This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

Management of Caching Policies and Redundancy over Unreliable Channels

Paulo Sena, Antonio Abelem, György Dán, Daniel Sadoc Menasché

***Abstract*—Caching plays a central role in networked systems, reducing the load on servers and the delay experienced by users. Despite their relevance, networked caching systems still pose a number of challenges pertaining their long term behavior. In this paper, we formally show and experimentally evidence conditions under which networked caches tend to synchronize over time. Such synchronization, in turn, leads to performance degradation and aging, motivating the monitoring of caching systems for eventual rejuvenation, as well as the deployment of diverse cache replacement policies across caches to promote diversity and preclude synchronization and its aging effects. Based on trace-driven simulations with real workloads, we show how hit probability is sensitive to varying channel reliability, cache sizes, and cache separation, indicating that the mix of simple policies, such as Least Recently Used (LRU) and Least Frequently Used (LFU), provide competitive performance against state-of-art policies. Indeed, our results suggest that diversity in cache replacement policies, rejuvenation and intentional dropping of requests are strategies that build diversity across caches, preventing or mitigating performance degradation due to caching aging.**

***Index Terms*—caching, networking, wireless, aging**

I. Introduction

CACHING is a fundamental technique for scaling the service capacity of networked systems [1]–[3]. By bringing content closer to users, caches reduce the service latency for end users and decrease the load in the network and at content servers, also known as content custodians. Users can access the caches either through reliable wired channels or through unreliable wireless channels, and a single user may be able to retrieve content from one or several caches, depending on the system topology and design choices [4].

Owing to their importance, several cache replacement policies have been proposed in the past, each one with distinct advantages. For example, the Least Recently Used (LRU) policy promotes file recency and optimizes performance under adversarial finite request sequences [5, Theorem 3.6]. Similarly, the Least Frequently Used (LFU) policy maximizes the hit ratio under stationary sequences by promoting contents with the highest request frequency [5, Table 3.1], [6], while Time-To-Live (TTL) policies are versatile policies that use timers to adjust the hit probability of each content, being able to approximate other policies such as LRU and FIFO [7].

A cache replacement policy determines how cache space is allocated, as a function of the requests for contents arriving to the cache. We refer to such an allocation as *intra-cache* space split, which may be dynamic, such as in LRU, LFU and TTL, or static, such as in Segmented LRU (SLRU) [8]. In addition, a system can have either a single cache or multiple caches. When the system has a single cache, it is called *cache pooling* as all the storage resources are concentrated in one unit [9]. If the cache space is divided across multiple physically separate caches, it is called *cache partitioning*. If a request cannot be served with the desired content from the cache(s), the system will forward the request to a content server. The flow of the resulting requests is referred to as *backhaul traffic*. Under cache partitioning, the cache space must be allocated across caches, which we refer to as *inter-cache* space separation. In wireless networks with unreliable channels, in particular, separating space across multiple caches can build robustness into the system, as the availability of each of the caches may vary over time [9].

Separating cache space among multiple caches can improve robustness, but it also introduces a new issue: synchronization. In the most extreme case, all caches may eventually store the same contents, which builds robustness against losses but may be suboptimal depending on the system setup. To solve this problem, promoting strategies to increase diversity among caches may be necessary. In essence, those strategies can be reactive, e.g., consisting of periodically revisiting content allocation across caches, or proactive, e.g., avoiding synchronization at first place.

The key to the effective use of multiple caches in networks with unreliable channels is to make the caches complement each other at the right level. An excessive amount of redundancy within the cache system can undermine diversity, preventing the caches from effectively complementing each other. On the contrary, an excessive amount of diversity can penalize redundancy, limiting the network's ability to fully capitalize on the advantages of using different caches in the event of channel failure. If a channel fails, the request is usually forwarded to another cache, but if the caches store significantly different contents, there is a high chance that the request will not be satisfied, leading to an increase in the cache miss rate of the network.

In summary, the performance of a caching system is affected by a number of factors, including (a) choice of content insertion and eviction policies used by the caches, (b) separation of cache space among different contents or flows, inside each cache and across caches, as well as by (c) networking aspects, such as network congestion and reliability. These factors have been individually analyzed in previous work, but their interplay is non-trivial and is not yet well understood.

P. Sena and A. Abelem are with the Dept. of Computer Science, UFPA, Brazil. G. Dán is with Dept. of Computer Science, EECS, KTH, Stockholm, Sweden. Daniel Sadoc Menasché is with the Institute of Computing, UFRJ, Rio de Janeiro, Brazil.

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

In particular, the role of cache replacement policies and cache separation over unreliable channels has not been previously investigated.

In this paper, we evaluate the roles that cache replacement policies and cache separation over unreliable channels play in determining performance, and find that cache replacement policies and cache separation are intrinsically related, and that their roles must be taken into account jointly. The study of the LRU cache replacement policy is an important first step towards understanding the interplay between networking and caching [9]. Nonetheless, we show that if system designers have the flexibility to choose the cache replacement policy and cache separation policy then cache replacement policies other than LRU may be preferable.

We summarize our contributions as follows:

- *Analytical results on the tradeoff between robustness and diversity accounting for content synchronization.* We provide an analytical characterization of the tradeoff between robustness to channel unreliability and content diversity among caches. Using the model, we quantify the extent to which an increase in channel reliability may decrease the cache hit probability due to a reduction in content diversity across caches, accounting for multiple replacement policies. In addition, we identify conditions under which caches initialized at different states eventually converge to the same state, where robustness is maximal and diversity is minimal (Section IV).
- *Identification of different sources of content diversity.* Motivated by the fact that content diversity across caches can increase hit probability, we identify two sources of content diversity, namely: 1) channel unreliability, which causes randomization of the request streams reaching different caches; and 2) deployment of distinct cache replacement policies across caches and/or randomized policies, both effective in breaking the symmetry of contents evicted from different caches (Sections IV and V).
- *Evaluation of the role of cache replacement policies and resource separation with real workloads.* We evaluate the impact of cache replacement policies and resource separation, with real workloads. Our trace based simulations validate the analytical results, indicating that our analytical results, e.g., pertaining the decrease in hit probability as channel reliability grows, and the advantage of deploying different policies across caches, hold for real traces, accounting for temporal locality and state-of-the-art cache replacement policies (Sections VII and VIII).

Each of the above three contributions, by itself, advances the state of the art. Indeed, we are unaware of previous work accounting for multiple replacement policies and quantifying the extent at which an increase in channel reliability may decrease hit probability. Despite the fact that channel diversity is a common theme in wireless networks, we are also unaware of previous work that considered the interplay between caching and channel diversity, accounting for potential content synchronization across caches under multiple replacement policies.

The rest of the paper is organized as follows. Section II reviews related work and Section III introduces the system under study. Section IV reports analytical results on the tradeoff between robustness and diversity, and Section V presents an analytical toy model which sheds further insights on the role of partitioning and cache policies over unreliable channels, followed by a discussion of our key assumptions in Section VI. Numerical evaluation accounting for Independent Reference Model (IRM) traffic under a Zipf workload is reported in Section VII. Section VIII accounts for real traffic, Section IX presents practical implications of our results and Section X concludes.

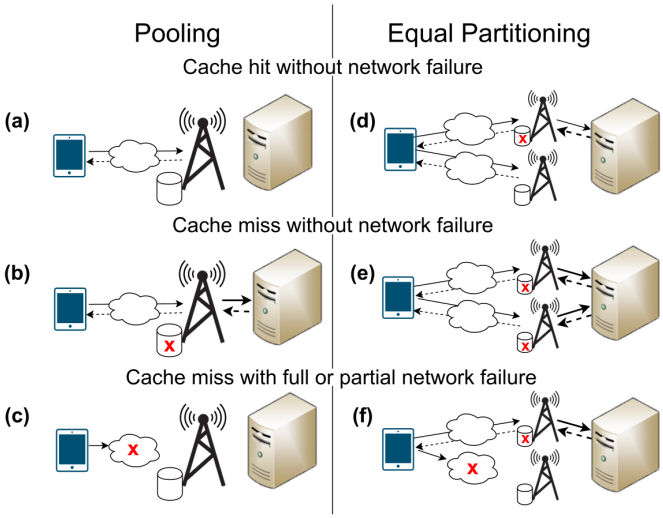

Fig. 1. Cache pooling and cache partitioning (resource separation). Cache pooling is depicted under (a) cache hit, (b) cache miss without network failure and (c) network failure cases. Cache under equal partitioning is depicted in scenarios involving backhaul traffic: (d) request broadcasted to two caches, followed by a hit and backhaul traffic to one of the two caches, (e) misses and backhaul traffic to both cache and (f) partial network failure, miss and backhaul traffic to a single cache. In this paper, misses depicted in the two bottom rows are assumed to be equivalent from the user perspective, and are referred to simply as *cache misses*.

II. Related work

In this section we discuss related literature on caching systems [5], [10], accounting for utility maximization [7], [11], caching networks [12], [13] and cache replication over unreliable channels, where each user is connected to multiple caches [14], [15], through channels with failures and losses [9], [16] and delays [17]–[19]. In what follows, we briefly review related work pertaining three main research threads related to our work, namely cache partitioning, caching over unreliable channels, and the tradeoff between parallelism and diversity.

Despite the vast literature on caching systems, the interplay between cache partitioning and cache replacement policies over unreliable channels is still poorly understood. Among our technical contributions on top of related work, we emphasize: i) an investigation of the phenomenon of cache synchronization, that is at the core of cache networks, significantly impacting its performance but which, to the best of our knowledge, has not been discussed in previous works [9], ii) a comparison between multiple cache replacement policies, which is beyond the scope of previous works on caching over unreliable channels [20] that focused exclusively on

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

LRU [9], iii) an assessment of the tradeoff between replication and diversity leveraging real workloads and a state-of-the-art synthetic workload generator [21].

A. Cache Partitioning, Cache Networks and Aging

The literature on cache partitioning is mostly comprised of works focusing on its practical aspects [22] and on partitioning among multiple flows [7], [23]. Recently, there has been significant progress on the formal analysis of strategies for cache partitioning [9], [24], [25]. While [7], [22], [23] indicate clear advantages of partitioning considering policies such as TTL, Adaptive Replacement Cache (ARC), SLRU and Sequential Prefetching ARC (SARC), other works accounting for lossy channels [9], [24], [25] focused on LRU systems. The purpose of this work is to bridge that gap, investigating how inter-cache partitioning impacts performance under policies beyond LRU, accounting for lossy broadcast channels.

As the name suggests, TTL caches account for the aging of contents, i.e., *content aging*, to determine what to evict [26]. Nonetheless, we are unaware of previous work that accounted for *caching aging* in networked systems. In networked systems, as indicated in this paper, aging occurs due to the tendency of caches to synchronize among themselves, and is orthogonal to the aging of the contents.

In [27] the authors observe that response time can increase as the number of caches grows, given the system overhead to maintain the caches. In our work, we observe a similar trend of response time increasing with respect to the number of caches, due to an eventual synchronization among them. Although our observations are in agreement with [27], and also relate to aging, the theme of cache synchronization is out of scope of [27].

Under cache networks, the benefits of randomized caching policies and of the use of distinct policies across caches have been discussed in [28]–[30]. Such previous works considered cache hierarchies, and the use of distinct policies at different levels of the hierarchies, e.g., in the context of Content Delivery Networks (CDNs). In this work, we indicate that the benefits of randomization and diversity also hold in the realm of lossy broadcast channels.

B. Caching Over Unreliable Channels

Caching over wireless networks has been considered under the framework of femtocaching [4], [31]–[33], wherein base stations are equipped with caching capabilities. Alternatively, set-top boxes equipped with caches close to users can be accessed through wifi, being managed partially by the network providers [34]. In both cases, the allocation and splitting of cache space among caches strategically placed close to users, as considered in this paper, are key elements.

In [20] the authors consider the joint problem of optimal content placement and routing, extending [35] to account for lossy channels. Our work differs from [20] in at least three aspects, as 1) [20] consider a multi-hop unicast network, while we consider a single-hop broadcast channel; 2) [20] analyze their proposed policy, together with LRU, FIFO and Random, while we consider a broader class of policies, including LFU, Bloom-filter LRU (BLRU), LRU with a threshold K (LRUK), LFU with Dynamic Allocation (LFUDA), among others; 3) we analyze the loss of efficiency caused by dependencies across caches when channels are reliable, suggesting potential benefits of diversity caused by lossy channels (see Section IV).

C. Replication and Diversity Tradeoff

Replicating content across multiple caches is a form of redundancy. Redundancy, in turn, builds robustness but decreases diversity.

Replication and diversity play a key role in computer systems. The tradeoff between replication and diversity has been considered, for instance, in the context of distributed systems [36], [37]. In those systems, jobs are assigned to multiple servers, and one may either submit the same job to multiple servers (replication, e.g., to decrease the response time of that particular job, assuming that a job gets completed when at least one of the servers executes its task) or different jobs are assigned to different servers (diversity, e.g., to decrease the overall system response time). In this paper, we show that such tradeoff also applies to caching systems, where spreading diverse contents among caches minimizes the cache miss probability assuming that the network delivers the content requests, whereas replication of the same content across multiple caches increases robustness against lossy channels.

In the realm of in-network cache provisioning, the tradeoff between network performance and storage cost has been addressed in [38]. In this work, we also account for tradeoffs involved in in-network cache allocation. However, while in [38] the authors consider a multi-hop network with reliable links, in this paper we focus on the network edge and account for lossy links.

It has been previously observed that a number of natural [39] and computational systems [40] have a tendency towards synchronization [41], leading to cache aging. In this paper, we show that networked caches also present such behavior. This, in turn, suggests that adding randomization to the system, either through the network (in the form of losses) or through the cache replacement policy (in the form randomized evictions) can increase hit probability, motivating our formal and experimental results.

This manuscript is an extended version of [42]. Among the contributions on top of [42] we emphasize (i) an enhanced analytical model including novel propositions concerning the impact of cache partitioning (Section IV); (ii) additional numerical results under synthetic workloads (Section VII); (iii) evaluation of our results accounting for real traffic, for a wide range of state-of-the-art cache replacement policies and large cache sizes (Section VIII) and (iv) an improved characterization of the impact of dependencies on system performance as well as additional numerical insights from a toy example (see Supplementary Material).

III. System Model

We consider a flow of requests issued by a set of users towards a set of caches. Each request is broadcast to all caches through replicas sent across independent channels. Channels

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

TABLE I
TABLE OF NOTATION

Variable	Description
M	Number of caches
N	Number of contents in catalog
x	Total cache space
b_i	Fraction of space allocated to cache i
p	Channel success probability

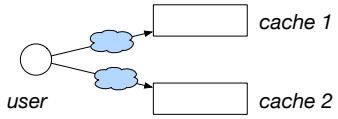

Fig. 2. System setup with $M = 2$ caches and equal split $b_1 = b_2 = 0.5$.

are unreliable, and the request may reach only a subset of the caches, which have no coordination among themselves, i.e., the control plane acts between users and caches but not across caches. Such system design is typically considered in the realm of femtocaches, and is depicted in Figures 1 and 2.

In particular, after a request reaches all caches and leads to hits in a situation where all caches have the content, the content metadata in all caches (popularity for LFU, recency for LRU, etc.) is locally updated. When a miss followed by an eviction occurs, such eviction is treated locally independently of the states of other caches. In Figures 1(a) and 1(d) we have cache hits without network failures. In Figures 1(b) and 1(e), in contrast, we have cache misses without network failures. Finally, in Figures 1(c) and 1(f) we have cache misses with full and partial network failures, where the latter corresponds to a system that has two channels, with one of them failing to deliver the request to the cache.

We denote by x the total cache size, measured in content items, when considering equal sized contents, or bytes, when considering contents with different sizes. Let M be the number of caches. In the case of pooling there is a single cache, $M = 1$. In the case of separation, $M \geq 2$, a fraction b_i of the cache space is allocated to cache i, $i \in \{1, \ldots, M\}$, as illustrated in Figure 2 for the case $M = 2$ and $b_1 = b_2 = 0.5$.

Let $F = \{f_1, ..., f_N\}$ be the set of unique items of content that can be requested. Let $V = \{v_1, \ldots, v_M\}$ be the set of M caches, $M = |V|$. For all $f_i \in F$, let λ_i be the probability that an upcoming request is issued for content f_i, i.e., λ_i is the popularity of f_i.

Definition 1. *The state of a cache system is the collection of cache states, where each cache state is a vector of stored contents in order of recency.*

Although the caches operate locally and independently, their states may be coupled through the request stream, which is common across caches. Indeed, a request that reaches multiple caches naturally leads to a coupling of the cache states, which may eventually synchronize themselves causing aging and degrading its performance. Before we present our results, we introduce additional definitions, adopting the terminology from [43].

We begin by introducing the notion of a positive request stream. We refer to a request stream as positive when the stream is comprised of independent and identically distributed requests and an upcoming request can be issued for any content in the catalog.

Definition 2. *A request stream is said to be positive if and only if its requests are independent and identically distributed and its domain is the set of all contents, i.e.,* $\forall f_n \in F$, $\lambda_n > 0$.

Then, we introduce the notion of an ergodic cache. We say that a cache is ergodic if, from each of its states, the cache can eventually transition to every other state.

Definition 3. *A cache is ergodic if each of its states can be reached from every other state given it is subject to a positive request stream.*

Finally, we introduce the notion of an individually ergodic cache system.

Definition 4. *A cache system is individually ergodic if its constituent caches are ergodic in isolation. This means that, for each cache* $v \in V$*, if* v *operates as a cache in isolation then* v *is ergodic.*

In the following section, we leverage the above definitions to characterize the system steady state behavior, and to establish conditions under which the caches eventually synchronize.

IV. Effects of Partitioning on Cache Synchronization, Aging and Performance

We begin this section by assessing the impact of initial conditions on steady state hit probability. Despite the fact that all previous works in the realm of caches connected through wireless channels assumed that initial conditions had no impact on steady state, we show through simple examples that steady state performance can be significantly affected by initial conditions (Section IV-A). Then, we consider the opposite case, and establish conditions under which independently of the initial conditions the system will converge to a state where all caches are synchronized (Section IV-B). Such a state corresponds to a low hit probability, motivating strategies to avoid it by diversifying contents across caches. In particular, we show that deploying distinct cache policies across caches can increase hit probability (Section IV-C).

A. Initial Condition Can Impact Steady State

Motivation and goals: In what follows, we aim at identifying factors that impact the steady state of content occupancy. Previous works study steady state behavior accounting for storage and link capacities, as well as network topology, but ignored the impact of the initial system configuration on steady state. The fact that the initial state is ignored in all previous works on caches connected through broadcast channels reflects the tacit assumption that after a transient phase the initial state has no impact on performance. Nonetheless, we will illustrate through simple examples that this assumption does not always hold. From an analytical perspective, this suggests the need for deeper investigation of the fundamentals behind caches connected through broadcast channels. From a practical standpoint, our results motivate strategies such as *rejuvenation* to "warmup" the caches on a regular basis (see Section IX).

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

Examples: We present four examples of caching systems in which the steady state depends on the initial condition. The first two examples involve request sequences characterized by recurring patterns over time. These sequences are realizations, for instance, of a non-positive request stream (see Definition 2). The third and fourth examples involve a cache system that lacks individual ergodicity, as described in Definition 4. Specifically, we consider a FIFO cache system with a small catalog relative to the cache size. Throughout this subsection, our focus is on a scenario involving two caches. We denote by $\{(f_{n_1} f_{n_2}), (f_{n_1'} f_{n_2'})\}$ the contents of these caches, i.e., the cache states. In this notation, position 1 corresponds to the most recent item in the cache. Additionally, we use $(f_{n_1}, \ldots, f_{n_k})$ to denote a sequence of requests and $(f_{n_1} | f_{n_1\prime}, \ldots, f_{n_k})$ to represent a sequence of requests where the first request is selected uniformly at random for f_{n_1} or $f_{n_1'}$.

1) Example 1 (non-positive request stream with FIFO over reliable channels): Let $p = 1$, $M = 2$, $x = 4$, $b_i = 0.5$ and $F = \{A, ..., Z\}$. The initial cache contents are $\{(AB), (AC)\}$ (in general, caches have the same item in position 1 and a different item in position 2). Let the first request be $(B|C)$, i.e., an item at position 2. Then we get to $\{(AB), (BA)\}$ or to $\{(CA), (AC)\}$, so both caches contain the same items, but in a different order. The next request can be any item not in the cache, e.g., (D). Then we go to $\{(DA), (DB)\}$ or $\{(DC), (DA)\}$, which are both states with the same item in position 1 and different items in position 2. We now repeatedly request an item in position 2 followed by an item not in the cache, and the caches never synchronize. Alternatively, if the caches start synchronized, they will remain synchronized forever. Interestingly, in this example the steady state cache occupancy depends on the initial condition, but the miss probability equals 0.5 for any of the considered initial conditions.

2) Example 2 (non-positive request stream with FIFO over reliable channels): We now extend the previous example in such a way that the steady state miss probability depends on the initial conditions. As before, let $p = 1$, $M = 2$, $x = 4$, $b_i = 0.5$ and $F = \{A, ..., Z\}$, and the initial cache contents are $\{(AB), (AC)\}$. Consider the request stream $(B|C, D, A, E, D, C, D, B)$. After serving these 8 requests, the caches contents are $\{(BC), (BD)\}$. Let the next 8 requests be $(C|D, E, B, A, E, D, E, C)$, i.e., we replaced A, B, C, D, E in the request stream by requests to B, C, D, E, A, respectively. The resulting contents at caches are $\{(CD), (CE)\}$. Repeating the above request streams, the caches eventually reach state $\{(AB), (AC)\}$, when the whole process restarts. Out of every 8 subsequent requests, 4 result in a cache miss, i.e., the steady state miss probability equals 0.5. Nonetheless, if the caches start synchronized then the steady state miss probability equals 0.75. Hence, the steady state miss probability depends on the initial conditions.

3) Example 3 (positive request stream with FIFO over reliable channels): Let $p = 1$, $M = 2$, $x = 4$, $xb_i = 2$, $N = 3$, $\lambda_n = 1/3$. Consider the following two scenarios. In the first scenario the two caches store the same contents initially, in the same order. Then, the system will cycle through

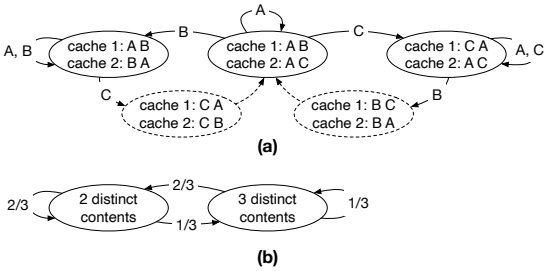

Fig. 3. Dependence on initial conditions: states of a FIFO cache system, with catalog size 3, with 2 caches with capacity 2 each, accounting for symmetries: (a) edges are labeled with requested content. From dotted states, the system immediately transitions to an equivalent state; (b) lumped state space of MC equivalent to (a). The hit probability equals 7/9, greater than 2/3 in case the two caches are started with the same contents, in the same order.

configurations where the two caches will always store the same contents in the same order. *For all practical purposes, the system behaves as a single cache with capacity to store two objects.* By symmetry, each content is stored in cache $2/3$ fraction of the time. Also by symmetry, the hit probability of each content is $2/3$, and the system hit probability is $2/3$.

In the second scenario one of the contents is initially stored in both caches, but the second position of each cache contains a distinct content. Then, the first request will generate a hit. Subsequently, the system will transition across states returning infinitely often to the state where a hit occurs with probability 1. Figure 3 shows the system states, already accounting for symmetries. States $\{(AB), (AC)\}$, $\{(BC), (BA)\}$ and $\{(CA), (CB)\}$, for instance, are equivalent, and are jointly denoted by $\{(AB), (AC)\}$. Clearly, in the middle state the hit probability equals 1 and in the other two states the hit probability equals $2/3$. The hit probability under this scenario is $7/9$, i.e., higher than in the first scenario. Thus, the steady state performance of a partitioned FIFO cache may depend on the initial configuration of the system. The example can be generalized to larger cache sizes, whenever $xb_i = N - 1$.

4) Example 4 (positive request stream with FIFO over unreliable channels): Let $p < 1$, $M = 2$, $xb_i = 2$, $N = 3$, and $\lambda_n = 1/3$. Consider two initial conditions, $\{(AB), (AC)\}$ and $\{(AB), (AB)\}$. As in the previous example, the set of visited states is different for the two initial conditions, i.e., the Markov chain is reducible and the stationary distribution of states is not unique, i.e., it depends on the initial condition. The steady state miss probability under the two initial conditions is given by

$$\mathbb{P}_{miss}^{FIFO/FIFO}(\{(AB), (AC)\}) = \frac{-2p^3 + 12p^2 - 18p + 9}{9 - 6p}, \tag{1}$$

$$\mathbb{P}_{miss}^{FIFO/FIFO}(\{(AB), (AB)\}) = \frac{-3p^3 + 20p^2 - 33p + 18}{9(2 - p)}. \tag{2}$$

Tables II and III show the system states starting at positions $\{(AB), (AB)\}$ and $\{(AB), (AC)\}$, respectively, accounting for symmetries, e.g., noting that states $\{(AB), (AC)\}$ and $\{(BA), (BC)\}$ are equivalent, and are jointly denoted by $\{(wx), (wy)\}$. In particular, state $\{(wx), (wy)\}$ is reachable

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

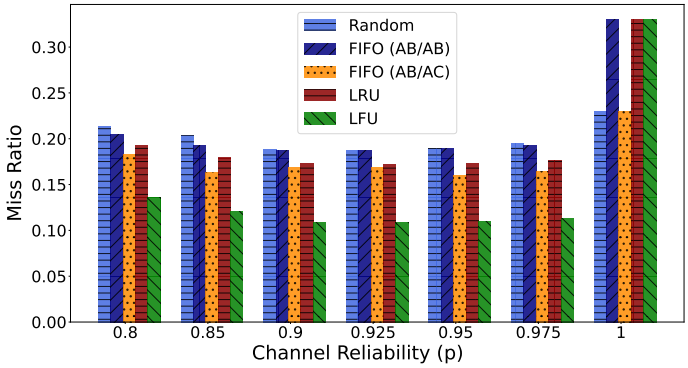

Fig. 4. Miss ratio of LRU, LFU, Random. Miss ratio of FIFO policy under different initial conditions.

in one transition from $\{(yw),(xw)\}$, as a request to x from the latter can produce $\{(xy),(xw)\}$ that is equivalent to $\{(wx),(wy)\}$. The example can be generalized to larger cache sizes as long as $xb_i = N-1$, and shows that the steady state performance of a partitioned FIFO cache may depend on its initial configuration. Figure 4 shows the miss ratio under different initial conditions as a function of the channel reliability (p) under LRU, LFU and FIFO. We note that depending on the initialization of the FIFO system, FIFO may outperform LRU.

Sufficient conditions for independence from initial state: The above examples had in common that the request stream was either non-positive (Examples 1 and 2) or that the FIFO cache sizes satisfy $xb_i = N-1$ (Examples 3 and 4). The following theorem shows that for LRU and Random policies, and for FIFO with $xb_i \leq N-2$, the initial state does not affect the steady state behavior when caches are subject to positive request streams.

Theorem 1 (Independence from initial state)**.** *When $p < 1$, under LRU and Random strategies the initial condition does not affect the steady state behavior if caches are subject to positive request streams. If $xb_i \leq N-2$, for $i = 1,\ldots,M$, the result also holds for FIFO.*

Proof. The proof is similar to that of [43, Theorem 2]. If $p < 1$ and the request stream is positive, one can construct a stream of requests that causes each of the caches to store any of the contents, in any desired order. This is because the MC corresponding to an isolated LRU or Random cache is irreducible. If $xb_i \leq N-2$, for $i = 1,\ldots,M$, an isolated FIFO cache is also irreducible.

Given that $p < 1$, the state of any cache can change without impacting the state of the other caches, and from any initial state one can reach any system state. As the above argument holds for any initial condition, it implies that the MC of the whole system is irreducible. □

It is also worth noting that if $p = 1$ and all caches have the same size the above result still holds. Indeed, in this case the independence from initial conditions follows from the eventual synchronization and aging of all caches, which we prove next.

TABLE II
TRANSITION PROBABILITIES IN A SYSTEM WITH 2 FIFO CACHES, WITH INITIAL STATE WHEREIN TWO CACHES ARE AT SAME CONFIGURATION

origin	transition description	destination	probability
$(yw),(xy)$	request for y	$(yw),(xy)$	1/3
	request for x or w not leading to insertion	$(yw),(xy)$	$\frac{2}{3}(1-(p(1-p)+p^2))$
	request for x leading to insertion	$(xy),(xy)$	$(p(1-p)+p^2)/3$
	request for w leading to insertion	$(yw),(wx) \equiv (wx),(xy)$	$(p(1-p)+p^2)/3$
$(xy),(xy)$	request for x or y	$(xy),(xy)$	2/3
	request for w not reach the caches	$(xy),(xy)$	$(1-p)^2/3$
	request for w leading to insertion in both caches	$(wx),(wx) \equiv (xy),(xy)$	$p^2/3$
	request for w leading to insertion only in 1st cache	$(wx),(xy)$	$p(1-p)/3$
	request for w leading to insertion only in 2nd cache	$(xy),(wx) \equiv (yw),(xy)$	$p(1-p)/3$
$(wx),(xy)$	request for x	$(wx),(xy)$	1/3
	request for w or y not leading to insertion	$(wx),(xy)$	$\frac{2}{3}(1-(p(1-p)+p^2))$
	request for w leading to insertion	$(wx),(wx) \equiv (xy),(xy)$	$(p(1-p)+p^2)/3$
	request for y leading to insertion	$(yw),(xy)$	$(p(1-p)+p^2)/3$

TABLE III
TRANSITION PROBABILITIES IN A SYSTEM WITH 2 FIFO CACHES, IN INITIAL STATE WHEREIN TWO CACHES ARE WITH SAME (RESP., DISTINCT) CONTENTS AT THEIR MOST RECENT (RESP., OLDEST) POSITIONS

origin	transition description	destination	probability
$(yw),(xw)$	request for w	$(yw),(xw)$	1/3
	request for x or y not leading to insertion	$(yw),(xw)$	$\frac{2}{3}(1-(p(1-p)+p^2))$
	request for x or y leading to insertion	$(xy),(xw) \equiv (wx),(wy) \equiv (yw),(yx)$	$\frac{2}{3}(p(1-p)+p^2)$
$(wx),(xw)$	request for w or x	$(wx),(xw)$	2/3
	request for y not reach the caches	$(wx),(xw)$	$(1-p)^2/3$
	request for y leading to insertion in both caches	$(yw),(yx) \equiv (wx),(wy)$	$p^2/3$
	request for y leading to insertion in one cache	$(yw),(xw)$	$2p(1-p)/3$
$(wx),(wy)$	request for w	$(wx),(wy)$	1/3
	request for x or y not leading to insertion	$(wx),(wy)$	$\frac{2}{3}(1-(p(1-p)+p^2))$
	request for x or y leading to insertion	$(wx),(xw)$	$\frac{2}{3}(p(1-p)+p^2)$

B. Conditions for Eventual Synchronization and Aging

Motivation and goals: Understanding conditions under which caches eventually synchronize is key in light of the decision between pooling and separation. Indeed, when channel reliability is close to 1 and caches tend to become synchronized, pooling should be preferred. Alternatively, if pooling cannot be implemented, strategies to counter synchronization and aging, such as purportedly dropping some requests at random or deploying different cache replacement policies across caches can increase hit probability, as further discussed in Section IV-C.

Eventual synchronization: Next, we establish conditions under which all caches will eventually synchronize.

Theorem 2 (Caches eventually synchronize)**.** *Consider $p = 1$, all caches having the same size, under LRU, subject to positive*

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

request streams. Then, eventually, the system will converge to a class of states where all caches store the same contents, in the same order. Once such a state is reached, the system will behave as a single cache with size xb_i. If $xb_i \leq N-2$, $1 \leq i \leq M$, the result also holds for FIFO subject to positive request streams.

Proof. We begin considering LRU caches. All caches have the same size xb_i. Observe that a stream of requests for contents stored in one of the caches will cause those contents to also populate the other caches. Then, the contents can be reordered through a subsequent series of xb_i requests for the stored contents. Once a state is reached where all caches store the same contents in the same order, the set of caches behaves, for all practical purposes, as a single cache with capacity xb_i.

Next, consider FIFO caches and $xb_i \leq N/2$. Consider a tagged cache, which serves as reference, and consider a stream of requests for the contents stored in the tagged cache. After serving such a request stream, all caches store the same contents, but possibly in different order. A request for a content not stored in any of the caches will cause it to be stored at the same position in all caches. If there are at least xb_i contents not stored in any of the two caches, which is guaranteed to be the case if $xb_i \leq N/2$, a stream of requests to those contents will cause xb_i misses to the two caches. The states of the caches will then be identical after these misses. □

A consequence of the above result is that the steady state behavior of the system is independent of the initial conditions under the considered conditions. Note that the above theorem is tight in the sense that if $xb_i = N-1$ then the initial conditions of a FIFO system may impact its steady state behavior, for any value of p, as shown in the previous section. In that case, the caches are not guaranteed to eventually synchronize.

It is interesting to note that starting from the configuration where all caches store the same contents in the same order, it may take a significant number of requests for the caches to store different contents, as formalized next.

Claim 1. *When caches start synchronized, the mean number of requests it takes for the caches to eventually store different contents, for $0 < p < 1$, can be expressed as*

$$\tau(p) = \frac{1}{1-(1-p)^M - p^M}. \tag{3}$$

Proof. The caches will store different contents as soon as a request reaches a subset of the caches. The request does not reach any of the caches (resp., reaches all caches) with probability $(1-p)^M$ (resp., p^M). Therefore, the mean time to break the symmetry of the caches is geometrically distributed, with success probability given by the denominator of (3). □

Note that as p approaches 0 or 1, the time it takes for the caches to store different contents increases. In particular, for $p = 1$ the result is in agreement with Theorem 2.

The above observations can be framed in light of [9]. In particular, it follows from Theorem 3 in [9] that pooling is the best configuration when $p = 1$. This observation reinforces the relevance of considering policies different from LRU in the regime where p is close to 1 and all requests are broadcasted to all caches. Indeed, the caches tend to become synchronized as p gets close to 1, which may degrade system performance due to cache aging. As a consequence, channel reliability can decrease hit probability and the Random policy may outperform LRU and FIFO, as discussed in the sequel.

C. Countering the Aging due to Synchronization

Motivation and goals: Next, we consider strategies to counter the impact of eventual cache synchronization and aging. Such strategies are used to promote diversity across caches. In particular, we show that dropping some requests at random (Theorem 3) or deploying different cache replacement policies across caches (Theorem 4) can increase diversity levels and therefore increase hit probability.

1) Increasing reliability may decrease hit probability: Next, we show that increasing the channel reliability p may decrease the hit probability in the case of partitioned LRU caches. This is due to the inherent dependence between caches, which increases as the channel reliability p increases. When channels are reliable ($p = 1$), in particular, the states of all caches are eventually always the same. As a consequence, as indicated in the following theorem, unreliability ($p < 1$) may be beneficial.

Theorem 3 (Unreliability may be beneficial)**.** *Decreasing channel reliability p may cause an increase in hit probability.*

In the supplementary material, we prove the above theorem in its simplest setting. In the following sections, we show that the theorem also holds in realistic settings, through trace driven simulations with real workloads (Sections VII and VIII).

Theorem 3 indicates that for cache replacement policies that result in cache contents being dependent, channel unreliability may benefit performance. Such a result does not hold when caches are independent due to the cache replacement policy.

2) Deploying distinct policies across caches can increase hit probability: If the channel does not provide randomness, it can be compensated by the cache policy. Indeed, (a) cache replacement policies, (b) unreliable channels and (c) random hashing mechanisms to dispatch requests in multi-cache systems [9] are three sources of randomness that contribute to diversity among contents across caches. To illustrate this for cache replacement policies, we now show that there are scenarios where the Random replacement policy is superior compared to LRU, LFU and FIFO.

Theorem 4 (Randomized policies and/or distinct policies across caches may be beneficial)**.** *For large enough reliability, the hit probability of a system counting with one of the following properties may be larger than that of a system where all caches deploy the same policy (LFU, LRU or FIFO),*

1) *diversity through distinct policies: one cache operating under a distinct (possibly Random) replacement policy*
2) *diversity through uniform randomization: all caches operating under the Random replacement policy*

Proof. The theorem is a consequence of the synchronization across caches (see Theorem 2). Consider $M = 2$ caches, each

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

with capacity to store $xb_i = 2$ contents, and a catalog of $N = 3$ contents, all of them with the same probability of being accessed at each request. Let $p = 1$. If all caches operate under the same policy, and caches are initialized with the same (resp., different) contents, the hit probability of FIFO equals 2/3=0.66 (resp., 7/9=0.77). The hit probability of LRU and LFU is 2/3, irrespectively of the cache initialization.

1) if one cache operates under Random, and the other under LFU, the hit probability is 8/9;
2) if all caches operate under Random, the hit probability equals 7/9.

□

Content diversity across caches in cache networks has a significant impact on system performance [35], [44]. The results we presented in this section show that channel unreliability is a source of randomness that may positively impact diversity and system performance. In particular, we have shown that the eventual synchronization and aging across caches may impact steady state performance, and that if the system does not have significant diversity across caches one can compensate such lack of diversity through the use of randomized or diverse policies across caches. In the following sections, we support those claims through an analytical toy model, followed by trace driven simulations with synthetic and realistic workloads.

V. Insights From an Analytical Toy Model

In this section, we develop an analytical model of the cache miss probability for LRU and LFU caching over unreliable channels. Our goals are to illustrate, in the simplest possible setting, the extent at which unreliability impacts the decision between pooling and separation and at which distinct policies across caches may be beneficial.

In the realm of the toy model that follows, LRU and LFU correspond to dynamic and static content placement, as detailed next.

A. Simple Toy Model

We consider a simple setting with a total cache size of $x = 2$. In the case of pooling there is a single cache, $M = 1$. In the case of separation there are two unit size caches, $M = 2$, and a fraction $b_i = 0.5$ of the cache space is allocated to cache i, $i \in \{1, 2\}$, as shown in Figure 2. There are $N = 2$ contents in the catalog, $f_1 = A$ and $f_2 = B$. Requests are made for contents A and B with probabilities $\lambda_1 = q$ and $\lambda_2 = 1-q$, respectively, i.e., in the particular case where $N = 2$, we denote by q the popularity of the most popular content, $q = \lambda_1 \geq \lambda_2$. Communication links used for sending requests are unreliable, the probability of a successful transmission is p (see Table I), and transmissions fail independently. Although the setting is admittedly simple, it allows us to explore issues involved in the choice of policy, and whether or not to pool or partition cache storage resources.

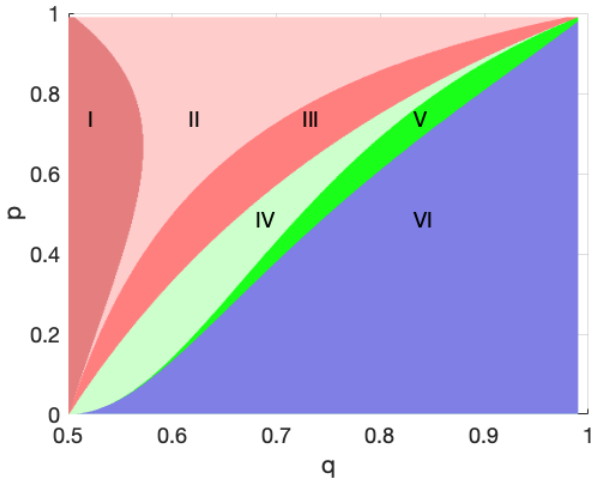

Fig. 5. Summary of operation regions: in regions I, II and III, pooling is preferred over separation; in regions IV, V and VI, (LFU, LFU) separation is optimal and (LRU, LFU) is the second best option. In region VI, pooling is the worst alternative.

B. Insights from Toy Model

For all practical purposes, in the context of this section LFU corresponds to a static policy that stores the most popular content in each cache, whereas LRU corresponds to a dynamic policy that evicts the content stored in cache after a miss, to leave space for the new content. Whereas LFU has the advantage of always caching the most popular content, it does not promote diversity. LRU, in contrast, has the advantage of promoting diversity at the expense of eventually caching the less popular content in both caches.

Under the toy model, 1) high channel reliability favors pooling whereas low channel reliability favors separation with LFU at both caches to promote robustness. In addition, 2) under high channel reliability, and in the impossibility of implementing pooling, a combination of (LRU, LFU) outperforms (LFU, LFU) as one LFU cache serves to store the most popular content whereas the LRU cache promotes diversity.

Generally speaking, deploying caches that implement different policies is a way to promote content diversity across caches. In particular, if the channel is reliable, enforcing diversity through distinct cache replacement policies is a way to improve hit probability. In practical scenarios, LRU has also the benefit of leveraging spatial correlations in the request stream. In this toy example, however, the request stream follows the independent reference model (IRM). For stationary IRM request streams, it is well known that the static LFU policy is optimal [6]. Our results suggest that this may still be the case for stationary IRM requests to a set of caches accessed through an unreliable broadcast channel, as far as the optimal decision between pooling or separation is taken.

When pooling is not an option, the above intuitive argument indicating why (LRU, LFU) outperforms (LRU, LRU) goes in the same vein as the rationale behind why a Random cache is beneficial in face of high reliability (see Theorem 4).

Figure 5 summarizes the regions of operation of the considered toy model:

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

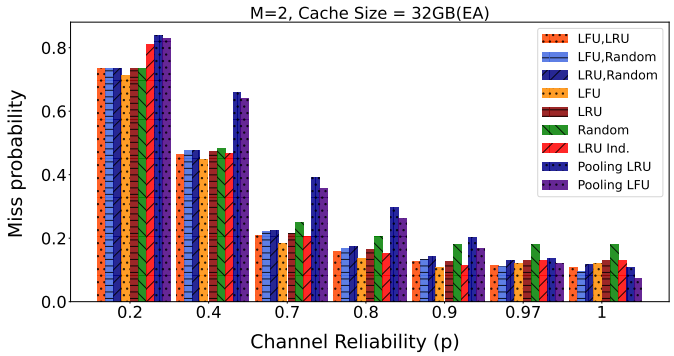

Fig. 6. Ranking policies under the Wikipedia trace: for $0.97 < p \leq 1$ pooling is the best alternative, for $0.9 < p \leq 0.97$ a combination of (LFU, Random) is the best alternative, followed by (LFU, LRU), and for $0 < p \leq 0.9$ (LFU, LFU) is optimal.

1) Regions I, II and III (red): pooling is the best alternative, and a mix of (LFU, LRU) is the second best option
2) Regions IV and V (green): pooling is neither the best nor the worst policy. The optimal policy is (LFU, LFU) and a mix of (LFU, LRU) is the second best option
3) Region VI (blue): in this region, pooling is the worst alternative. The optimal policy is (LFU, LFU).

C. Lessons Learned

In summary, the analytical toy model supports two key observations: 1) as channel reliability increases, pooling becomes more attractive; 2) if pooling cannot be implemented even though it is the best alternative then using heterogeneous policies across caches is beneficial. To further illustrate how the toy example provides insight that generalizes to realistic settings, we consider the Wikipedia trace, to be further detailed in Section VIII. Under the Wikipedia trace, accounting for a large catalog of contents, a cache size of $x = 32$GB and temporal locality, the above bullets 1), 2) and 3) correspond to channel reliability in the following three ranges, respectively: 1) $0.97 < p \leq 1$, 2) $0.9 < p \leq 0.97$ and 3) $0 < p \leq 0.9$ (see Figure 6).

Note that under the toy model (LFU, LFU) is the best alternative in regions IV and V, given the assumption of IRM traffic. Under the real workload, in contrast, (LFU, Random) slightly outperforms (LFU, LFU) for $0.9 < p \leq 0.97$. In the other regimes, the ordering of policies under the toy model is in agreement with that observed under the Wikipedia trace.

VI. Discussion of Assumptions and of the Feasibility of the Considered Model

In this section, we present practical considerations and challenges associated with applying our model to a real-world scenario.

A. Assumptions

Assumption 1. *A phone/device needs to connect to different base stations and send the same request to each of them.*

In existing mobile network architectures, for a device to connect to multiple base stations, the base stations would have to perform handover and the devices would have to adapt to different channel conditions towards multiple antennas. Such handovers and adaptations, in turn, may consume battery and cause delays. A potential workaround would be to use adaptive strategies that dynamically select a subset of base stations to contact based on the channel conditions. Doing so would reduce the number of handovers required, and could be captured in our model by introducing a new set of variables that capture the probability that a station is selected at a given slot. Once the subset of base stations to contact is selected, the current model is readily applicable, setting the channel success probability equal zero for each base station that was not selected at a given time slot, and keeping unchanged the success probability for the other base stations.

Going beyond existing technologies, multi-connectivity, which allows a mobile user to communicate with multiple base stations at the same time [45], is considered a key enabler in 5G mobile networks, and hence our model is directly applicable in future mobile network architectures.

Assumption 2. *When multiple base stations have a cache hit, they may all send the same content to the user.*

The practical aspects of wireless content delivery depend on the available infrastructure. If base stations can coordinate for the purposes of beamforming, a cache hit at multiple base stations may increase throughput. Such analysis, accounting for throughput as a metric to be optimized, is left as subject for future work. Alternatively, the proposed solution can be coupled with other mechanisms of coordination between base stations, possibly with the assistance of the mobile devices. The latter can employ algorithms to determine the optimal base station from which to receive the content, taking into account factors such as signal strength, channel conditions, and content availability, and inform such decisions to the base stations where a hit occurred. Caches can communicate with each other to make informed decisions on which cache should send the content, avoiding unnecessary duplication or flooding. Those mechanisms are out of the scope of this paper, and are left as subject of future work.

Assumption 3. *When there are cache misses at multiple caches, all caches may request content from the origin server simultaneously. The origin server must have enough capacity to serve those requests.*

To mitigate the potential issue of caches flooding the origin server with simultaneous requests, one could implement a mechanisms to coordinate the requests to the original server using a coordination protocol. Coordination can be achieved in a distributed manner, using cache-to-cache communication, in a centralized manner, which allows caches to negotiate and determine the cache that should request the content from the origin server (and then exchange content among themselves). Additionally, mechanisms like request throttling or randomized back-off algorithms can be employed to regulate the request traffic and avoid overwhelming the origin server. The models presented in this work are applicable in a setup where those mechanisms are implemented at a fine timescale, and cache requests occur at a coarser scale.

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

B. Extensions of the Model

In the supplementary material, we consider an extension of our model to illustrate how some of the above assumptions can be relaxed. The extension involves three parameters, corresponding to the above three assumptions, respectively. The first parameter, denoted by s, represents the probability that a phone or device issues a request to a single cache rather than connecting to multiple base stations. The second parameter, denoted by e, represents the probability of effective service. When multiple base stations have a cache hit, if all go through effective services, all of them send the same content to the user. Otherwise, only a subset of base stations will serve the request. The third parameter, denoted by u, represents the probability of an update occurring after a cache miss, and controls number of caches that simultaneously request content from the origin server. In the supplementary material, we indicate how these parameters provide insights into the system.

It is worth noting that by relaxing some of the assumptions listed in this section, related to no coordination among caches, it is possible to devise caching policies beyond the scope of this work. Such policies may outperform the caching schemes analyzed in this paper, at the cost of control and synchronization overhead. The analysis of the tradeoff between performance and control overhead is left as subject of future work.

VII. Numerical Evaluation

In what follows we use simulations to validate the analytical findings, to indicate how those findings generalize for large cache space, number of contents and number of caches, and to study the sensitivity of different cache replacement policies to cache partitioning. Whereas the results in this section are based on an IRM workload, in the following section we consider real-world traces. In particular, the results in this section serve also to evaluate different policies against related LRU-based solutions presented in [9].[1]

A. Simulation Methodology

We consider M caches in the system. As in [9], the total cache size x varies from 10 to 8500 content items, and a fraction b_i of the total cache size is allocated to cache i, $1 \leq i \leq M$. The content catalog includes 10 million data items, $N = 10^7$, and the content popularity distribution follows Zipf's law with exponent $\alpha = 1.8$,

$$\lambda_i = \frac{1/i^\alpha}{\sum_{j=1}^{N} 1/j^\alpha}. \tag{4}$$

Each content has unit size. Unless otherwise noted, all requests are sent to all caches. We consider channel reliability values ranging from 0.2 to 1 in increments of 0.2, $p \in \{0.2, 0.4, 0.6, 0.8, 1\}$, where p = 1 corresponds to a reliable channel. Each experiment is executed for 50 runs, to produce 95% confidence intervals.

[1]All source code available at https://tinyurl.com/unrcache

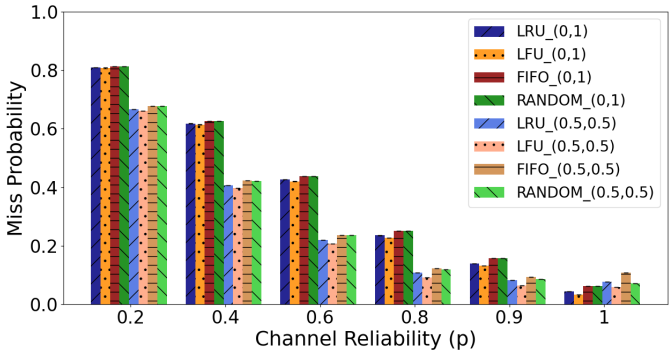

Fig. 7. Miss ratio vs. channel reliability p for LFU and LRU cache replacement policies, with pooling (0,1) and with separation (0.5,0.5), $x = 50$.

B. The Case of Two Caches

We start with validating the observations made based on the analytical model for a system with $M = 2$ caches and equal split $b_i = 0.5$, compared against pooling. Figure 7 shows the cache miss ratio as a function of channel reliability for the LRU, LFU, FIFO and Random cache replacement policies, under pooling (first four bars in each group) and partitioning (last four bars in each group). The figure confirms all the observations made based on the analytical model, i.e., there is a threshold $p^\star$ for LRU and for LFU above which pooling is optimal, and LFU with $b_1 = b_2 = 0.5$ (for $p < 1$) or LFU with pooling (for $p = 1$) outperforms its counterparts, including LRU with separation or pooling.

C. Small Sensitivity of LFU to Cache Split

Before turning to more than $M = 2$ caches, we further investigate unequal cache space allocation. For this we consider $M = 2$ caches and assess how the optimal splitting depends on the algorithm used (LFU, LRU, FIFO, Random). Recall that for LRU splitting the cache has an impact on the miss ratio, and there is an optimal split that depends on the content popularity distribution and the channel reliability [9]. Figure 8(a) shows the cache miss ratio for various cache splits, from $(0, 1)$ (i.e., pooling), to $(0.5, 0.5)$ (i.e., equal split) for a channel reliability of $p = 0.6$. The subfigures show results for caches size $x \in \{50, 100, 300, 500\}$. The figure allows us to make two important observations. First, it confirms that LFU outperforms LRU in the considered settings, independent of the cache split. Second, it shows that for the LFU cache replacement policy the cache split has rather little effect on the cache miss ratio. Figure 8(b) further shows corresponding results for a channel reliability of $p = 0.97$ for the same cache sizes for cache splits (b_1, b_2). The figure confirms that LFU outperforms LRU irrespective of the cache split, and for $p = 0.97$ the miss probability is slightly more sensitive to the cache split when compared against $p = 0.6$.

D. Multiple Caches

Figure 9(a) shows the cache miss ratio as a function of the channel reliability p for four cache replacement policies

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

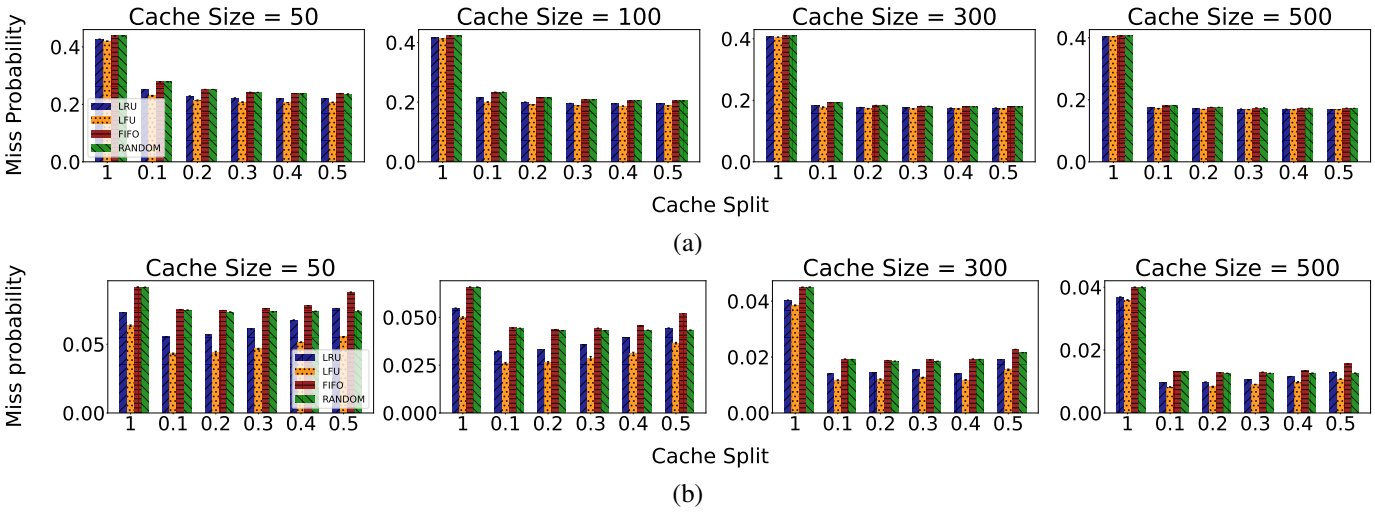

Fig. 8. Miss ratio for various cache splits for LFU, LRU, FIFO and Random caching. $M = 1$ and $M = 2$ caches, channel reliability (a) $p = 0.6$ and (b) $p = 0.97$, for various cache sizes.

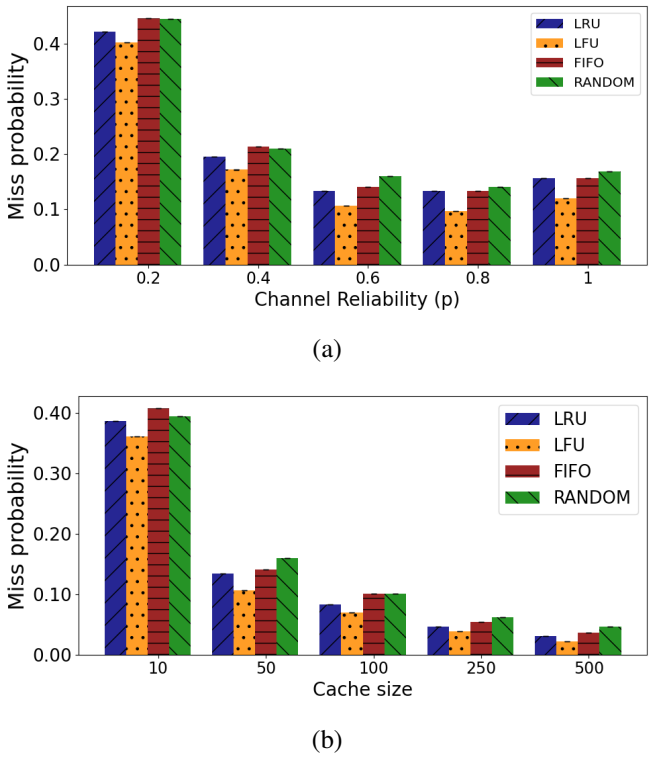

Fig. 9. Miss ratio for LFU, LRU, and Random caching. $M = 5$ caches: (a) cache size $x = 50$ and (b) channel reliability $p = 0.6$

(LRU, LFU, FIFO and Random) for a cache size of $x = 50$ and $M = 5$ caches, under equal split. The figure shows that LFU outperforms all other policies in terms of miss ratio for all channel reliability values. In addition, it shows that increasing the channel reliability can increase the miss rate for LRU. Figure 9(b) shows the cache miss ratio for various caches sizes, for $p = 0.6$ for the same cache replacement policies. The results are consistent, and show no significant impact of the cache size on the *relative* miss rates across the considered policies. In particular, in all the considered scenarios, LFU performed the best, followed by LRU.

To further investigate the impact of cache separation, Figure 10(a) shows the cache miss ratio as a function of the channel reliability p for the LRU, LFU, FIFO and Random policies, for $M \in \{1, 2, 3, 4\}$ caches with a total size of $x = 12$. The figure shows that LFU outperforms the other policies irrespective of the number of caches and the channel reliability, with a significant gain. It is also noteworthy to observe that the cache miss ratio is a convex function for all scenarios, similar to the analytical results, and it increases with p for $M = 3$ and $M = 4$ caches.

Figure 10(b) show corresponding results for a total cache size of $x = 120$. The figure allows us to draw similar conclusions, even though the difference in terms of miss rates is almost negligible, due to the low aggregate popularity of the tail of the content popularity distribution. We can conclude that LFU works significantly better than LRU, FIFO and Random when cache storage is scarce and the workload is stationary, regardless of the number of partitions. In the following section, we consider a real traffic workload.

VIII. Benefits of Randomness Under Real Traffic

In this section we present a trace-driven evaluation of a wide range of state-of-the-art cache replacement policies with cache partitioning, over unreliable channels. Our objective with the evaluation is to assess the impact of cache size and of cache partitioning on system performance for a large content catalogue and considering state-of-the-art policies. In particular, our results extend related work on LRU-based solutions presented in [9], [46] by $i)$ accounting for different policies and $ii)$ considering realistic traces.

Our selection of benchmark policies was informed by recent surveys on caching policies, specifically focusing on scenarios involving unreliable channels and partitioned caches [47], [48]. In addition, the chosen policies were aligned with the ones used in [46]. Notably, the benchmark includes classical and machine learning-based policies that are readily available at WebCacheSim [49], [50].

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

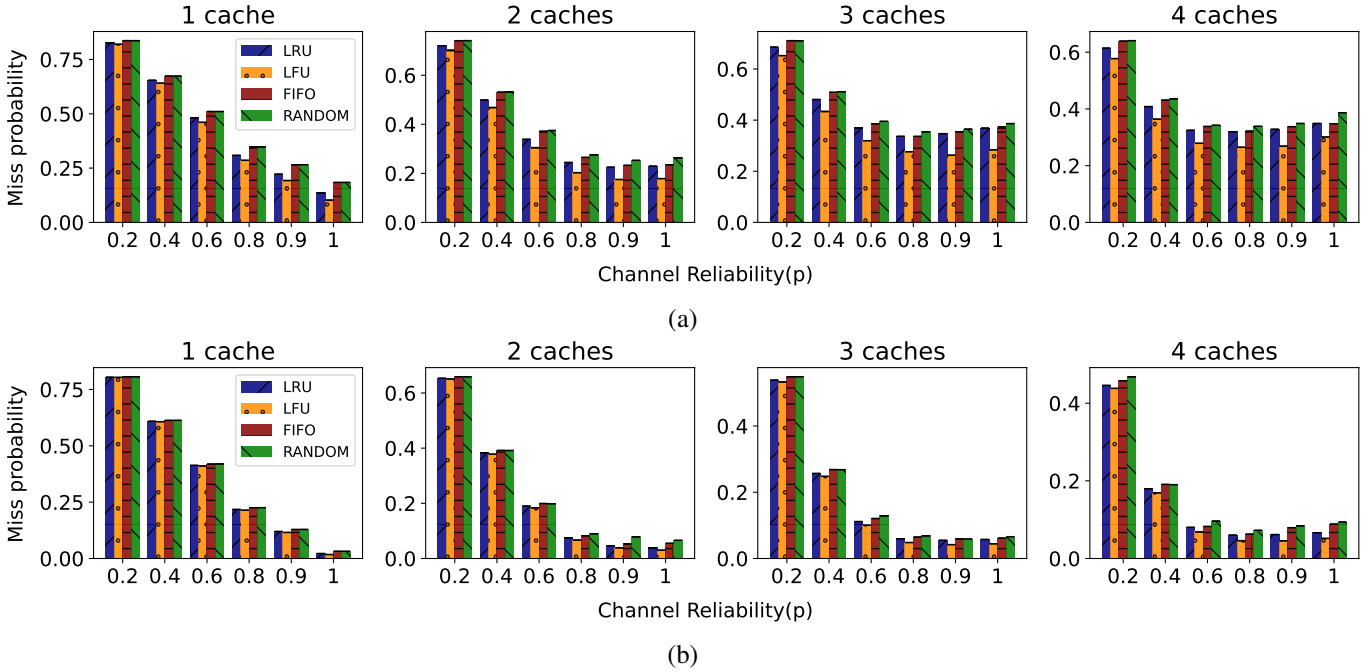

Fig. 10. Miss ratio vs. channel reliability p for a cache size of (a) 12 and (b) 120, for LFU, LRU, FIFO and Random, 1 to 4 caches.

A. Evaluation Methodology

We adapted the C++ based simulator WebCacheSim v2 [49], [50] to work with partitioned caches and unreliable channels. The cache replacement policies tested in our experiments include the well-known LRU and LFU and their variations: Bloom-filter LRU (BLRU), LRUK ($K = 4$) and LFU with Dynamic Allocation (LFUDA). We also consider the learning-based policies such as Learning Relaxed Belady (LRB) [50] and Learning Cache Replacement (LeCaR) [51], and the Random policy.

For the evaluation we use a real trace of requests collected from a Content Delivery Network (CDN) node located on the US West coast, which serves photos and other media contents for Wikipedia pages[2], referred to as the Wikipedia trace. The trace is divided into a warmup phase and a stationary regime [50]. Table IV shows statistics of the trace. The wikipedia trace works with sized content objects in bytes. Therefore, the performance metrics we use are content miss probability, which not accounts for content sizes, i.e, considers each content as unique object, and the byte miss probability which accounts for content sizes.

TABLE IV
SUMMARY OF THE WIKIPEDIA TRACE.

Duration (Days)	14
Total Requests (Millions)	2,800
Unique Obj Requested (Millions)	37
Total Bytes Requested (TB)	90
Unique Bytes Requested (TB)	6
Warmup Requests (Millions)	2,400
Request Obj Size Mean (KB)	33
Request Obj Size Max (MB)	1,200

B. Impact of Cache Size

We start with addressing the impact of the cache size on the performance of the cache replacement policies for a large content catalogue. Figure 11 shows the miss probability of the cache replacement policies for $M = 2$ partitions with Unequal Allocation (UA), and channel reliability of $p = 0.8$, for various cache sizes. The results show that as the cache size increases, miss probability decreases. We also note that byte miss probability is higher than content miss probability, due to the large variety of content sizes. Indeed, a missing content can severely impact the byte miss probability if having a large size, e.g., 1GB, which rather increases the metric values, whilst other contents of, e.g., 100KB are hitting the cache. Furthermore, when the cache is full and a request for large-sized contents arrives, the insertion of this single content requires the eviction of many small-sized contents, or the insertion does not occur. These results validate the numerical results with synthetic traces shown in Figures 8 and 9(b). In what follows, we provide further insights on the role of cache size, e.g., comparing pooling against partitioning, so as to reason about the impact of partitioning.

C. Impact of Partitioning

Next, we consider the impact of partitioning under unreliable channels. Figures 12(a) and 12(b) show the cache miss probability for the LRU policy with cache size 64GB, for various number of partitions under the equal allocation (EA) policy. The results for the miss probability show that cache partitioning is mostly beneficial when the channel is unreliable, as the probability of all requests getting lost decreases as the number of partitions increases. Interestingly, the benefit of partitioning is significantly lower when considering the

[2]The trace is available for download at the Github page of WebCacheSim: https://github.com/sunnyszy/lrb

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

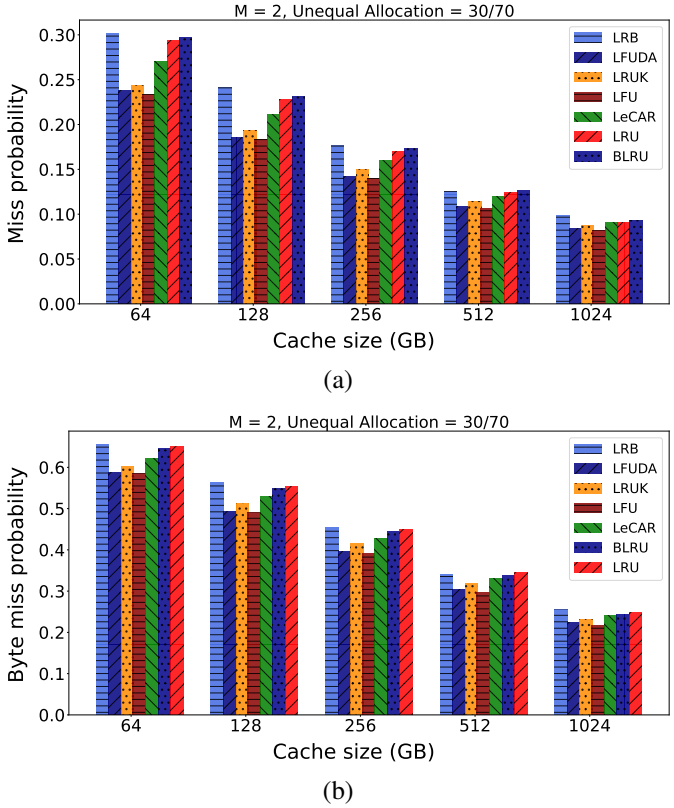

Fig. 11. Miss probability of variety of cache replacement policies. $M = 2$. Cache sizes = 64GB, 128GB, 256GB, 512GB and 1TB, channel reliability $p = 0.8$. Figure (a) shows content miss probability and (b) shows miss probability in Bytes.

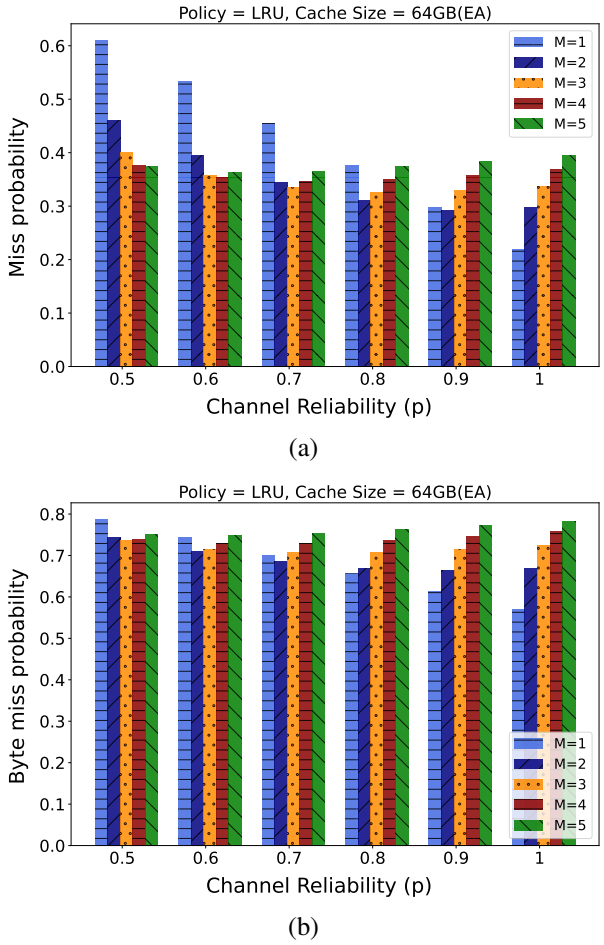

Fig. 12. Miss probability of LRU policy, Equal Allocation and cache sizes = 64GB. Figure (a) shows content miss probability and (b) shows miss probability in bytes.

byte miss probability (Fig. 12(b)), as moderately popular large contents do not get cached due to the lower cache space per partition.

D. Joint Impact of Cache Size and Partitioning

So far we evaluated the advantage of partitioned caches over unreliable channels for a 64GB cache. In what follows, we explore the joint impact of cache size and partitioning on system performance.

We begin by evaluating the impact of cache size under cache pooling and cache partitioning for a variety of cache replacement policies, with $p = 0.8$, as shown in Figure 13. The results show that the benefit of partitioning increases with the cache size. As discussed above, one of the reasons for this phenomenon is the increasing ability to store large contents in each partition, which is possible when the cache size is large enough. These results validate the numerical results shown in Figure 8, where we have already observed that significant gains due to partition occur for larger cache sizes.

Figure 14 shows the miss probability as a function of the channel reliability, for different cache sizes and number of partitions for LRU caching. Although the cache size significantly affects the miss probability (see also Figure 11), it does not qualitatively affect the system behavior. When the channel reliability is small and cache size is large, the system benefits from more partitions. As the channel reliability increases,

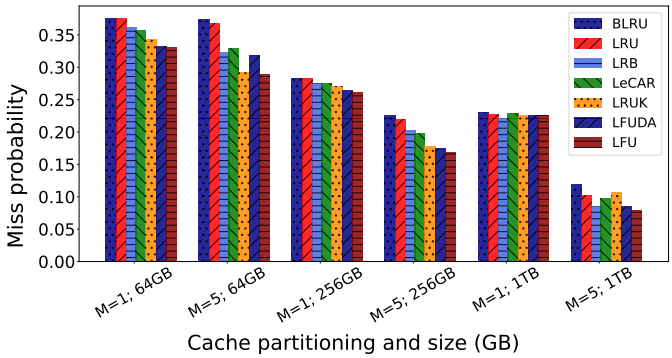

Fig. 13. Miss probability of variety of cache replacement policies under cache pooling ($M = 1$) and cache partitioning ($M = 5$) with EA, for different cache sizes and $p = 0.8$.

less partitions yield lower miss probability, i.e., diversity is preferred over replication.

Following the analysis, we evaluated a variety of cache replacement policies over unreliable channels and cache partitioning, varying channel reliability from 0.6 up to 1. Figure 15 shows the content miss probability as a function of the channel reliability for $M = 3$ partitions, a cache size of 64GB, and equal allocation. The shape of the curves resembles that of

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

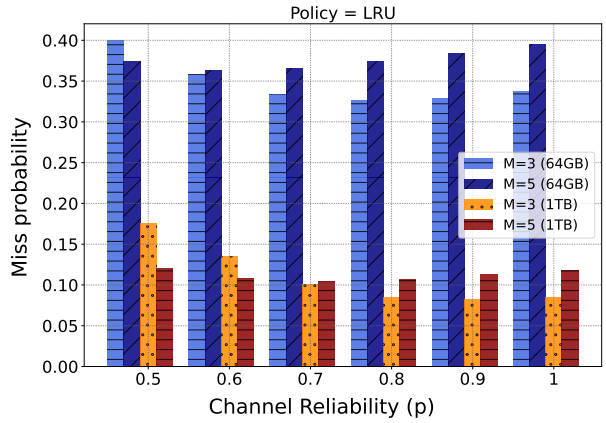

Fig. 14. Miss probability of LRU policy with $M = 3$ and $M = 5$, Equal Allocation and cache sizes = 64GB and 1TB.

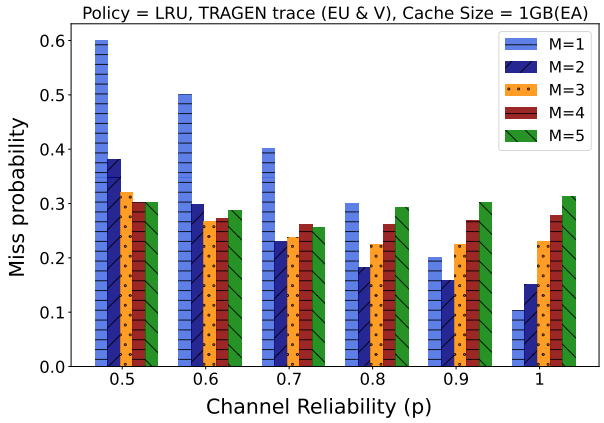

Fig. 16. Miss probability of LRU policy, Equal Allocation and cache sizes = 1GB under TRAGEN trace.

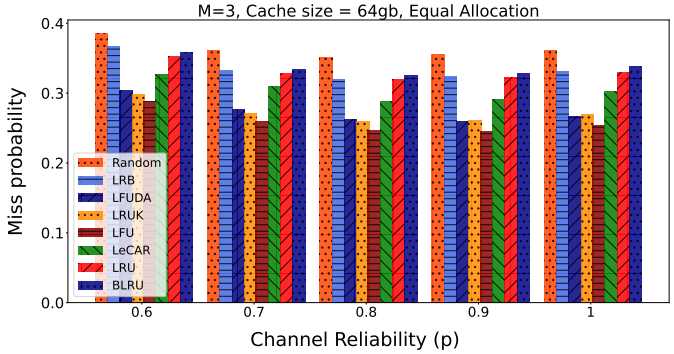

Fig. 15. Miss probability of state-of-art cache replacement policies with $M = 3$, Equal Allocation and cache sizes = 64GB.

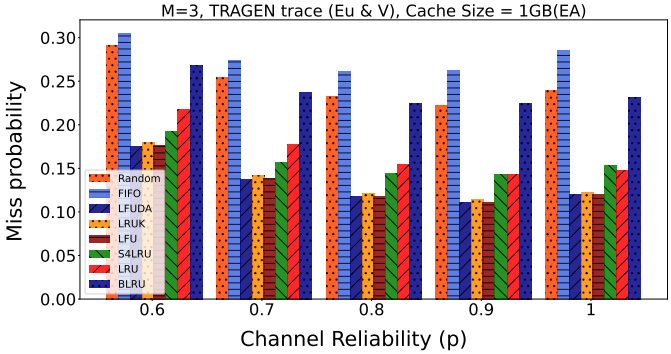

Fig. 17. Miss probability of cache replacement policies with $M = 3$, Equal Allocation and cache size 1GB.

the LRU policy in Figure 12, and comparing the different policies, we observe that LFU and LRU-K policies typically perform best both in terms of content and in terms of byte miss probability. It is interesting to observe that the miss probability curves are non-monotonic convex, and as such they follow Proposition 3, which showed that there is a threshold above which increased reliability leads to increased miss probability.

Remarks: Our results show that there is an intricate relationship between partitioning, channel reliability, cache replacement policy, and that the miss probability can increase with the channel reliability under cache partitioning. This observation, in agreement with the formal results presented in Section VIII under simplistic scenarios, suggests that our findings also hold true in realistic settings. In particular, in a system where caches are equally partitioned and channels are reliable, dropping a fraction of the requests on purpose could improve system performance.

E. Additional Validation with TRAGEN

We extended our webcachesim results to include traces generated by TRAGEN [21], a synthetic trace generator for realistic cache simulations. TRAGEN produces a synthetic trace with characteristic similar to traffic observed at Akamai's production CDN. In the following experiment, in particular, we use two of the four classes of workload offered by TRAGEN, namely Video and EU, which TRAGEN combines to produce the trace considered in the sequel (see Table V).

Results obtained with TRAGEN, shown in Figure 16 and 17, are in agreement with those reported in the previous sections. Figure 16 shows that under LRU caches with Equal Allocation and cache size of 1GB, as M and p increase, partitioning negatively impacts network performance. This result is in agreement with Figure 12.

Figure 17 shows the impact of cache replacement policies under the TRAGEN trace. For $M = 2$ and a cache size of 1GB, LFU, LFUDA and LRUK (with k=4) policies have similar performance, with LFUDA having a slight advantage. Figure 17 also confirms that as p grows the miss probability first decreases and then increases, noting that the final increase is more evident for FIFO and LRU when compared against LFU, LFUDA and LRUK, in agreement with our previous results with synthetic and real world traces.

IX. Practical Implications

Next, we discuss the practical implications of our results from a system operation point of view. In particular, femto-caching systems typically deploy high storage capacity caches at base stations to compensate for the weak backhaul capacity (see Figure 1). We envision that rejuvenation, monitoring, measurements and resource separation are key elements in

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

TABLE V
SUMMARY OF TRAGEN TRACES STATISTICS

		EU Trace									
	Video (V)	Media-0	Media-1	Media-2	Media-3	Media-4	Media-5	Media-6	Web-7	Web-8	Web-9
Length (mil. reqs)	596	32.04	109.3	70.3	91.92	43.98	66.48	36.56	9.73	128.44	6.95
Req. rate (reqs/sec)	382	20.64	70.44	45.32	59.2	28.33	42.82	23.55	6.248	82.73	5.38
Traffic (MBps)	1536	12	480	13	36	288.3	434.8	26.8	0.8	27.682	0.756
No. of objects (mil.)	127	15.55	2.66	18.62	39.64	2.31	2.49	14.45	0.028	22.56	0.02
Avg. object size (KB)	1756	679.2	9727	286.4	653	10286	10291	1026	71.65	151.3	69.83

the management of those systems that can benefit from the insights derived from this paper, as further detailed below.

- *deploy rejuvenation:* as caches tend to synchronize and age, degrading system performance, one possible countermeasure consists in rejuvenating the system from time to time, replacing contents to re-establish favorable initial conditions to "warmup" the system (Section IV and Theorem 1);
- *monitor diversity:* it is key to monitor content diversity across caches, to avoid the negative side-effects of eventual synchronization as aging (Theorem 2);
- *integrate caching and networking measurements:* the diversity versus replication tradeoff implies that network quality of service and cache states should be analyzed together, e.g., to determine the best rejuvenation rate and replacement policies (Theorem 3);
- *separation versus pooling of cache resources:* in a coarse time scale, separation of cache resources across geographically disperse nodes should be considered, specially when robustness is needed against channel unreliability. Alternatively, if separation is deployed and pooling is not feasible, alternatives to increase content diversity and hit probability include establishing cache policy diversity across caches and randomization at the network or cache policy levels (Sections VII and VIII).

X. CONCLUSION

In this paper we considered the interplay between cache replacement policy, channel reliability and cache partitioning. We showed that the impact of reliability and cache partitioning on cache performance depends to a large extent on the cache replacement policy, and thus partitioning and the replacement policy should be designed jointly for optimal performance.

We identified different sources of content diversity that can be leveraged to avoid cache aging due to the synchronization across caches. In particular, our results indicate that channel unreliability is a source of randomness that can contribute to content diversity, countering cache aging, promoting rejuvenation, and complementing diversity due to the initial cache configuration and cache replacement policy. Our numerical and experimental results validate the analytical findings on large-scale realistic workloads, and indicate that there is a large design space left to be explored for optimizing cache performance over unreliable channels.

Future work. Our results indicate that a policy that deliberately discards a certain percentage of requests based on a threshold balancing robustness and diversity can address the problem of eventual cache synchronization. This observation opens up a number of interesting directions for future research, e.g., leveraging machine learning-based replacement policies that can adapt to network conditions. In this vein, one direction of future research consists of developing a reinforcement learning approach for unreliable channels, extending previous work in that domain [1], [52]–[54]. Learning could eliminate the need for manually setting thresholds to change the network configuration, e.g., in response to changes in network reliability.

Another direction consists of investigating the scalability of the proposed approach and its performance in large-scale distributed cache systems, including distributed approaches for managing the threshold, e.g., using multi-agent learning, under a diverse set of scenarios, including different types of networks, content, and traffic patterns.

ACKNOWLEDGMENT

This work was partially supported by the São Paulo Research Foundation (FAPESP), grant 2021/00199-8, CPE SMARTNESS, and Grant 2020/04031-1. It was also financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brasil (CAPES), Finance Code 001, by The National Education and Research Network (RNP), by CNPq and by FAPERJ under Grants 26/201.376/2021, 315110/2020-1, E-26/211.144/2019 and JCNE/E-26/203.215/2017. G. Dán was partly funded by the Vinnova Center for Trustworthy Edge Computing Systems and Applications (TECoSA) and the Swedish Research Council through project 2020-03860.

REFERENCES

[1] Q. Fan, X. Li, J. Li, Q. He, K. Wang, and J. Wen, "Pa-cache: Evolving learning-based popularity-aware content caching in edge networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1746–1757, 2021.

[2] K. Poularakis, G. Iosifidis, I. Pefkianakis, L. Tassiulas, and M. May, "Mobile data offloading through caching in residential 802.11 wireless networks," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 71–84, 2016.

[3] J. Shuja, K. Bilal, W. Alasmary, H. Sinky, and E. Alanazi, "Applying machine learning techniques for caching in next-generation edge networks: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 181, p. 103005, 2021.

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

[5] G. Paschos, G. Iosifidis, and G. Caire, "Cache optimization models and algorithms," *arXiv:1912.12339*, 2019.

[6] Z. Liu, P. Nain, N. Niclausse, and D. Towsley, "Static caching of web servers," in *Multimedia Computing and Networking 1998*, vol. 3310. International Society for Optics and Photonics, 1997, pp. 179–190.

[7] M. Dehghan, W. Chu, P. Nain, D. Towsley, and Z.-L. Zhang, "Sharing cache resources among content providers: A utility-based approach," *IEEE/ACM Trans. on Networking*, vol. 27, no. 2, pp. 477–490, 2019.

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

[8] R. Karedla, J. S. Love, and B. G. Wherry, "Caching strategies to improve disk system performance," *Computer*, vol. 27, no. 3, pp. 38–46, 1994.

[9] G. Quan, J. Tan, and A. Eryilmaz, "Counterintuitive characteristics of optimal distributed lru caching over unreliable channels," *IEEE/ACM Transactions on Networking*, 2020.

[10] G. Hasslinger, "Markov analysis of optimum caching as an equivalent alternative to Belady's algorithm without look-ahead," in *International Conference on Measurement, Modelling and Evaluation of Computing Systems*. Springer, 2018, pp. 35–52.

[11] M. Dehghan, L. Massoulie, D. Towsley, D. S. Menasche, and Y. C. Tay, "A utility optimization approach to network cache design," *IEEE/ACM Trans. on Netw.*, vol. 27, no. 3, pp. 1013–1027, 2019.

[12] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal routing and content caching in heterogeneous networks," in *Proc. of IEEE INFOCOM*, 2015, pp. 936–944.

[13] V. Pacifici and G. Dán, "Coordinated selfish distributed caching for peering content-centric networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3690–3701, 2016.

[14] P. N. Muralidhar, D. Katyal, and B. S. Rajan, "Maddah-Ali-Niesen scheme for multi-access coded caching," in *2021 IEEE Information Theory Workshop (ITW)*. IEEE, 2021, pp. 1–6.

[15] F. Brunero and P. Elia, "The exact load-memory tradeoff of multi-access coded caching with combinatorial topology," *arXiv preprint arXiv:2202.03039*, 2022.

[16] R. Nakamura and N. Kamiyama, "Content availability at network failure in information-centric networking," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3889–3899, 2021.

[17] G. I. Ricardo, A. Tuholukova, G. Neglia, and T. Spyropoulos, "Caching policies for delay minimization in small cell networks with coordinated multi-point joint transmissions," *IEEE/ACM Transactions on Networking*, 2021.

[18] G. Ricardo, G. Neglia, and T. Spyropoulos, "Caching policies for delay minimization in small cell networks with joint transmissions," in *ICC*. IEEE, 2020, pp. 1–6.

[19] G. Neglia, E. Leonardi, G. Iecker, and T. Spyropoulos, "A swiss army knife for dynamic caching in small cell networks," *arXiv:1912.10149*, 2019.

[20] W. Chu, Z. Yu, J. C. Lui, and Y. Lin, "Jointly optimizing throughput and content delivery cost over lossy cache networks," *IEEE Transactions on Communications*, 2021.

[21] A. Sabnis and R. K. Sitaraman, "Tragen: a synthetic trace generator for realistic cache simulations," in *Proceedings of the 21st ACM Internet Measurement Conference*, 2021, pp. 366–379.

[22] M. Perham, "Slabs, pages, chunks and memcached," 2009, https://www.mikeperham.com/2009/06/22/slabs-pages-chunks-and-memcached/.

[23] A. Araldo, G. Dán, and D. Rossi, "Caching encrypted content via stochastic cache partitioning," *IEEE/ACM Trans. on Netw.*, vol. 26, no. 1, pp. 548–561, Feb. 2018.

[24] J. Tan, G. Quan, K. Ji, and N. Shroff, "On resource pooling and separation for LRU caching," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 1, pp. 1–31, 2018.

[25] K. Ji, G. Quan, and J. Tan, "Asymptotic miss ratio of LRU caching with consistent hashing," in *Proc. of IEEE INFOCOM*, 2018, pp. 450–458.

[26] A. O. Al-Abbasi and V. Aggarwal, "Ttlcache: taming latency in erasure-coded storage through ttl caching," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1582–1596, 2020.

[27] A. Avritzer, J. Kondek, D. Liu, and E. J. Weyuker, "Software performance testing based on workload characterization," in *Proceedings of the 3rd International Workshop on Software and Performance*, 2002, pp. 17–24.

[28] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy, "Performance evaluation of the random replacement policy for networks of caches," *Performance Evaluation*, vol. 72, pp. 16–36, 2014.

[29] S. Tarnoi, K. Suksomboon, W. Kumwilaisak, and Y. Ji, "Performance of probabilistic caching and cache replacement policies for content-centric networks," in *39th Annual IEEE Conference on Local Computer Networks*. IEEE, 2014, pp. 99–106.

[30] E. Newberry and B. Zhang, "On the power of in-network caching in the Hadoop distributed file system," in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, 2019, pp. 89–99.

[31] L. Chhangte, N. Karamchandani, D. Manjunath, and E. Viterbo, "Towards a distributed caching service at the wifi edge using wi-cache," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4489–4502, 2021.

[32] T. Nie, J. Luo, L. Gao, F.-C. Zheng, and L. Yu, "Cooperative edge caching in small cell networks with heterogeneous channel qualities," in *Vehicular Technology Conference*. IEEE, 2020, pp. 1–6.

[33] G. Quan, J. Tan, A. Eryilmaz, and N. Shroff, "A new flexible multi-flow lru cache management paradigm for minimizing misses," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 2, pp. 1–30, 2019.

[34] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello, "Push-to-peer video-on-demand system: Design and evaluation," *IEEE JSAC*, vol. 25, no. 9, pp. 1706–1716, 2007.

[35] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," *ACM SIGMETRICS Performance Evaluation Review*, vol. 44, no. 1, pp. 113–124, 2016.

[36] P. Peng, M. Noori, and E. Soljanin, "Distributed storage allocations for optimal service rates," *arXiv preprint arXiv:2102.04322*, 2021.

[37] P. Peng, E. Soljanin, and P. Whiting, "Diversity/parallelism trade-off in distributed systems with redundancy," *arXiv:2010.02147*, 2020.

[38] Y. Li, H. Xie, Y. Wen, C.-Y. Chow, and Z.-L. Zhang, "How much to coordinate? optimizing in-network caching in content-centric networks," *IEEE Transactions on network and service management*, vol. 12, no. 3, pp. 420–434, 2015.

[39] S. H. Strogatz and I. Stewart, "Coupled oscillators and biological synchronization," *Scientific American*, no. 6, pp. 102–109, 1993.

[40] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. H. Strogatz, "Distributed synchronization in wireless networks," *IEEE Signal Processing Magazine*, vol. 25, no. 5, pp. 81–97, 2008.

[41] S. H. Strogatz, *Sync: How order emerges from chaos in the universe, nature, and daily life*. Hachette UK, 2012.

[42] P. Sena, I. Carvalho, A. Abelem, G. Dán, D. Menasche, and D. Towsley, "Caching policies over unreliable channels," in *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*. IEEE, 2020, pp. 1–8.

[43] E. J. Rosensweig, D. S. Menasche, and J. Kurose, "On the steady-state of cache networks," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 863–871.

[44] W. Caarls, E. Hargreaves, and D. S. Menasché, "Q-caching: an integrated reinforcement-learning approach for caching and routing in information-centric networks," *arXiv preprint arXiv:1512.08469*, 2015.

[45] S. Jung and S. Bahk, "Online control of traffic split and distributed cell group state decisions for multi-connectivity in 5g and beyond," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 3, pp. 2843–2858, 2021.

[46] G. Quan, J. Tan, and A. Eryilmaz, "Counterintuitive characteristics of optimal distributed lru caching over unreliable channels," *IEEE/ACM Transactions on Networking*, vol. 28, no. 6, pp. 2461–2474, 2020.

[47] S. Safavat, N. N. Sapavath, and D. B. Rawat, "Recent advances in mobile edge computing and content caching," *Digital Communications and Networks*, vol. 6, no. 2, pp. 189–194, 2020.

[48] M. I. Zulfa, R. Hartanto, and A. E. Permanasari, "Caching strategy for web application–a systematic literature review," *International Journal of Web Information Systems*, vol. 16, no. 5, pp. 545–569, 2020.

[49] D. S. Berger, R. K. Sitaraman, and M. Harchol-Balter, "Adaptsize: Orchestrating the hot object memory cache in a content delivery network," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 483–498.

[50] Z. Song, D. S. Berger, K. Li, A. Shaikh, W. Lloyd, S. Ghorbani, C. Kim, A. Akella, A. Krishnamurthy, E. Witchel *et al.*, "Learning relaxed Belady for content distribution network caching," in *NSDI*, 2020, pp. 529–544.

[51] G. Vietri, L. V. Rodriguez, W. A. Martinez, S. Lyons, J. Liu, R. Rangaswami, M. Zhao, and G. Narasimhan, "Driving cache replacement with ml-based lecar," in *HotStorage*, 2018.

[52] Y. Liu, J. Jia, J. Cai, and T. Huang, "Deep reinforcement learning for reactive content caching with predicted content popularity in three-tier wireless networks," *IEEE Transactions on Network and Service Management*, 2022.

[53] A. Lekharu, M. Jain, A. Sur, and A. Sarkar, "Deep learning model for content aware caching at mec servers," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1413–1425, 2021.

[54] S. Qiu, Q. Fan, X. Li, X. Zhang, G. Min, and Y. Lyu, "Oa-cache: Oracle approximation based cache replacement at the network edge," *IEEE Transactions on Network and Service Management*, 2023.

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3334559

Paulo Sena received his B.S degree in Computer Science from the Federal University of Pará in Belém, capital of the Brazilian state of Pará, in 2016. He obtained his M.Sc. degree in Computer Science from the Federal University of Pará, in 2019. Currently, he is pursuing his Ph.D. degree from the Federal University of Pará. He is full professor of Computing Faculty at Estácio de Sá University. He is a researcher member of the Laboratory in Computer Network and Multimedia Communication (GERCOM). His research interests are focused on the future internet, information-centric network, performance analysis, caching systems, and the internet of things.

Antonio Abelem is full Professor in the Computer Science Faculty at the Federal University of Pará (UFPA) in Belém, capital of the Brazilian state of Pará. He holds a Ph.D. in Computer Science from the Catholic University of Rio de Janeiro (PUC-Rio, 2003). He is an associate and member of the Board of the Brazilian Computer Society (SBC). He coordinates the Research Laboratory in Computer Network and Multimedia Communication (GERCOM) from which participates and coordinates several national and international projects. He is a TPC member and reviewer of national and international conferences and journals. His research is focused on future internet, software-defined networking, network function virtualization, internet of things, performance analysis, and network security. In 2020, he was visiting professor at the University of Massachusetts (UMass), in Amherst-MA, where he carried out research on quantum networks.

György Dán is a Professor at the Department of Network and Systems Engineering at KTH. His research focuses on the design, modeling and evaluation of networked systems for distributed storage, computing and control. His research group works on resource allocation and learning problems for performance, resilience and security risk management with applications in content distribution, critical infrastructures, and mobile computation offloading.

Daniel Sadoc Menasché received the Ph.D. degree in Computer Science from the University of Massachusetts, Amherst, in 2011. Currently, he is an Associate Professor with the Computer Science Department, Federal University of Rio de Janeiro, Brazil. His interests are in modeling, analysis, security and performance evaluation of computer systems, being a recipient of best paper awards at GLOBECOM 2007, CoNEXT 2009, INFOCOM 2013 and ICGSE 2015. He is an alumni affiliated member of the Brazilian Academy of Sciences.

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on November 22,2023 at 08:24:24 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.