem can be approximated by the normal distribution in the limit of large K — has mean K/Λ and standard deviation $\sqrt{K(\Lambda-1)/\Lambda^2}$. This simply means that the values within three standard deviations of the mean account for about $99.7\%\approx100\%$ of the set. This in turn means that $P_{\tilde{j}}\approx1$ as early on as $\tilde{j}=K/\Lambda+3\sqrt{K(\Lambda-1)/\Lambda^2}<<K$. Since $P_j\approx1$ for $j\geq\tilde{j}$, implies that (13) can be rapidly evaluated with high precision.

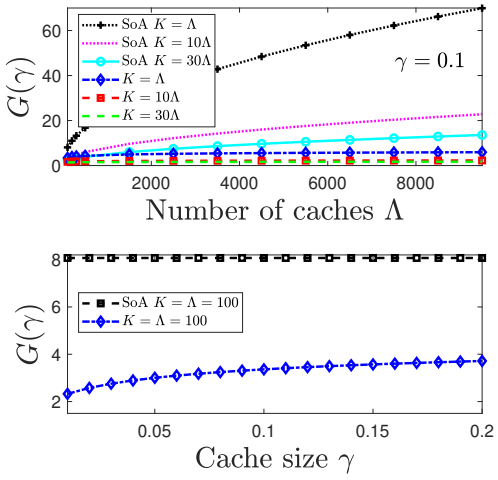

Fig. 4. Upper bound comparison with SoA.

scales as

$$G(\gamma)=\begin{cases}\Theta\left(\frac{\Lambda\log\Lambda}{K\log\frac{\Lambda\log\Lambda}{K}}\right) & \text{if } K\in\left[\frac{\Lambda}{\text{polylog}(\Lambda)},o\left(\Lambda\log\Lambda\right)\right]\\ \Theta\left(1\right) & \text{if } K=\Omega\left(\Lambda\log\Lambda\right),\end{cases} \tag{17}$$

which in turn leads to the following corollary.

Corollary 1. *The performance deterioration $G(\gamma)$ due to association randomness, scales as $\Theta\left(\frac{\log\Lambda}{\log\log\Lambda}\right)$ at $K=\Theta\left(\Lambda\right)$, and as K increases, this deterioration gradually reduces, and ceases to scale when $K=\Omega\left(\Lambda\log\Lambda\right)$.*

Proof: The proof is straightforward from Theorem 3. ■

In identifying the exact scaling laws of the problem, Theorem 3 nicely captures the following points.

- It describes the extent to which the performance deterioration increases with Λ and decreases with $\frac{K}{\Lambda}$.
- It reveals that the performance deterioration can in fact be unbounded.
- It shows how in certain cases, increasing Λ may yield diminishing returns due to the associated exacerbation of the random association problem. For example, to avoid a scaling $G(\gamma)$, one must approximately keep Λ below $e^{W(K)}$ ($W(.)$ is the Lambert W-function) such that $\Lambda\log\Lambda\leq K$.

The detrimental impact of the user-to-cache association's randomness on the delivery time motivates the need of techniques to mitigate this impact. In Section III, we show how incorporating load-balancing methods in shared cache setting can play a vital role in mitigating this impact.

D. Furthering the SoA on the subpacketization-constrained decentralized coded caching setting

As mentioned before, in general the shared cache setting is isomorphic to the subpacketization-constrained coded caching setting [33], where each cache-enabled user is forced to store the content from one of Λ cache states. In particular, the work in [37] proposed a decentralized coded caching in this subpacketization-constrained setting, where each cache-enabled user stores the content from one of Λ cache states with equal probability, which is exactly equivalent to our setting where each user appears in the coverage area of any particular cache-enabled cell with equal probability. We briefly mention below the utility of our results in this latter context.

	$\overline{T}^*(\gamma)$ in [37]	$\overline{T}^*(\gamma)$ in our work
$K=\Theta\left(\Lambda\right)$	$O\left(\sqrt{\Lambda}\right)$	$\Theta\left(\frac{\log\Lambda}{\log\log\Lambda}\right)$
$K=\Theta\left(\Lambda^a\right)$ for $1<a<2$ and $K=\Omega\left(\Lambda\log\Lambda\right)$	$O\left(\Lambda^{a/2}\right)$	$\Theta(T_{min})=\Theta\left(\frac{K}{\Lambda}\right)$ $=\Theta\left(\Lambda^{a-1}\right)$
$K=\Omega\left(\Lambda^2\right)$	$O\left(\frac{K}{\Lambda}\right)$	$\Theta(T_{min})=\Theta\left(\frac{K}{\Lambda}\right)$

TABLE II
SOA COMPARISON OF SCALING LAWS.

- Theorem 1 now identifies the exact optimal performance, as well as provides a more efficient way (see Remark 1) to evaluate this performance.
- Theorem 2 offers a new tighter upper bound on $\overline{T}^*(\gamma)$ (see Figure 4) and the only known lower bound on $\overline{T}^*(\gamma)$.
- Finally Theorem 3 completes our understanding of the scaling laws of the random association setting. For example, for the case where $K=\Theta\left(\Lambda\right)$, prior to our work, $G(\gamma)$ was known to scale at most as $\Theta\left(\sqrt{\Lambda}\right)$, whereas now we know that this deterioration scales exactly as $\Theta\left(\frac{\log\Lambda}{\log\log\Lambda}\right)$. Please refer to Table II for a detailed comparison of the known upper bounds and our exact scaling results.

III. Cache Load Balancing in Heterogeneous Networks

In the previous section, we explored the performance of coded caching when each user is associated, at random and with equal probability, to one of Λ caches. Our aim now is to reduce the detrimental impact of the user-to-cache association's randomness on the delivery time, by using load-balancing methods that introduce a certain element of choice in this association, and thus allow for *better* profile vectors. Such choice can exist naturally in different scenarios, like for example in the wireless cache-aided heterogeneous network setting, where each user can be within the communication range of more than one cache helper node.

We define a generic load-balancing method ϕ to be a function that maps the set of users $[K]$ into a cache population vector $\mathbf{V}=\phi([K])$ as a result of the load-balancing choice. Similarly as in (2), the optimal delay, given a certain load-balancing policy ϕ, is defined as

$$\overline{T}^*_\phi(\gamma)=\min_{\mathcal{X}}E_{\mathbf{V}}\left[\max_{\mathbf{d}}T\left(\phi([K]),\mathbf{d},\mathcal{X}\right)\right]. \tag{18}$$

The above definition is the same as the one in (2), with the only difference that the random variable representing the cache population vector $\mathbf{V}$ is now following a different probability distribution that depends on the load-balancing method ϕ. Employing the optimal scheme $\mathcal{X}$ from Theorem 1, the average delivery time takes the form (cf. equation (7))

$$\overline{T}_\phi(\gamma)=\sum_{\lambda=1}^{\Lambda-t}E[l_\lambda]\frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}}, \tag{19}$$

where $[l_1,l_2,\ldots,l_\Lambda]=sort(\phi([K]))$. It is important to point out that the choice of the load-balancing method can be

in general limited by some practical constraints, such as geographical constraints and operational constraints[10]. We will focus on analyzing the above, for two load-balancing methods which will prove to allow for unbounded gains.

A. Randomized load balancing with multiple choices

In the original scenario, for any given user, one cache is randomly picked to assist this user. Now we consider a load-balancing method ϕ_r which, for any given user, picks $h \geq 2$ candidate caches at random, and then associates each such user with the least loaded cache among these h caches. This static method is referred to as *randomized load balancing with multiple choices* [49], and is considered in a variety of settings (see for example [50]). The performance of this method is presented in the following result, for the limiting case of large Λ.

Theorem 4. *In the K-user, Λ-cell heterogeneous network with normalized cache size γ, where each user benefits from the least loaded cache among h randomly chosen caches, the limiting optimal delay converges to*

$$\overline{T}^*_{\phi_r}(\gamma)=\begin{cases}\Theta\left(T_{min}\frac{\Lambda\log\log\Lambda}{K\log h}\right) & \text{if } K=o\left(\frac{\Lambda\log\log\Lambda}{\log h}\right)\\ \Theta\left(T_{min}\right) & \text{if } K=\Omega\left(\frac{\Lambda\log\log\Lambda}{\log h}\right).\end{cases} \tag{20}$$

Proof: The achievability part of the theorem is deferred to Appendix D. After noticing that the definition of the optimal delay in (18) is equal to (2), optimality is proven the same way as for the optimality of Theorem 1 by following the same steps as in equations (35)-(36). Following those steps requires (cf. [21]) that $P(\mathbf{V})$ remains fixed for any $\mathbf{V}$ such that $sort(\mathbf{V}) = \mathbf{L}$; which is true also for the considered load-balancing method ϕ_r because the method is not biased to any specific cache, i.e. ϕ_r assigns each user to one of the available caches only based on the load of the caches and independently from the cache identity. Therefore, the proof follows the same steps as for the case where there is no load balancing. ∎

The above theorem naturally implies that the performance deterioration, due to random association, scales as

$$G_r(\gamma)=\begin{cases}\Theta\left(\frac{\Lambda\log\log\Lambda}{K\log h}\right) & \text{if } K=o\left(\frac{\Lambda\log\log\Lambda}{\log h}\right)\\ \Theta\left(1\right) & \text{if } K=\Omega\left(\frac{\Lambda\log\log\Lambda}{\log h}\right),\end{cases} \tag{21}$$

as well as implies the following corollary.

Corollary 2. *In the K-user, Λ-cell heterogeneous network with random-selection load balancing, the performance deterioration due to random association, scales as $\Theta\left(\frac{\log\log\Lambda}{\log h}\right)$ when $K = \Theta(\Lambda)$, and then as K increases, this deterioration gradually reduces, and ceases to scale when $K = \Omega\left(\frac{\Lambda\log\log\Lambda}{\log h}\right)$.*

Proof: The proof is direct from (21). ∎

We can see that the above method can dramatically ameliorate the random association effect, where (for example when K is in the same order as Λ) even a small choice among $h = 2$ caches, can tilt the scaling of $G(\gamma)$, from the original $\Theta\left(\frac{\log\Lambda}{\log\log\Lambda}\right)$ to a much slower $\Theta(\log\log\Lambda)$.

[10]Removal of all these constraints naturally brings us back to the ideal user-to-cache association where each cache is associated to an equal number of users.

B. Load balancing via proximity-based cache selection

The aforementioned randomized load-balancing method, despite its substantial impact, may not apply when the choice is limited by geographical proximity. To capture this limitation, we consider the load-balancing approach ϕ_p where the set of Λ caches is divided into Λ/h disjoint groups $\left[\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{\Lambda/h}\right]$ of h caches each[11]. Once a user is associated at random, with uniform probability, to one of these groups, then we choose to associate this user to the least loaded cache from that group. The performance of this method is presented in the following result, for the limiting case of large Λ.

Theorem 5. *In the K-user, Λ-cell heterogeneous network with normalized cache size γ, where each user benefits from the least loaded cache among h neighboring caches, then the limiting optimal delay converges to*

$$\overline{T}^*_{\phi_p}(\gamma)=\begin{cases}\Theta\left(\frac{T_{min}\Lambda\log\frac{\Lambda}{h}}{hK\log\frac{\Lambda\log\frac{\Lambda}{h}}{hK}}\right) & \text{if } K\in\left[\frac{\Lambda}{h\,\mathrm{polylog}(\frac{\Lambda}{h})}, o\left(\frac{\Lambda}{h}\log\frac{\Lambda}{h}\right)\right]\\ \Theta\left(T_{min}\right) & \text{if } K=\Omega\left(\frac{\Lambda}{h}\log\frac{\Lambda}{h}\right).\end{cases} \tag{22}$$

Proof: The achievability proof is deferred to Appendix E, while the optimality part of the theorem follows the same argument as the proof of Theorem 4. ∎

The above implies a performance deterioration of

$$G_p(\gamma)=\begin{cases}\Theta\left(\frac{\frac{\Lambda}{hK}\log\frac{\Lambda}{h}}{\log\frac{\Lambda\log\frac{\Lambda}{h}}{hK}}\right) & \text{if } K\in\left[\frac{\Lambda/h}{\mathrm{polylog}(\frac{\Lambda}{h})}, o\left(\frac{\Lambda}{h}\log\frac{\Lambda}{h}\right)\right]\\ \Theta\left(1\right) & \text{if } K=\Omega\left(\frac{\Lambda}{h}\log\frac{\Lambda}{h}\right),\end{cases} \tag{23}$$

which in turn implies the following.

Corollary 3. *In the K-user, Λ-cell heterogeneous network with proximity-bounded load balancing, the performance deterioration due to random association scales as $\Theta\left(\frac{\log(\Lambda/h)}{\log\log(\Lambda/h)}\right)$ when $K = \Theta\left(\frac{\Lambda}{h}\right)$, and as K increases, this deterioration gradually reduces, and ceases to scale when $K = \Omega\left(\frac{\Lambda}{h}\log\frac{\Lambda}{h}\right)$.*

Proof: The proof is straightforward from Theorem 5. ∎

We can see that proximity-bounded load balancing significantly ameliorate the random association effect, where now deterioration ceases to scale when $K = \Omega\left(\frac{\Lambda}{h}\log\frac{\Lambda}{h}\right)$ compared to the original $K = \Omega(\Lambda\log\Lambda)$.

IV. Numerical Validation

We proceed to numerically validate our results, using two basic numerical evaluation approaches. The first is 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 $E_{\mathbf{L}}[T(\mathbf{L})]$ as

$$E_{\mathbf{L}}[T(\mathbf{L})] \approx \frac{1}{|\mathcal{L}_1|}\sum_{\mathbf{L}\in\mathcal{L}_1} T(\mathbf{L}), \tag{24}$$

where we recall that $T(\mathbf{L})$ is defined in (4). The corresponding approximate performance deterioration is then evaluated by dividing the above by T_{min}.

[11]In this method, our focus is in the asymptotic setting, thus we do not need to assume that h divides Λ.

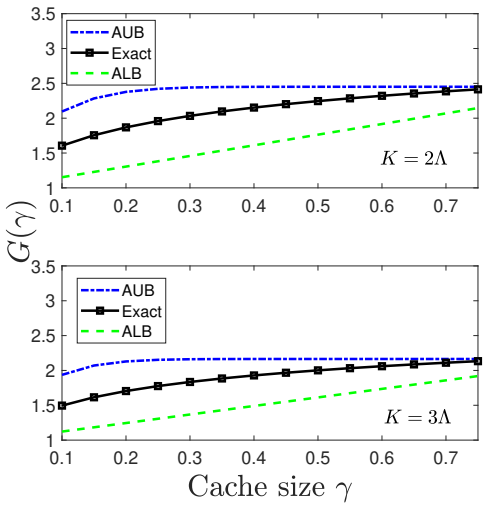

Fig. 5. Analytical upper bound (AUB) from (14) vs. analytical lower bound (ALB) from (15) vs. exact $G(\gamma)$ from (10) ($\Lambda = 20$).

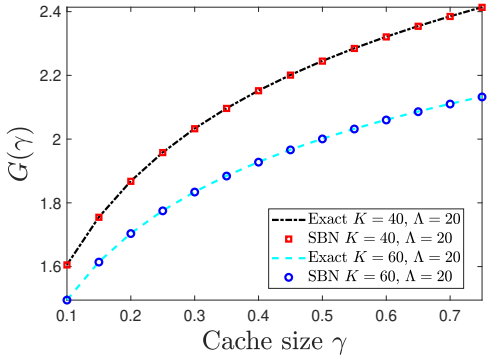

Fig. 6. Exact $G(\gamma)$ from (10) vs. sampling-based numerical (SBN) approximation from (24) ($|\mathcal{L}_1| = 10000$).

The second is a *threshold-based numerical* method, whose first step is to generate a set $\mathcal{L}_2 \subseteq \mathcal{L}$ of profile vectors $\mathbf{L}$ such that $\sum_{\mathbf{L}\in\mathcal{L}_2} P(\mathbf{L}) \approx \rho$, for some chosen threshold value $\rho \in [0, 1]$. Recall that the closed form expression for $P(\mathbf{L})$ is given in equation (33). Subsequently, with this subset $\mathcal{L}_2$ at hand, we simply have the numerical lower bound (NLB)

$$E_{\mathbf{L}}[T(\mathbf{L})] \geq \sum_{\mathbf{L}\in\mathcal{L}_2} P(\mathbf{L})T(\mathbf{L}) + (1-\rho)\,T_{min}, \qquad (25)$$

by considering the best-case delay for each $\mathbf{L} \in \mathcal{L}/\mathcal{L}_2$, and similarly have the numerical upper bound (NUB)

$$E_{\mathbf{L}}[T(\mathbf{L})] \leq \sum_{\mathbf{L}\in\mathcal{L}_2} P(\mathbf{L})T(\mathbf{L}) + (1-\rho)\,K(1-\gamma), \qquad (26)$$

by considering the worst possible delay $K(1-\gamma)$ for every $\mathbf{L} \in \mathcal{L}/\mathcal{L}_2$. The bounding of $G(\gamma)$ is direct by dividing the above with T_{min}.

Naturally the larger the threshold ρ, the tighter the bounds, the higher the computational cost. The additive gap between the bounds on $G(\gamma)$, takes the form $(1-\rho)\left(\frac{K(1-\gamma)}{T_{min}} - 1\right) \approx (1-\rho)\,t$, revealing the benefit of increasing ρ.

First, Figures 5-7 include comparisons that involve the *exact* $G(\gamma)$ from (10), and thus — due to the computational cost — the number of caches remains at a modest $\Lambda = 20$ (and a relatively larger $\Lambda = 30$ for Figure 7). In particular, Figure 5 compares the exact $G(\gamma)$ with the analytical bounds in (14) and (15), where it is clear that both AUB and ALB yield sensible bounds, and AUB becomes much tighter as γ increases. Figure 6 compares the exact $G(\gamma)$ with the sampling-based numerical (SBN) approximation in (24) (for $|\mathcal{L}_1| = 10000$), where it is evident that the SBN approximation is consistent with the exact performance. Finally, Figure 7 compares the exact $G(\gamma)$ (for $\Lambda = 30$) with the threshold-based numerical bounds that are based on (25) and (26), using $\rho = 0.95$. Interestingly, the threshold-based NLB turns out to be very tight in the entire range of γ, whereas the NUB tends to move away from the exact performance as γ increases.

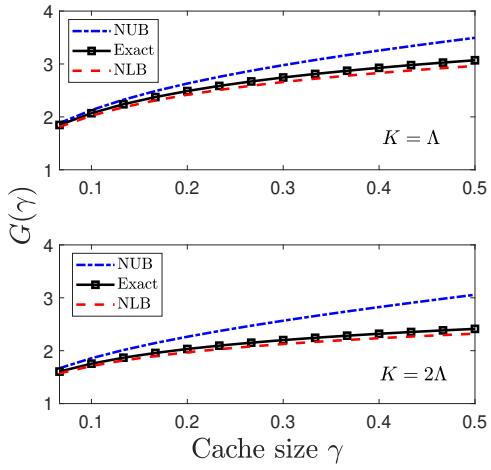

Fig. 7. Threshold-based numerical upper bound (NUB) from (26) vs. threshold-based numerical lower bound (NLB) from (25) vs. exact $G(\gamma)$ from (10) ($\Lambda = 30$ and $\rho = 0.95$).

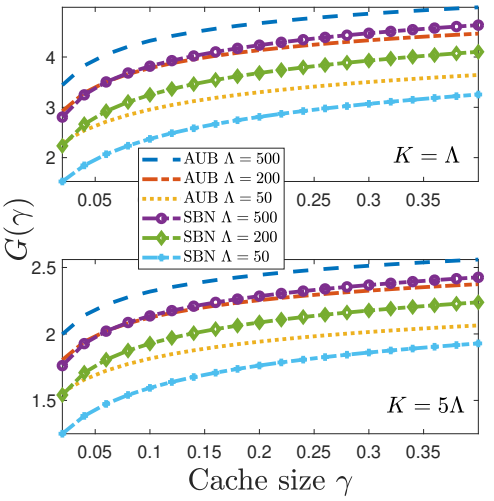

Fig. 8. Analytical upper bound (AUB) from (14) vs. sampling-based numerical (SBN) approximation from (24).

Subsequently, for much larger dimensionalities, Figure 8 compares the AUB from (14) with the SBN approximation from (24) for $|\mathcal{L}_1| = 10000$. The figure highlight the extent to which the ratio $\frac{K}{\Lambda}$ affects the performance deterioration.

Finally, Figure 9 uses a suitably modified analytical upper bound to explore the effect of h when applying proximity-bounded load balancing. We know from (40) that the expected number of users in the most populous cache group (i.e, $E[l_1^h]$), when each user can be associated to any cache group with equal probability $\frac{h}{\Lambda}$ is bounded as $E[l_1^h] \leq K - \sum_{j=0}^{K-1} \max\left(1 - \frac{\Lambda}{h}(1-P_j^h), 0\right)$, where $P_j^h = \sum_{i=0}^{j} \binom{K}{i} \left(\frac{h}{\Lambda}\right)^i \left(1-\frac{h}{\Lambda}\right)^{K-i}$. Also, from (59), the expected number of users in the most populous cache (i.e., $E[l_1]$) under proximity-bounded load balancing is bounded as $E[l_1] < \frac{E[l_1^h]}{h} + 1$. Thus using (47), the analytical upper bound on the $\overline{T}^*_{\phi_p}(\gamma)$ is given by

$$\overline{T}^*_{\phi_p}(\gamma) \leq \frac{\Lambda - t}{1+t} E[l_1] < \frac{\Lambda - t}{1+t}\left(\frac{E[l_1^h]}{h} + 1\right) = \frac{\Lambda - t}{1+t}\left(1 + \frac{K}{h} - \frac{1}{h}\sum_{j=0}^{K-1} \max\left(1 - \frac{\Lambda}{h}(1-P_j^h), 0\right)\right). \tag{27}$$

From Figure 9, we can see that, as expected, the performance deterioration decreases as h increases.

V. Extension to the Case of Non-Uniform Cache Population Intensities

In this section, we extend our study to the scenario where cache population intensities (i.e, probability that a user can appear in the coverage area of any particular cache-enabled cell) are following a non-uniform distribution[12]. For any cache $\lambda \in [\Lambda]$, let p_λ be the probability that a user can appear in the coverage area of λth cache-enabled cell such that $\mathbf{p} = [p_1, p_2, \cdots, p_\Lambda]$, where $\sum_{\lambda \in [\Lambda]} p_\lambda = 1$, denotes the cache population intensities vector.

A. Analytical Bounds

Considering the uncoded cache placement scheme in [2], and the delivery scheme in [21], the following theorem bounds the average delay $\overline{T}(\gamma)$, when cache population intensities are following a non-uniform distribution.

Theorem 6. *In the K-user, Λ-cache setting with normalized cache size γ and a random user-to-cache association with cache population intensities $\mathbf{p}$, the average delay $\overline{T}(\gamma)$ is bounded by*

$$\overline{T}(\gamma) \leq K\frac{\Lambda - t}{1+t} - \sum_{\lambda=1}^{\Lambda - t} \frac{\binom{\Lambda - \lambda}{t}}{\binom{\Lambda}{t}} \sum_{j=0}^{K-1} \max\left(0, 1 - \frac{\Lambda - F(j)}{\lambda}\right), \tag{28}$$

and

$$\overline{T}(\gamma) \geq \frac{\Lambda - t}{1+t}\left(\frac{Kt\max(\mathbf{p})}{(\Lambda - 1)} + \frac{K}{\Lambda}\frac{(\Lambda - t - 1)}{(\Lambda - 1)}\right), \tag{29}$$

where

$$F(j) = \sum_{k=1}^{\Lambda}\sum_{i=0}^{j} \binom{K}{i} (p_k)^i (1-p_k)^{K-i}. \tag{30}$$

Proof: The proof can be found in the longer version of this work [43, Appendix G]. ∎

It is fast to numerically evaluate the analytical bound proposed in Theorem 6 for any given distribution of cache population intensities $\mathbf{p}$. However, in order to gain some simple and insightful form of the performance in the presence of non-uniform cache population intensities, we proceed with the asymptotic analysis of the $\overline{T}(\gamma)$ under the assumption that cache population intensities $\mathbf{p}$ follows the Zipf distribution[13]. For the Zipf distribution, cache population intensities $\mathbf{p}$ are given by[14] $p_\lambda = \frac{\lambda^{-\alpha}}{H_\alpha(\Lambda)}, \quad \forall \ \lambda \in [\Lambda]$, where $\alpha > 0$ is the Zipf exponent, and $H_\alpha(\Lambda) = \sum_{i=1}^{\Lambda} i^{-\alpha}$ is a normalization constant formed as the generalized harmonic number.

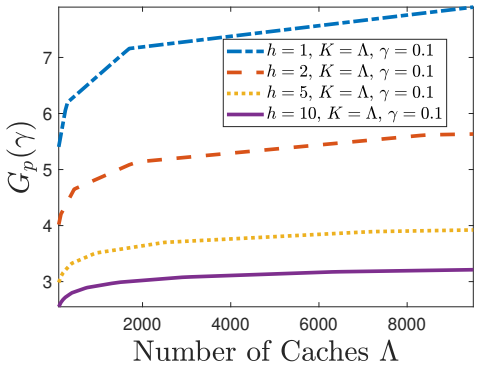

Fig. 9. Analytical upper bound (AUB) from (27) without ($h = 1$) and with ($h > 1$) proximity-bounded load balancing.

B. Scaling Laws

The following theorem provides the asymptotic analysis of the $\overline{T}(\gamma)$, in the limit of large Λ.

Theorem 7. *In the K-user, Λ-caches setting with normalized cache size γ and random user-to-cache association with cache population intensities $\mathbf{p}$ following the Zipf distribution with the Zipf exponent α, the delay scales as*

$$\overline{T}(\gamma) = \begin{cases} \Theta\left(T_{min}\Lambda\right) & \alpha > 1 \\ O\left(T_{min}\sqrt{\frac{\Lambda^2}{K} + \frac{\Lambda^2}{(\log\Lambda)^2}}\right) & \alpha = 1 \\ \quad \textit{and} \quad \Omega\left(T_{min}\frac{\Lambda}{\log\Lambda}\right) & \\ O\left(T_{min}\sqrt{\frac{\Lambda^2}{K} + \Lambda^{2\alpha}}\right) & 0.5 < \alpha < 1 \\ \quad \textit{and} \quad \Omega\left(T_{min}\Lambda^\alpha\right) & \\ O\left(T_{min}\sqrt{\frac{\Lambda^2}{K} + \Lambda\log\Lambda}\right) & \alpha = 0.5 \\ \quad \textit{and} \quad \Omega\left(T_{min}\sqrt{\Lambda}\right) & \\ O\left(T_{min}\left(\sqrt{\Lambda + \frac{\Lambda^2}{K}}\right)\right) & \alpha < 0.5. \\ \quad \textit{and} \quad \Omega\left(T_{min}\Lambda^\alpha\right) & \end{cases} \tag{31}$$

Proof: The proof can be found in the longer version of this work [43, Appendix H]. ∎

In identifying the scaling laws of the problem, Theorem 7 nicely captures the following points:

- It describes to what extent the performance deterioration

[12] All the results presented in this section are optimal for the case when the cache population intensities are not known during the cache placement phase.

[13] There are several studies that propose different user distribution models (i.e., distribution of cache population intensities) for wireless networks [51], [52]. We use the Zipf distribution as it nicely covers a wide range of non-uniform patterns by only tuning one parameter.

[14] Without loss of generality, we assume a descending order between cache population intensities of the Λ caches.

increases with α (i.e., the skewness in cache population intensities).

- It shows that, in some cases there is no global caching gain, e.g., the performance deterioration scales as $\Theta(\Lambda)$ for $\alpha > 1$.
- It reveals that unlike the case of uniform cache population intensities – where the deterioration can be avoided as long as $K = \Omega(\Lambda \log \Lambda)$ – the existence of skewness in cache population intensities can lead to an unbounded deterioration irrespective of the relation between K and Λ.
- It highlights the importance of incorporating the knowledge of the cache population intensities vector while designing the placement and delivery scheme. As pointed out earlier, being unaware of the severeness of this non-uniformity may lead to the vanishing of the coding gain, and the system may eventually need to confine itself to the local caching gain.

C. Randomized load balancing with multiple choices under non-uniform cache population intensities

We consider a load-balancing method ϕ_n which, for any given user, picks $h \geq 2$ candidate caches at random based on the cache population intensities $\mathbf{p}$ following the Zipf distribution, and then associates each such user to the least loaded cache among these h caches. The performance of this method is presented in the following result, for the limiting case of large Λ.

Theorem 8. *In the K-user, Λ-cell heterogeneous network with normalized cache size γ, where each user benefits from the least loaded cache among h randomly chosen caches based on the cache population intensities $\mathbf{p}$ with the Zipf exponent α, the limiting delay converges to*

$$\overline{T}_{\phi_n}(\gamma) = \begin{cases} O\left(T_{min} \frac{\Lambda \log\log \Lambda}{K}\right) & \text{if } K = o(\Lambda \log\log \Lambda) \\ O(T_{min}) & \text{if } K = \Omega(\Lambda \log\log \Lambda), \end{cases} \tag{32}$$

when $h = \Theta(\log \Lambda)$.

Proof: The proof can be found in the longer version of this work [43, Appendix I]. ∎

We can see that load balancing can dramatically ameliorate the random association effect. For example, when $\alpha > 1$, picking any $\log \Lambda$ candidate caches is sufficient to tilt the scaling of $G(\gamma)$ from $\Theta(\Lambda)$ to a much slower $O(\log\log \Lambda)$ and $O(1)$, when $K = \Theta(\Lambda)$ and $K = \Omega(\Lambda \log\log \Lambda)$ respectively. As long as h is in the same order as $\log \Lambda$, significant improvements can be achieved irrespective of the level of skewness of the cache population intensities. In conclusion, even for the non-uniform cache population intensities, load balancing can still be impactful. However, for non-uniform cache population intensities setting $h = 2$ may not bring significant gains which were observed for the case of uniform cache population intensities case (cf. Corollary 2) as now h must be in the order of $\log \Lambda$.

D. Numerical validation for the non-uniform cache population intensities

We now numerically validate our results for the case of non-uniform cache population intensities. For the numerical analysis, we assume that cache population intensities $\mathbf{p}$ follows the Zipf distribution. Figure 10 compares the AUB from (28) with the SBN approximation from (24) for $|\mathcal{L}_1| = 10000$. Note that $\mathcal{L}_1$ is generated based on cache population intensities $\mathbf{p}$. The figure highlights the extent to which the Zipf exponent α (i.e., the skewness in cache population intensities) affects the performance deterioration.

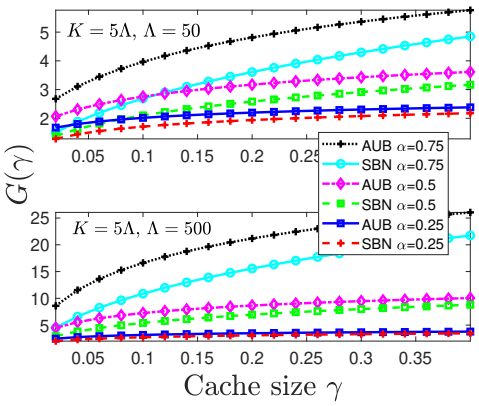

Fig. 10. Analytical upper bound (AUB) from (28) vs. sampling-based numerical (SBN) approximation from (24).

VI. Conclusions

In this work we identified the exact optimal performance of coded caching with random user-to-cache association when users can appear in the coverage area of any particular cache-enabled cell with equal probability. In our opinion, the random association problem has direct practical ramifications, as it captures promising scenarios (such as the heterogeneous network scenario) as well as operational realities (namely, the subpacketization constraint). The problem becomes even more pertinent as we now know that its effect can in fact scale indefinitely.

Key to our effort to identify the effect of association randomness, has been the need to provide expressions that can either be evaluated in a numerically tractable way, or that can be rigorously approximated in order to yield clear insight. The first part was achieved by deriving exact expressions as well as new analytical bounds that can be evaluated directly, while the second part was achieved by studying the asymptotics of the problem which yielded simple performance expressions and direct operational guidelines. This same approach also allowed us to clearly capture the effect and importance of basic load-balancing techniques that are used to mitigate the detrimental effect coming from the aforementioned randomness in user-to-cache association.

Finally, we extended our analysis for the case where cache population intensities are following a non-uniform distribution. We provided analytical bounds and studied the asymptotics of the problem. Also, we show that load balancing can help mitigating the effect of randomness even when cache population intensities are non-uniform.

Appendix

Detailed proofs of all the analytical results in this paper can be found in [43].

A. Proof of Theorem 1

We first note that the probability $P(\mathbf{L})$ of observing a specific profile vector $\mathbf{L} \in \mathcal{L}$ is simply the cumulative probability over all $\mathbf{V}$ for which $sort(\mathbf{V}) = \mathbf{L}$. This probability takes the form

$$P(\mathbf{L}) = \overbrace{\frac{1}{\Lambda^K}}^{\text{term 1}} \times \frac{K!}{\prod_{i=1}^{\Lambda} l_i!} \times \overbrace{\frac{\Lambda!}{\prod_{j=1}^{|B_L|} b_j!}}^{\text{term 2}}. \tag{33}$$

To see this, we analyze the different terms of the above equation. The first term in (33) accounts for the fact that there are Λ^K different user-to-cache associations, i.e., there are Λ^K different ways that K users can be allocated to the Λ different caches. It also accounts for the fact that each user can be associated to any one particular cache, with equal probability $\frac{1}{\Lambda}$. The second term in (33) indicates the number of all user-to-cache associations that leads[15] to a specific $\mathbf{V}$ for which $sort(\mathbf{V}) = \mathbf{L}$, for some fixed $\mathbf{L}$. Consequently term 1 in (33) is simply $P(\mathbf{V})$, which naturally remains fixed for any $\mathbf{V}$ for which $sort(\mathbf{V}) = \mathbf{L}$, and which originates from the well-known probability mass function of the multinomial distribution. Consequently this implies that $P(\mathbf{L}) = |\{\mathbf{V} : sort(\mathbf{V}) = \mathbf{L}\}| \times P(\mathbf{V})$. Finally, term 2 describes the number of all possible cache population vectors $\mathbf{V}$ for which $sort(\mathbf{V})$ is equal to some fixed $\mathbf{L}$.

We now proceed to insert (33) into (7), which yields the average delay

$$\begin{aligned} E_{\mathbf{L}}[T(\mathbf{L})] &= \sum_{\lambda=1}^{\Lambda-t} \sum_{\mathbf{L}\in\mathcal{L}} P(\mathbf{L}) l_\lambda \frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}} \\ &= \sum_{\lambda=1}^{\Lambda-t} \sum_{\mathbf{L}\in\mathcal{L}} \frac{K!\ t!\ (\Lambda-t)!\ l_\lambda \binom{\Lambda-\lambda}{t}}{\Lambda^K \prod_{i=1}^{\Lambda} l_i! \prod_{j=1}^{|\mathbf{B_L}|} b_j!}, \end{aligned} \tag{34}$$

which concludes the achievability part of the proof for the expression in Theorem 1.

Optimality of the aforementioned expression can be proved by means of the lower bound developed in [21]. We notice that the optimal delay $\overline{T}^*(\gamma)$ can be lower bounded as

$$\begin{aligned} \overline{T}^*(\gamma) &= \min_{\mathcal{X}} E_{\mathbf{L}} \left[E_{\mathbf{V_L}} \left[\max_{\mathbf{d}} T(\mathbf{V}, \mathbf{d}, \mathcal{X}) \right] \right] \\ &\geq \min_{\mathcal{X}} E_{\mathbf{L}} \left[\max_{\mathbf{d}} E_{\mathbf{V_L}} \left[T(\mathbf{V}, \mathbf{d}, \mathcal{X}) \right] \right] \\ &\geq E_{\mathbf{L}} \left[\min_{\mathcal{X}} \max_{\mathbf{d}} E_{\mathbf{V_L}} \left[T(\mathbf{V}, \mathbf{d}, \mathcal{X}) \right] \right] \\ &\geq E_{\mathbf{L}} \Big[\underbrace{\min_{\mathcal{X}} E_{\mathbf{d}\in\mathcal{D}_{wc}} E_{\mathbf{V_L}} \left[T(\mathbf{V}, \mathbf{d}, \mathcal{X}) \right]}_{T^*(\mathbf{L})} \Big], \end{aligned} \tag{35}$$

where $\mathcal{D}_{wc}$ denoted the set of demand vectors with distinct users' file-requests. Next, exploiting the fact that $P(\mathbf{V})$ is the same for any $\mathbf{V}$ for which $sort(\mathbf{V}) = \mathbf{L}$, we notice that

$$T^*(\mathbf{L}) \triangleq \min_{\mathcal{X}} E_{\mathbf{d}\in\mathcal{D}_{wc}} E_{\mathbf{V_L}} \left[T(\mathbf{V}, \mathbf{d}, \mathcal{X}) \right]$$

is lower bounded by equation (53) in [21], which then proves that $T^*(\mathbf{L})$ is bounded as

$$T^*(\mathbf{L}) \geq \sum_{\lambda=1}^{\Lambda-t} l_\lambda \frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}}. \tag{36}$$

This concludes the proof for the optimality of the delivery time in Theorem 1.

B. Proof of Theorem 2

We start our proof by deriving the expected number of users in the λ-th most populous cache (i.e., $E[l_\lambda]$), which is given by

$$\begin{aligned} E[l_\lambda] &= \sum_{j=0}^{K-1} P[l_\lambda > j] = \sum_{j=0}^{K-1} (1 - P[l_\lambda \leq j]) \\ &= K - \sum_{j=0}^{K-1} P[l_\lambda \leq j]. \end{aligned} \tag{37}$$

where $P[l_\lambda \leq j]$ is the probability that λ-th most populous cache is associated to no more than j requesting users. From [53, Proposition 2], we have

$$P[l_\lambda \leq j] \geq \max\left(1 - \frac{\Lambda}{\lambda}(1 - P_j), 0\right), \tag{38}$$

where P_j is the probability that a cache is associated to no more than j requesting users. Recalling that each user can be assigned to any particular cache with equal probability, we can conclude that P_j is given as

$$P_j = \sum_{i=0}^{j} \binom{K}{i} \left(\frac{1}{\Lambda}\right)^i \left(1 - \frac{1}{\Lambda}\right)^{K-i}. \tag{39}$$

$E[l_\lambda]$ is upper bounded by

$$E[l_\lambda] \leq K - \sum_{j=0}^{K-1} \max\left(1 - \frac{\Lambda}{\lambda}(1 - P_j), 0\right). \tag{40}$$

Consequently the upper bound of $\overline{T}^*(\gamma)$ is given as

$$\begin{aligned} \overline{T}^*(\gamma) &= \sum_{\lambda=1}^{\Lambda-t} E[l_\lambda] \frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}} \\ &\leq \sum_{\lambda=1}^{\Lambda-t} \frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}} \left(K - \sum_{j=0}^{K-1} \max\left(1 - \frac{\Lambda}{\lambda}(1 - P_j), 0\right) \right) \\ &\overset{(a)}{=} \frac{\binom{\Lambda}{t+1}}{\binom{\Lambda}{t}} K - \sum_{\lambda=1}^{\Lambda-t} \frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}} \sum_{j=0}^{K-1} \max\left(1 - \frac{\Lambda}{\lambda}(1 - P_j), 0\right), \end{aligned} \tag{41}$$

where in step (a), we used the column-sum property of Pascal's triangle, which is $\sum_{k=0}^{n} \binom{k}{t} = \binom{n+1}{t+1}$. This concludes the proof of the upper bound in (11).

Next, we prove the lower bound in (12). Crucial to this proof is the exploitation of the fact that $\sum_{\lambda=1}^{\Lambda} E[l_\lambda] = K$ and of the fact that both $E[l_\lambda]$ and $\binom{\Lambda-\lambda}{t}$ in (7) are non-increasing with λ. We first see that

$$\overline{T}^*(\gamma) = \sum_{\lambda=1}^{\Lambda-t} E[l_\lambda] \frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}} \geq \frac{E[l_1]\binom{\Lambda-1}{t} + \sum_{\lambda=2}^{\Lambda-t} B\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}}, \tag{42}$$

where $B = \frac{K - E[l_1]}{\Lambda - 1}$. This can be simplified as

$$\overline{T}^*(\gamma) \geq E[l_1] \frac{\binom{\Lambda-1}{t}}{\binom{\Lambda}{t}} + B \frac{\binom{\Lambda-1}{t+1}}{\binom{\Lambda}{t}}$$

[15] Recall that different user-to-cache associations can lead to the same cache population vector $\mathbf{V}$. For example, when $K = \Lambda = 3$, the following 6 user-to-cache associations, $[1, 2, 3]$, $[1, 3, 2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3, 2, 1]$, and $[3, 1, 2]$ — each describing which user is associated to which cache — in fact all correspond to the same $\mathbf{V} = [1, 1, 1]$, because always each cache is associated to one user.

$$= E[l_1]\frac{\Lambda-t}{\Lambda} + B\frac{(\Lambda-t)(\Lambda-t-1)}{(1+t)\Lambda}$$
$$= (\Lambda-t)\left(\frac{E[l_1]}{\Lambda} + \frac{K-E[l_1]}{\Lambda-1}\frac{\Lambda-t-1}{(1+t)\Lambda}\right)$$
$$= \frac{\Lambda-t}{1+t}\left(\frac{E[l_1]t}{\Lambda-1} + \frac{K}{\Lambda}\frac{\Lambda-t-1}{\Lambda-1}\right). \quad (43)$$

To conclude the proof, we need to derive $E[l_1]$. It is straightforward that $l_1 \geq \lceil \frac{K}{\Lambda} \rceil$, thus for $j = [0, 1, 2, \cdots, \lceil \frac{K}{\Lambda} \rceil - 1]$ we have $P[l_1 \leq j] = 0$, and for $j = [\lceil \frac{K}{\Lambda} \rceil, \lceil \frac{K}{\Lambda} \rceil + 1, \cdots, K]$, from [53, Proposition 1] we have

$$P[l_1 \leq j] \leq \min(P_j, 1) = P_j, \quad (44)$$

where P_j is defined in (39). Therefore, using (37), $E[l_1]$ is lower bounded as

$$E[l_1] = K - \sum_{j=0}^{K-1} P[l_1 \leq j] \geq K - \sum_{j=\lceil \frac{K}{\Lambda} \rceil}^{K-1} P_j. \quad (45)$$

Finally, combining (43) and (45), we obtain

$$\overline{T}^*(\gamma) \geq \frac{\Lambda-t}{1+t}\left(\frac{t}{\Lambda-1}\left(K - \sum_{j=\lceil \frac{K}{\Lambda} \rceil}^{K-1} P_j\right) + \frac{K}{\Lambda}\frac{\Lambda-t-1}{\Lambda-1}\right), \quad (46)$$

which concludes the proof of Theorem 2.

C. Proof of Theorem 3

The fact that both $E[l_\lambda]$ and $\binom{\Lambda-\lambda}{t}$ in (7) are non-increasing with λ, we see that $\overline{T}^*(\gamma)$ is bounded by

$$\overline{T}^*(\gamma) = \sum_{\lambda=1}^{\Lambda-t} E[l_\lambda]\frac{\binom{\Lambda-\lambda}{t}}{\binom{\Lambda}{t}} \leq \frac{1}{\binom{\Lambda}{t}}\sum_{\lambda=1}^{\Lambda-t} E[l_1]\binom{\Lambda-\lambda}{t}$$
$$\overset{(a)}{=} \frac{E[l_1]}{\binom{\Lambda}{t}}\sum_{\lambda=1}^{\Lambda-t}\binom{\Lambda-\lambda}{t} = E[l_1]\frac{\binom{\Lambda}{t+1}}{\binom{\Lambda}{t}}$$
$$= E[l_1]\frac{\Lambda-t}{1+t} = \frac{K(1-\gamma)}{1+t}\frac{\Lambda E[l_1]}{K}, \quad (47)$$

where in step (a), we used the column-sum property of Pascal's triangle, which is $\sum_{k=0}^{n}\binom{k}{t} = \binom{n+1}{t+1}$. Thus from (47), we get

$$\overline{T}^*(\gamma) = O\left(\frac{K(1-\gamma)}{1+t}\frac{\Lambda E[l_1]}{K}\right) \quad (48)$$

and from (43), we have

$$\overline{T}^*(\gamma) = \Omega\left(\frac{K(1-\gamma)}{1+t}\frac{\Lambda E[l_1]\gamma}{K}\right). \quad (49)$$

As γ is a constant, we can conclude that the expressions in (48) and (49) asymptotically match, and thus

$$\overline{T}^*(\gamma) = \Theta\left(\frac{K(1-\gamma)}{1+t}\frac{E[l_1]\Lambda}{K}\right). \quad (50)$$

Combining (50) and (6), we obtain

$$\overline{T}^*(\gamma) = \Theta\left(T_{min}\frac{E[l_1]\Lambda}{K}\right). \quad (51)$$

For the remaining part, which is to develop the asymptotics of $E[l_1]$, we proceed with the following lemma which is adopted and adapted here directly from the work of [54] on the *Balls into Bins problem*.

Lemma 1 ([54, Theorem 1] - adaptation)**.** *In a Λ-cell K-user setting where each user can be associated with equal probability to any of the caches, the tail of l_1 takes the form*

$$P[l_1 > k_\beta] = \begin{cases} o(1) & \text{if} \quad \beta > 1 \\ 1 - o(1) & \text{if} \quad 0 < \beta < 1, \end{cases} \quad (52)$$

for

$$k_\beta = \begin{cases} \frac{\log\Lambda}{\log\frac{\Lambda\log\Lambda}{K}}\left(1+\beta\frac{\log\log\frac{\Lambda\log\Lambda}{K}}{\log\frac{\Lambda\log\Lambda}{K}}\right) \text{ if } \frac{\Lambda}{\text{polylog}(\Lambda)} \leq K = o(\Lambda\log\Lambda) \\ \Theta(\beta\log\Lambda) \quad \text{if } K = \Theta(\Lambda\log(\Lambda)) \\ \frac{K}{\Lambda} + \beta\sqrt{\frac{K\log(\Lambda)}{0.5\Lambda}} \quad \text{if } \omega(\Lambda\log\Lambda) = K \leq \Lambda\,\text{polylog}(\Lambda) \\ \frac{K}{\Lambda} + \sqrt{\frac{K\log(\Lambda)}{0.5\Lambda}\left(1 - \frac{\log\log\Lambda}{2\beta\log\Lambda}\right)} \text{ if } K = \omega\left(\Lambda(\log\Lambda)^3\right). \end{cases} \quad (53)$$

Proof: The result comes directly from [54, Theorem 1]. ∎

With Lemma 1 at hand, we consider the case of $\beta > 1$, for which we get that

$$E[l_1] = \sum_{j=0}^{k_\beta-1} P[l_1 > j] + P[l_1 > k_\beta] + \sum_{j=k_\beta+1}^{K-1} P[l_1 > j]$$
$$\overset{(a)}{\leq} k_\beta + o(1) + \sum_{j=k_\beta+1}^{K-1} P[l_1 > j]$$
$$\overset{(b)}{\leq} k_\beta + o(1) + (K - k_\beta - 1)o(1)$$
$$= k_\beta(1 - o(1)) + Ko(1) = O(k_\beta), \quad (54)$$

where in step (a), we use the fact that $P[l_1 > j]$ is at most 1 for $j = [0, 1, \cdots k_\beta - 1]$ and in step (b), we use the fact if $P[l_1 > k_\beta] = o(1)$ then $P[l_1 > j]$ is at most $o(1)$ for $j = [k_\beta + 1, \cdots K - 1]$. Similarly, for $0 < \beta < 1$, we have

$$E[l_1] \overset{(a)}{\geq} k_\beta(1 - o(1)) + 1 - o(1) \geq k_\beta(1 - o(1)) = \Omega(k_\beta), \quad (55)$$

where in step (a), we use the fact that $\sum_{j=k_\beta+1}^{K-1} P[l_1 > j] \geq 0$ and if $P[l_1 > k_\beta] = 1 - o(1)$ then $P[l_1 > j]$ is at least $1 - o(1)$ for $j = [0, 1, \cdots k_\beta - 1]$. Combining (53), (54), and (55), we have

$$E[l_1] = \begin{cases} \Theta\left(\frac{\log\Lambda}{\log\frac{\Lambda\log\Lambda}{K}}\right) \text{ if } K \in [\frac{\Lambda}{\text{polylog}(\Lambda)}, o(\Lambda\log\Lambda)] \\ \Theta\left(\frac{K}{\Lambda} + \sqrt{\frac{K\log(\Lambda)}{\Lambda}}\right) \text{ if } K = \Omega(\Lambda\log\Lambda). \end{cases} \quad (56)$$

Combining (51) with (56), allows us to directly conclude the proof of Theorem 3.

D. Proof of Theorem 4

Directly from the result in [50, Corollary 1.4] on the *Balanced Allocations* problem, we can conclude that for $h > 1$, the $E[l_1]$ asymptotically converges to

$$E[l_1] = \frac{\log\log\Lambda}{\log h} + \frac{K}{\Lambda} \pm \Theta(1). \quad (57)$$

Consequently combining (51) and (57), directly yields (20) which concludes the proof of Theorem 4.

E. Proof of Theorem 5

We start our proof by deriving the expected number of users in the most populous cache (i.e., $E[l_1]$). Recall that under the proximity-based load-balancing technique, each user can be associated to any cache *group* with equal probability $\frac{h}{\Lambda}$. Once a user is associated to a group, then this user will be associated to the least loaded cache from that group. Let l_1^h be

the number of users that are associated to the most populous group of caches, then the number of users in the most populous cache is given by $l_1 = \left\lceil \frac{l_1^h}{h} \right\rceil$. Thus, we have

$$E[l_1] = \sum_{i=1}^{K} P[l_1^h = i] \left\lceil \frac{l_1^h}{h} \right\rceil, \tag{58}$$

where $P[l_1^h = i]$ is the probability that i users are associated to the most populous group of caches. Let $S_1 \subseteq [K]$ be the set of elements such that for each element $i \in S_1$, $\frac{i}{h}$ is integer. Then, we have

$$\begin{aligned} E[l_1] &= \sum_{i \in S_1} P[l_1^h = i] \frac{l_1^h}{h} + \sum_{i \in S_1/[K]} P[l_1^h = i] \left\lceil \frac{l_1^h}{h} \right\rceil \\ &= \frac{E[l_1^h]}{h} + \sum_{i \in S_1/[K]} P[l_1^h = i] \left(\left\lceil \frac{l_1^h}{h} \right\rceil - \frac{l_1^h}{h} \right). \end{aligned} \tag{59}$$

It is straightforward to see that $0 < \sum_{i \in S_1/[K]} P[l_1^h = i] \left(\left\lceil \frac{l_1^h}{h} \right\rceil - \frac{l_1^h}{h} \right) < 1$, Therefore, $E[l_1]$ is bounded as $\frac{E[l_1^h]}{h} < E[l_1] < \frac{E[l_1^h]}{h} + 1$, and we can conclude that

$$E[l_1] = \Theta\left(\frac{E[l_1^h]}{h} \right). \tag{60}$$

Evaluating $E[l_1^h]$ from (56) by treating each group as a single cache, we conclude that

$$E[l_1] = \begin{cases} \Theta\left(\frac{\log \frac{\Lambda}{h}}{h \log \frac{\Lambda \log \frac{\Lambda}{h}}{hK}} \right) \text{ if } K \in \left[\frac{\Lambda/h}{\text{polylog}(\frac{\Lambda}{h})}, o\left(\frac{\Lambda}{h} \log \frac{\Lambda}{h}\right) \right] \\ \Theta\left(\frac{K}{\Lambda} + \sqrt{\frac{K \log(\frac{\Lambda}{h})}{h\Lambda}} \right) \text{ if } K = \Omega\left(\frac{\Lambda}{h} \log \frac{\Lambda}{h}\right). \end{cases} \tag{61}$$

Finally combining (51) and (61), directly yields (22), and thus concludes the proof of Theorem 5.

REFERENCES

[1] 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.

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

[3] 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.

[4] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 349–366, Jan 2018.

[5] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "On the average performance of caching and coded multicasting with random demands," in *Proc. 11th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Barcelona, Aug. 2014, pp. 922–926.

[6] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Characterizing the rate-memory tradeoff in cache networks within a factor of 2," *IEEE Trans. Inf. Theory*, vol. 65, no. 1, pp. 647–663, Jan 2019.

[7] K. Wan, D. Tuninetti, and P. Piantanida, "An index coding approach to caching with uncoded cache placement," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1318–1332, 2020.

[8] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Trans. Inf. Theory*, vol. 64, no. 2, pp. 1281–1296, 2018.

[9] S. S. Bidokhti, M. Wigger, and R. Timo, "Erasure broadcast networks with receiver caching," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Jul. 2016, pp. 1819–1823.

[10] J. Zhang and P. Elia, "Wireless coded caching: A topological perspective," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Jun. 2017, pp. 401–405.

[11] E. Lampiris, J. Zhang, and P. Elia, "Cache-aided cooperation with no CSIT," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Jun. 2017, pp. 2960–2964.

[12] 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.

[13] J. Zhang and P. Elia, "Fundamental limits of cache-aided wireless BC: Interplay of Coded-Caching and CSIT feedback," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 3142–3160, May 2017.

[14] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-server Coded Caching," *IEEE Trans. Inf. Theory*, vol. 62, no. 12, pp. 7253–7271, 2016.

[15] A. Tölli, S. P. Shariatpanahi, J. Kaleva, and B. H. Khalaj, "Multi-antenna interference management for coded caching," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2091–2106, 2020.

[16] S. Shariatpanahi and B. H. Khalaj, "On multi-server Coded Caching in the low memory regime," *arXiv preprint arXiv:1803.07655*, 2018.

[17] J. Zhang, F. Engelmann, and P. Elia, "Coded caching for reducing CSIT-feedback in wireless communications," in *2015 53rd Annual Allerton Conf. on Commun., Cont., and Comput. (Allerton)*, 2015, pp. 1099–1105.

[18] E. Piovano, H. Joudeh, and B. Clerckx, "On Coded Caching in the overloaded MISO Broadcast Channel," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 2795–2799.

[19] Y. Cao and M. Tao, "Treating content delivery in multi-antenna coded caching as general message sets transmission: A DoF region perspective," *IEEE Trans. Wireless Commun.*, vol. 18, no. 6, pp. 3129–3141, Jun. 2019.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] Y. Cao, M. Tao, F. Xu, and K. Liu, "Fundamental storage-latency tradeoff in cache-aided mimo interference networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5061–5076, Aug. 2017.

[25] 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.

[26] 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.

[27] E. Lampiris and P. Elia, "Achieving full multiplexing and unbounded caching gains with bounded feedback resources," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, Jun. 2018, pp. 1440–1444.

[28] N. S. Karat, S. Dey, A. Thomas, and B. S. Rajan, "An optimal linear error correcting delivery scheme for coded caching with shared caches," 2019. [Online]. Available: http://arxiv.org/abs/1901.03188

[29] K. Wan, D. Tuninetti, M. Ji, and G. Caire, "On the fundamental limits of fog-ran cache-aided networks with downlink and sidelink communications," *IEEE Transactions on Information Theory*, pp. 1–1, 2021.

[30] T. X. Vu, S. Chatzinotas, and B. Ottersten, "Coded caching and storage planning in heterogeneous networks," in *Proc. IEEE Wireless Commun. and Netw. Conf. (WCNC)*, San Francisco, CA, Mar. 2017, pp. 1–6.

[31] E. Parrinello and P. Elia, "Coded caching with optimized shared-cache sizes," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Visby, Sweden, Aug. 2019, pp. 1–5.

[32] A. Sengupta and R. Tandon, "Improved approximation of storage-rate tradeoff for caching with multiple demands," *IEEE Trans. Commun.*, vol. 65, no. 5, pp. 1940–1955, May 2017.

[33] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis, "Finite-length analysis of caching-aided coded multicasting," *IEEE Trans. Inf. Theory*, vol. 62, no. 10, pp. 5524–5537, Oct. 2016.

[34] L. Tang and A. Ramamoorthy, "Low subpacketization schemes for coded caching," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Jun. 2017, pp. 2790–2794.

[35] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," *IEEE Trans. Inf. Theory*, vol. 64, no. 8, pp. 5755–5766, Aug. 2018.

[36] K. Shanmugam, A. M. Tulino, and A. G. Dimakis, "Coded caching with linear subpacketization is possible using Ruzsa-Szemeredi graphs,"

in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Jun. 2017, pp. 1237–1241.
[37] S. Jin, Y. Cui, H. Liu, and G. Caire, "A new order-optimal decentralized coded caching scheme with good performance in the finite file size regime," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5297–5310, Aug. 2019.
[38] J. G. Andrews, S. Singh, Q. Ye, X. Lin, and H. S. Dhillon, "An overview of load balancing in hetnets: old myths and open problems," *IEEE Wireless Commun.*, vol. 21, no. 2, pp. 18–25, 2014.
[39] Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, and J. G. Andrews, "User association for load balancing in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 6, pp. 2706–2716, 2013.
[40] H. Kim, G. de Veciana, X. Yang, and M. Venkatachalam, "Distributed α-optimal user association and cell load balancing in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, p. 177–190, Feb. 2012. [Online]. Available: https://doi.org/10.1109/TNET.2011.2157937
[41] H. Jo, Y. J. Sang, P. Xia, and J. G. Andrews, "Heterogeneous cellular networks with flexible cell association: A comprehensive downlink sinr analysis," *IEEE Trans. Wireless Commun.*, vol. 11, no. 10, pp. 3484–3495, 2012.
[42] T. Han and N. Ansari, "Network utility aware traffic load balancing in backhaul-constrained cache-enabled small cell networks with hybrid power supplies," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2819–2832, 2017.
[43] A. Malik, B. Serbetci, E. Parrinello, and P. Elia, "Fundamental limits of stochastic shared caches networks," 2020. [Online]. Available: https://arxiv.org/pdf/2005.13847.pdf
[44] "*NIST Digital Library of Mathematical Functions*," http://dlmf.nist.gov/, Release 1.0.26 of 2020-03-15, f. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds. [Online]. Available: http://dlmf.nist.gov/26.9.iv
[45] I. Stojmenović and A. Zoghbi, "Fast algorithms for generating integer partitions," *Int. J. Comput. Math.*, vol. 70, no. 2, pp. 319–332, 1998.
[46] V. Vajnovszki, "Generating permutations with a given major index," 2013. [Online]. Available: https://arxiv.org/pdf/1302.6558.pdf
[47] M. Merca, "Fast algorithm for generating ascending compositions," *J. Math. Model. and Algorithms*, vol. 11, pp. 89–104, 2012.
[48] J. Kelleher and B. O'Sullivan, "Generating all partitions: A comparison of two encodings," 2009. [Online]. Available: http://arxiv.org/abs/0909.2331
[49] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1094–1104, 2001.
[50] B. Petra, C. Artur, S. Angelika, and V. Berthold, "Balanced allocations: The heavily loaded case," *SIAM J. Comput.*, vol. 35, no. 6, pp. 1350–1385, Jun. 2006.
[51] C. Li, A. Yongacoglu, and C. D'Amours, "Heterogeneous cellular network user distribution model," in *Proc. IEEE Latin-American Conference on Commun. (LATINCOM)*, Medellin, Nov. 2016, pp. 1–6.
[52] G. George, A. Lozano, and M. Haenggi, "Distribution of the number of users per base station in cellular networks," *IEEE Wireless Commun. Letters*, vol. 8, no. 2, pp. 520–523, Apr. 2019.
[53] G. Caraux and O. Gascuel, "Bounds on distribution functions of order statistics for dependent variates," *Statist. Probab. Lett.*, vol. 14, no. 2, pp. 103–105, May 1992.
[54] R. Martin and S. Angelika, "Balls into bins — A simple and tight analysis," in *Proc. Int. Workshop Randomization Approx. Techn. Comput. Sci.*, 1998, pp. 159–170.

Adeel Malik received the B.S. degree in Electrical (Telecommunication) Engineering from the COMSATS Institute of Information and Technology, Pakistan, in 2013. During 2014–2016, he worked as a research assistant with Dr. Jalaluddin Qureshi on Namal College funded research projects focusing on the construction of wireless transmission protocols. In 2018, he graduated with an M.Sc. in Computer Science and Engineering from Dankook University, South Korea. Currently, he is working at EURECOM's Duality project as a PhD student under the supervision of Prof. Petros Elia. His research focuses on content-centric wireless networks.

Berksan Serbetci received the B.Sc. degree in Electrical and Electronics Engineering from Middle East Technical University in 2009, the M.Sc. degree in Electrical and Electronics Engineering from Bogazici University in 2012, and the Ph.D. degree in Applied Mathematics from the University of Twente in 2018. He is currently a postdoctoral fellow with the Communication Systems Department, EURECOM. His research interests include caching, wireless networks, optimization theory, stochastic processes, stochastic geometry, information theory and machine learning.

Emanuele Parrinello received the B.Sc. degree in telecommunication engineering from the Politecnico di Torino in 2015, the M.Sc. degree (Hons.) in communications and computer networks engineering from the Politecnico di Torino in 2018, and the M.Sc. degree in mobile communications from EURECOM, Telecom ParisTech, in 2018. He is currently pursuing the Ph.D. degree with the Communication Systems Department, EURECOM, Sorbonne University. His research interests lie in caching networks, network information theory, and wireless communication.

Petros Elia received the B.Sc. degree from the Illinois Institute of Technology, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Southern California (USC), Los Angeles, in 2001 and 2006 respectively. He is now a professor with the Department of Communication Systems at EURECOM in Sophia Antipolis, France. His latest research deals with the intersection of coded caching and feedback-aided communications in multiuser settings. He has also worked in the area of complexity-constrained communications, MIMO, queueing theory and cross-layer design, coding theory, information theoretic limits in cooperative communications, and surveillance networks. He is a Fulbright scholar, the co-recipient of the NEWCOM++ distinguished achievement award 2008-2011 for a sequence of publications on the topic of complexity in wireless communications, and the recipient of the ERC Consolidator Grant 2017-2022 on cache-aided wireless communications.