# On the Interaction between Content Caching and Request Assignment in Cellular Cache Networks

Kolar Purushothama Naveen, Laurent Massoulié, Emmanuel Baccelli, Aline Carneiro Viana, Don Towsley

# On the Interaction between Content Caching and Request Assignment in Cellular Cache Networks

K.P. Naveen
INRIA Saclay, Campus de l'Ecole Polytechnique
Palaiseau 91120, France
naveenkp@inria.fr

Laurent Massoulie
MSR-INRIA, Campus de l'Ecole Polytechnique
Palaiseau 91120, France
laurent.massoulie@inria.fr

Emmanuel Baccelli
INRIA Saclay, Campus de l'Ecole Polytechnique
Palaiseau 91120, France
emmanuel.baccelli@inria.fr

Aline Carneiro Viana*
INRIA Saclay, Campus de l'Ecole Polytechnique
Palaiseau 91120, France
aline.viana@inria.fr

Don Towsley
Dept. of Computer Science
University of Massachusetts
Amherst MA 01003, USA
towsley@cs.umass.edu

## ABSTRACT

The potential availability of storage space at cellular and femtocell base-stations (BSs) raises the following question: How should one optimize performance through both load balancing and content replication when requests can be sent to several such BSs? We formally introduce an optimization model to address this question and propose an online algorithm for dynamic caching and request assignment. Crucially our request assignment scheme is based on a server price signal that jointly reflects content and bandwidth availability. We prove that our algorithm is optimal and stable in a limiting regime that is obtained by scaling the arrival rates and content chunking. From an implementation standpoint, guided by the online algorithm we design a light-weight scheme for request assignments that is based on load and cache-miss cost signals; for cache replacements, we propose to use the popular LRU (Least Recently Used) strategy. Through simulations, we exhibit the efficacy of our joint-price based request assignment strategy in comparison to the common practices of assigning requests purely based on either bandwidth availability or content availability.

## CCS Concepts

**•Networks → Network performance modeling; Network simulations; •Theory of computation → Design and analysis of algorithms;**

## Keywords

Cellular cache networks; online algorithm; fluid approximations; Lyapunov stability; LRU caches.

*This work was supported by the EU FP7 ERANET program under grant CHIST-ERA-2012 MACACO.

## 1. INTRODUCTION

Mobile data traffic is expected to increase by more than 57% annually to 24.3 exabytes per month by 2019 [1]. This increase will likely result in durably congested wireless access at the edge despite advances in radio technologies. Congestion is also anticipated at the core, driven not only by mobile Internet access but also by the emergence of new cloud services and Machine-to-Machine (M2M) communications.

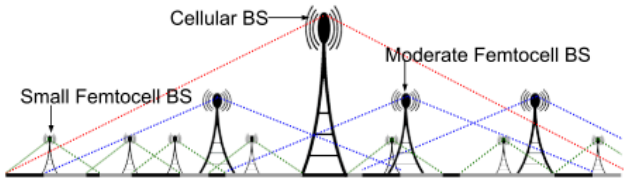


**Figure 1: A cellular network with a dense deployment of heterogeneous base-stations (BS).**

To alleviate congestion at the Internet edge, one promising approach is to target denser deployments of wireless base-stations (BSs), such as cache enabled femtocells in addition to the existing cellular BSs [12, 18]. Such a scenario is depicted in Figure 1, where the alternating black and gray horizontal sections represent regions that are within the range of distinct subset of BSs. Thus, mobile users, depending on their respective locations, will be potentially connected to several BSs from which content may be directly downloaded. In this context, BSs can be heterogeneous in terms of bandwidth and memory capacities. For instance, a nearby femtocell BS will provide more bandwidth than the heavily loaded cellular BS; however, the chance of finding a desired content at the femtocell is lower due to its lower cache capacity.

Thus, both bandwidth and content availability critically affect "by which BS" and "with which performance" incoming requests for contents can be served. In turn, content request assignment strategies impact targeted BSs through both their bandwidth loads and cache contents as they trigger cache update decisions. The goal of this paper is to understand this intricate interplay between content request assignment strategies, bandwidth load, and cache management in cellular cache networks. Specifically, we strive to

determine routing mechanisms and associated cache update rules that together achieve a target trade-off between cache-miss probability and bandwidth loads.

**Related Work:** The problem of caching at femtocell base stations or heterogeneous caches has been recently studied in [10, 12, 18]. Authors in [12] address content placement at the femtocell base stations to maximize the probability of cache hit. A similar setup, but with mobile users, is studied in [18]. In both these works, request routing is purely based on content availability (so that cache hits are maximized), while ignoring the bandwidth costs at the base stations. Further, the authors focus on static content placement algorithms. For instance, in [12] the content placement problem is formulated as that of maximizing a monotone submodular function under matroid constraints; a static greedy algorithm that is within a constant factor of the optimal is proposed.

Borst et al. in [7] address the problem of content placement and routing to optimize the total bandwidth cost, but their formulation is limited to linear cost functions, in contrast to our work where we allow for general convex costs. Moreover, the focus in [7] is again on designing static algorithms. Similarly, there are other works in the literature addressing the problem of static content placement so as to optimize some linear system cost [4, 8, 13].

There is an extensive literature on online cache management algorithms and in particular on LRU (Least Recently Used) caches [5, 9, 11]. There is also an abundant literature on dynamic request routing [6, 15], but very few work address the design of online algorithms for both routing and cache management, and the interaction between the two. Notable exceptions are the works on cache networks [2, 3], which focus on stabilizing queues of pending requests by joint management of bandwidth and caches. In contrast we instead focus on loss-based models and on the objective of cost minimization rather than queue stabilization.

With a similar focus as ours, the authors in [14] study the problem of cache replication and request forwarding in CDNs comprising distributed caches, e.g., household set-top boxes. However, the objective is again limited to linear traffic costs. Moreover, they focus on scenarios with many servers with identical characteristics. This is in contrast with our work as we consider a system with few potentially heterogeneous caches; our limiting regime is instead obtained by scaling the arrival rates and the chunk size. There are other work in the literature, e.g., [16, 19], where proportional placement strategies and matching algorithms for routing requests to content available caches are studied. These differ again from our present work by focusing on scenarios with many identical servers.

## 2. SYSTEM MODEL

Consider a system comprising a finite collection of *base-stations (BSs)* $\mathcal{S}$ and *locations* $\mathcal{L}$ where a location $\ell \in \mathcal{L}$ can be thought of as representing the set of all physical locations within the range of a subset of BSs. For instance, in Figure 1 the alternating black and grey shaded horizontal segments represents different locations. Thus, each location is connected to a subset of the BSs. These interconnections can be effectively depicted as in Figure 2, where location $\ell$ is connected to BSs $s$ and $s'$ while $\ell'$ is connected to $s'$ and $s''$. Let $A$ denote the adjacency matrix representing the interconnections, i.e., $A_{\ell s} = 1$ implies that location $\ell$ is within

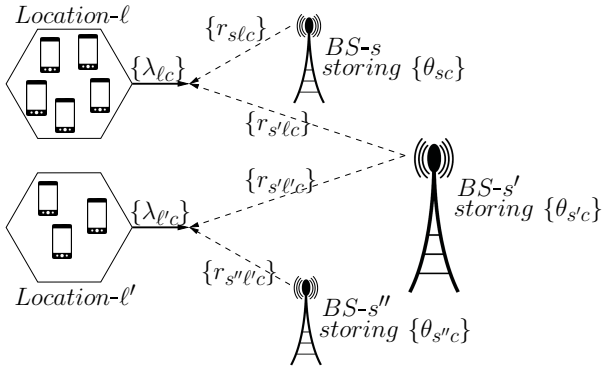


**Figure 2: Illustration of the interconnections between BSs and locations.**

the range of BS $s$; $A_{\ell s} = 0$ otherwise. We assume a finite set $\mathcal{C}$ of *contents* in the system with the size of each content being one unit. We allow *content chunking* whereby a content can be completely reconstructed using $k$ independently coded packets of the content. Each BS $s \in \mathcal{S}$ includes a cache that can store up to $M_s$ units of contents (*memory constraints*). We allow the BSs to store chunks of each content instead of complete copies. We assume that the chunks stored at different servers are always independent so that it only suffices to download $k$ chunks from any subset of BSs to completely reconstruct the content.

Requests for content $c$ arrive at location $\ell$ according to a Poisson process with rate $\lambda_{\ell c}^{(k)} = k\lambda_{\ell c}$, which can be thought as the aggregate rate for $c$ from of all users stationed at $\ell$. The number of chunks of content $c$, available at server $s$, restricts the rate at which $s$ can serve incoming requests for $c$ (*content-availability constraints*). Content download requests, if accepted, result in service at unit rate. In this sense we consider a loss model (in contrast to a queuing model).

The bandwidth availability at a server $s \in \mathcal{S}$ is reflected using a *bandwidth-cost function*, $C_s^{(k)}(r)$, which represents the cost incurred by $s$ for serving requests at a total rate of $r$. We assume that $C_s^{(k)}$, $k \geq 1$, can be expressed as $C_s^{(k)}(r) = C_s(\frac{r}{k})$, where $C_s$, $s \in \mathcal{S}$, is strictly convex and increasing. We refer to the above system as the *$k$-th system*.

To obtain a benchmark for performance comparison, we consider a system where the caching and routing variables are relaxed to be reals. Let $\theta_{sc} \in [0, 1]$ denote the fraction of content $c$ stored at BS $s$, and $kr_{s\ell c}$ the rate at which $s$ serves requests for $c$ from $\ell$. Then, the memory constraint at BS $s \in \mathcal{S}$ is given by $\sum_c \theta_{sc} \leq M_s$. Next, since only a fraction $\theta_{sc}$ of a request for $c$ can be served by $s$, we have the following content availability (CA) constraints: $kr_{s\ell c} \leq A_{\ell s} k\lambda_{\ell c}\theta_{sc}$, $s \in \mathcal{S}, \ell \in \mathcal{L}, c \in \mathcal{C}$, where we have approximated the number of ongoing requests (of content $c$ at $\ell$) by the steady state arrival rate, $k\lambda_{\ell c}$ (assuming an average service time of one).

In order to obtain an unconstrained optimization formulation, we replace the above hard memory and CA constraints with strictly convex and increasing penalty functions, $\widehat{C}_s$ and $C_{s\ell c}$, in the cost to be minimized[1]. $\widehat{C}_s(\theta)$ denotes the cost of storing $\theta$ units of contents at BS $s$. For the CA constraint, denoting $x := kr_{s\ell c} - k\lambda_{\ell c}\theta_{sc}$, $C_{s\ell c}(\frac{x}{k})$ represents the cost of respecting ($x$ negative) the CA constraint, or

[1]Formulation directly involving the hard constraints is available in our technical report [17].

exceeding the constraint ($x$ positive) by $x$ units. Since the BSs are physically constrained to always satisfy the CA constraints, a positive $x$ represents the content cache-miss rate from BS $s$ for requests from $\ell$ for $c$. Thus, $C_{s\ell c}(\frac{x}{k})$ represents the cost incurred for fetching the missed content rate from the main data center at the backend. Another possibility is to let $C_{s\ell c}(\frac{x}{k}) = b\max(\frac{x}{k}, 0)$ for some $b > 0$, which can be interpreted as a cost of $b$ per cache miss.

We now define the *total cost* incurred as,

$$C(\{\theta_{sc}\},\{r_{s\ell c}\}) = \sum_s C_s\Big(\sum_{\ell c} A_{\ell s} r_{s\ell c}\Big) + \sum_s \widehat{C}_s\Big(\sum_c \theta_{sc}\Big) + \sum_{s\ell c} A_{\ell s} C_{s\ell c}\Big(r_{s\ell c} - \lambda_{\ell c}\theta_{sc}\Big) \quad (1)$$

and propose the following optimization problem:
**Total Cost (TC):**

$$\begin{aligned}
\text{Minimize:} \quad & C(\{\theta_{sc}\},\{r_{s\ell c}\}) && \text{(2a)}\\
\text{Over:} \quad & \theta_{sc} \in [0,1], s \in \mathcal{S}, c \in \mathcal{C} && \text{(2b)}\\
& r_{s\ell c} \geq 0, s \in \mathcal{S}, \ell \in \mathcal{L}, c \in \mathcal{C} && \text{(2c)}\\
\text{Subject to:} \quad & \sum_s A_{\ell s} r_{s\ell c} = \lambda_{\ell c}, \ell \in \mathcal{L}, c \in \mathcal{C}. && \text{(2d)}
\end{aligned}$$

The equality constraint (2d) is used to ensure that all incoming requests are handled by the BSs. In Section 3, we propose an *online algorithm* to solve this optimization problem. We show that our algorithm converges to the optimal solution of (2) as the scale of the system in terms of arrival rate and number of chunks per content (i.e., $k$) increases.

# 3. ONLINE ALGORITHM

The main objective of this section is to propose an algorithm for assigning incoming requests to BSs based on their current "prices"; the key idea is that these prices reflect both the load as well as the content availability at the respective BSs (which is in contrast to the common practice of assigning requests purely based on either load or content availability). Due to space constraints, we only briefly mention about the optimality and stability properties of our algorithm; theoretical details and proofs are available in our technical report [17].

Let $\Theta_{sc}(t)$ denote the number of chunks of content $c$ stored at BS $s$ at time $t$, and $G_{slc}(t)$ the total number of chunks of content $c$ served (possibly for more than one request) from $s$ to $\ell$ at time $t$. Thus, $\Theta_{sc}(t)/k$ is the fraction of content $c$ stored at $s$ at time $t$, and $R_{slc}(t) := G_{slc}(t)/k$ is the rate at which $s$ serves requests for $c$ from $\ell$. The chunk service times are i.i.d (independent and identically distributed) exponential random variables with unit rate.

**Cache Updates:** Updating a BS's cache for a content involves either adding new chunks or removing some existing chunks. In the $k$-th system, let $k\nu_{sc}$ denote the rate at which BS $s$ updates the cache for content $c$. Suppose BS $s$ initiates a cache update for $c$ at time $t$. Then, the number of chunks that are added or removed is proportional to

$$\Delta\Theta_{sc}(t) \quad = \quad -Q_s(t) + \sum_\ell A_{\ell s}\lambda_{\ell c} P_{s\ell c}(t), \quad (3)$$

where (using $F'$ to denote the derivative of $F$),

$$Q_s(t) \quad := \quad \widehat{C}'_s\Big(\sum_c \frac{\Theta_{sc}(t)}{k}\Big) \quad (4)$$

$$P_{s\ell c}(t) \quad := \quad C'_{s\ell c}\Big(\frac{R_{s\ell c}(t)}{k} - \lambda_{\ell c}\frac{\Theta_{sc}(t)}{k}\Big). \quad (5)$$

Formally, $\Theta_{sc}(t)$ is updated as follows: $\Theta_{sc}(t^+) = \Theta_{sc}(t) + [\epsilon\Delta\Theta_{sc}(t)]$, where $t^+$ denotes the time instant just after $t$, $\epsilon > 0$ is a constant, and the notation $[x]$ represents the integer closest to $x$. We refer to $Q_s(t)$ and $P_{s\ell c}(t)$, respectively, as the *memory price* and the *content availability (CA) price* (w.r.t $(\ell, c)$) of BS $s$ at time $t$.

*Remark:* For the sake of analysis, we assume that the caches are updated instantaneously at the update instants, although in practice a small time is required to download new chunks from a backend data center. We further assume that the backend data center always supplies a new batch of independently coded chunks, so that it is possible to reconstruct a content by downloading $k$ chunks from any subset of BSs, without worrying about chunks being dependent.

Note that $\Delta\Theta_{sc}(t)$ is simply the negative of the gradient (w.r.t $\theta_{sc}$) of the cost function in (1). Thus, our cache update strategy is essentially the gradient descent algorithm. The novelty of our work appears next, where we propose a request assignment strategy that couples the update instants of $\{R_{s\ell c}(t)\}$ variables with the request arrival times. Further, incoming requests are assigned to download chunks from a (state dependent) selected subset of BSs, thus always increasing the current $R_{s\ell c}(t)$ values of the selected BSs. This differs from the gradient descent algorithm where, depending on the gradient, an update could also recommend decreasing the value of some rate variables. We formally discuss our request assignment strategy next.

**Request Assignments:** Suppose a request for content $c$ arrives at location $\ell$ at time $t$. Then, the decision as to how many chunks should be downloaded from different BSs is based on the "prices" associated with the BSs connected to $\ell$. Formally, for each $(\ell, c)$, we define the price of BS $s$ (such that $A_{\ell s} = 1$) at time $t$ as,

$$\Pi_{s\ell c}(t) \quad = \quad P_s(t) + P_{s\ell c}(t) \quad (6)$$

where, $P_{s\ell c}(t)$ is the CA price (recall (5)) and

$$P_s(t) \quad := \quad C'_s\Big(\sum_{\ell' c'} A_{\ell' s}\frac{R_{s\ell' c'}(t)}{k}\Big)$$

represents the *bandwidth price* of BS $s$.

*Remark:* Although the price depends on various caching and routing variables, its calculation in practice can be based on, for instance, an estimate of the current load at a BS to compute its bandwidth price; The content availability price can be simplified to reflect the cost of fetching the unavailable chunks from the backend data center. We will study the performance of such a light-weight scheme in Section 4.

Now, given the BS prices, let $\mathcal{S}_{\ell c}(t)$ denote the set of all minimum price BSs reachable from $\ell$, i.e.,

$$\mathcal{S}_{\ell c}(t) = \operatorname*{argmin}_s\Big\{\Pi_{s\ell c}(t) : A_{\ell s} = 1\Big\}. \quad (7)$$

Our request assignment strategy centers on downloading sufficient chunks of $c$ from the BSs in $\mathcal{S}_{\ell c}(t)$ to construct the content at $\ell$. However, if there are an insufficient number of chunks at these BSs, signals in the form of *dummy chunk* assignments are issued to increase their CA price and thus trigger future replication of content $c$ chunks at these BSs. Formal details follow, where we have chosen to distribute the dummy chunks equally among all the BSs in $\mathcal{S}_{\ell c}(t)$ (see (8a) and (8a)); however, note that any other dummy chunk

assignment would work equally well, as long as the total number of chunks allocated (actual+dummy) is $k$.

Let $n$ denote the number of BSs in $\mathcal{S}_{\ell c}(t)$ (i.e., $n = |\mathcal{S}_{\ell c}(t)|$). Now, arrange the BSs in $\mathcal{S}_{\ell c}(t)$ in any arbitrary order, for instance, e.g., in increasing order of the BS IDs (assuming that each BS is given a unique integer ID). Suppose $(s_1, s_2, \cdots, s_n)$ is such an arrangement, then BS $s_i$ is assigned to serve $\Gamma_{s_i\ell c}(t) = \Gamma^{(1)}_{s_i\ell c}(t) + \Gamma^{(2)}_{s_i\ell c}(t)$ chunks (actual+dummy, resp.) of content $c$, where (using $\lfloor x \rfloor$ to denote the largest integer smaller than $x$), for $i = 1, \cdots, n$,

$$\begin{aligned}
\Gamma^{(1)}_{s_i\ell c}(t) &= \min\Big\{k - \sum_{j=1}^{i-1} \Gamma^{(1)}_{s_j\ell c}(t), \Theta_{s_i c}(t)\Big\}, \\
\Gamma^{(2)}_{s_i\ell c}(t) &= \left\lfloor \frac{k - \sum_{j=1}^{n} \Gamma^{(1)}_{s_j\ell c}(t)}{n} \right\rfloor, i = 1, \cdots, n-1 \\
\Gamma^{(2)}_{s_n\ell c}(t) &= k - \sum_{j=1}^{n} \Gamma^{(1)}_{s_j\ell c}(t) - \sum_{j=1}^{n-1} \Gamma^{(2)}_{s_j\ell c}(t).
\end{aligned}$$

No chunks are downloaded from the BSs not in $\mathcal{S}_{\ell c}(t)$, i.e., we have $\Gamma_{s\ell c}(t) = 0$ for $s \notin \mathcal{S}_{\ell c}(t)$.

Thus, $\Gamma^{(1)}_{s_i\ell c}(t)$ is the actual number of chunks that $s_i$ can serve, while $\sum_{j=1}^{n} \Gamma^{(2)}_{s_j\ell c}(t)$ represents the total number of chunks that are not present in the caches, i.e., results in cache misses. Although, physically it is not possible for $s_i$ to serve $\Gamma^{(2)}_{s_i\ell c}(t)$ number of additional chunks, we proceed to allocate these as dummy chunks served by $s_i$. As mentioned earlier, the idea is to increase the CA price $P_{s_i\ell c}(t)$ (recall (5)), so that more chunks of $c$ will be replicated by $s$ during its subsequent cache updates (see (3)). Practically, the dummy chunks are downloaded from the backend data center.

Finally, the associated rate variables are updated according to the following expression: $R_{s\ell c}(t^+) = R_{s\ell c}(t) + \Gamma_{s\ell c}(t)$.

**Theoretical Guarantees:** Using mean field approximation and Lyapunov function techniques, we show that our online algorithm is optimal and stable in a limiting fluid regime that is obtained by scaling $k$, i.e., the arrival rates and the number of chunks per content; formal details are available in our technical report [17]). From a practical standpoint, our theoretical results imply that for a large system[2] the performance of the online algorithm is expected to converge in time to a close-to-optimal performance.

# 4. SIMULATION EXPERIMENTS

We first consider a simple setting comprising 2 locations and 3 BSs; the content catalog size is $|\mathcal{C}| = 10$. After demonstrating the efficacy of the joint-price based request assignment schemes for this setting, we present results for a larger system.

We assume that BSs 1 and 3 are *femtocells* serving locations 1 and 2, respectively, while BS 2 is a common *cellular BS* connected to both the locations (recall Figure 2). Thus, the requests arriving at a location can be assigned either to the respective femtocell or to the cellular BS. The total

[2]Note that large here refers to a system with large arrival rates and content chunking. It does not imply a large network in terms of number of BSs and locations; all our analysis holds for a network of any finite size.

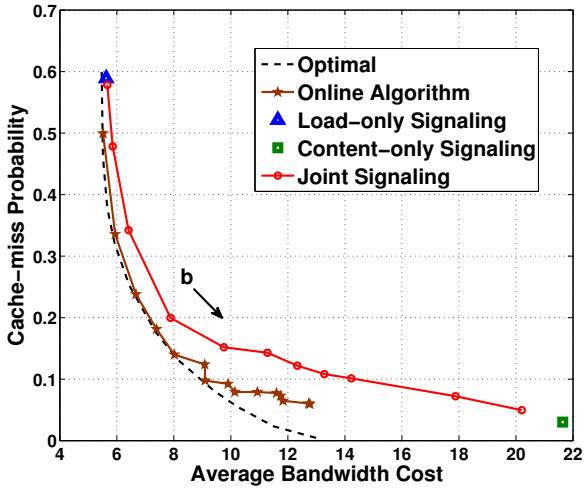


**Figure 3: Performance curves.**

(unscaled) content request rate from each location is 1, i.e., $\sum_c \lambda_{\ell c} = 1$ for $\ell = 1, 2$. The content popularity at each location follows a Zipf distribution with parameter 0.8, i.e., the probability that an incoming request is for the $i$-th most popular content is proportional to $1/i^{0.8}$. In order to obtain a scenario where the same content can have different popularities at different locations, we randomly shuffle the distributions for the two locations. The (unscaled) cache update rate of each BS for each content is 1, i.e., $\nu_{sc} = 1$.

The various costs (recall (1)) used in the simulations are as follows:

• Bandwidth cost: $C_s(r) = \exp(2(r - R_s))$, where $R_1 = R_3 = 0.5$ and $R_2 = 0.1$; thus, the cellular BS is costlier than the femtocell BSs.

• Memory cost: $\widehat{C}_s(\theta) = \exp(5(\theta - M_s))$ where $M_1 = M_3 = 2$ while $M_2 = 8$; thus, the cellular BS has more memory compared with the femtocells.

• Content availability (CA) cost: $C_{s\ell c}(x) = \exp(bx)$; we are particularly interested in studying the effect of $b$ on performance. A large value of $b$, which yields a steep cost function, results in a system where cache misses are costlier than bandwidth. Thus, $b$ can be used to trade off bandwidth cost and cache-miss probability. We refer to $b$ as the *cache-miss exponent*.

**Performance of the online algorithm:** In Figure 3 we first plot the optimal performance curve, obtained by directly solving the problem in (2) for different values of $b$ (curve labeled 'Optimal'). As highlighted in the figure, the cache-miss exponent $b$ increases from left to right along all the curves. Also shown in the figure is the performance of the online algorithm, described in Section 3, for a system with $k = 10$ chunks per content. Although the online algorithm is proven to converge to the optimal performance as $k \to \infty$, it is interesting to observe that its performance is comparable with the optimal already for $k = 10$.

Before proceeding further, let us briefly discuss the trade-off curves. Note that, as $b$ increases the cache-miss probability improves, however at the expense of an increased bandwidth cost. This is because, for a larger $b$, the cellular BS offers a lower price for the incoming requests as it holds more contents. Thus, most of the requests are forwarded to the cellular BS, increasing its (and the overall) bandwidth

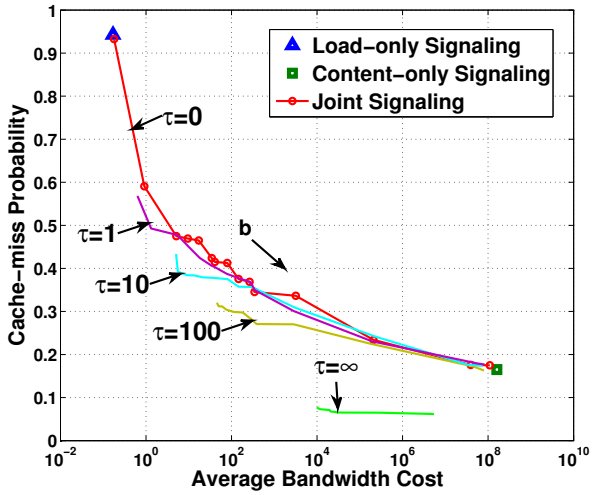


**Figure 4: Performance of the light-weight schemes for a large network.**

cost. In contrast, when $b$ is small, bandwidth cost is the key component in deciding the BSs' prices. Since femtocells are less expensive in this regard, more requests are now forwarded to the femtocells, resulting in more cache misses as they only hold few contents.

**Drawbacks of the online algorithm:** The implementation of the online algorithm however requires the BSs to maintain routing variables in the form of number of chunks of each content downloaded by requests from each location. This would require identifying each chunk with its request's location, which is not possible in practice when the number of ongoing requests are large. Further, estimates of the request arrival rates, $\lambda_{\ell c}$, are required by the BSs to compute their content availability prices. Finally, the cache update in the online algorithm is *server-driven* requiring the BSs to update their caches regularly. This would result in unnecessary updates whenever the request rate is less.

**A light-weight signaling scheme:** In view of the above drawbacks, we modify the online algorithm to obtain a light-weight signaling scheme for request assignments. We refer to it as the *joint-price* scheme since, as in the online algorithm, the request assignment here is based jointly on the content and bandwidth availability. The bandwidth price, as before, remains to be the same convex increase function of the current load on the BS; thus, by using the congestion signals from the BS, it is possible to estimate the bandwidth cost at a location. However, for the content availability price, we now simply use the cost of fetching unavailable chunks if an insufficient number of chunks of the requested content is available at the BS. Formally, the content availability price of BS $s$ for request $c$ (from any location) at time $t$ is given by $b(k - \Theta_{sc}(t))/k$, where $\Theta_{sc}(t)$ is the number of chunks of content $c$ at BS $s$ at time $t$ and $b$ now actually represents the cost of fetching the unavailable fraction of content $c$. Thus, as with the online algorithm, a larger $b$ implies that the cache misses are costly.

We use the above joint price signaling scheme in conjunction with the LRU (Least Recently Used) cache eviction algorithm [5, 9], which is a *request driven* cache management strategy. Under LRU, cache replacements are initiated by incoming requests that do not find the desired content in the cache (a cache-miss event). As a consequence, the least recently used content is removed from the cache to free memory for storing the new content. Since we deal with contents at the chunk level, we have implemented a variation of LRU, referred to as the *CLRU (Chunk LRU)* strategy. In CLRU, the incoming request at a cache for $m$ chunks of a content triggers the relocation of such $m$ chunks to the head of the cache queue to be served. The placement of extra cached chunks at the cache (i.e., if more than $m$ chunks are cached) remain unchanged. In the opposite case, if chunks are missing (i.e., if less than the requested $m$ chunks are cached) and the cache is full, new chunks of the content are downloaded and placed at the head of the queue by removing least recently used chunks at the tail of the queue.

**Performance of joint-price scheme:** In Figure 3 we have depicted the performance of the joint-price signaling scheme. Also shown in the figure are the performance of two heuristic policies (commonly used in practice) where the request assignment is purely based on bandwidth availability or content availability signals (points 'Load-only Signaling' and 'Content-only Signaling', respectively). The joint-price scheme provides a favorable trade off between bandwidth cost and cache-miss probability, encompassing the heuristic policies as its extreme end points.

Although the joint-price scheme suffers some degradation in performance when compared with the online algorithm, it is useful to emphasize that the simplicity of the former scheme would render it more favorite than the latter when it comes to considerations for practical implementation. Moreover, the performance degradation is less for low to moderate values of cache-miss cost $b$, which is typically the case in practice.

**Large network:** Motivated by the observations made in the previous section, we proceed to study the performance of the joint-price scheme for a large network. We now consider a content catalog of size 1000. The number of locations are increased to 10. It will be useful to think of a location $\ell \in \{1, 2, \cdots, 10\}$ as the segment $[0, \ell]$ on the real line (recall Figure 1 for illustration). A cellular base station is located at the center (i.e., at 5) so that, as in the earlier case, it can cover all the locations. However, we now increase the number of (*small-size*) femtocells to 9, which are located at positions $1, 2, \cdots, 9$; each small-size femtocell has a range of 1. Thus, each location (except the end locations) is served by two adjacent femtocells. To model the heterogeneity in the femtocell base stations, we introduce 5 *moderate-size femtocells*. The moderate femtocells have a larger range of 2 and are located at $3, 4, \cdots, 7$. Note that, the inner locations are covered by more femtocells (small+moderate) than the outer ones.

Small-size femtocells have smaller caches (of size 10) compared to moderate-size femtocells (whose cache capacity is 100), and the latter's capacity is smaller than the cellular BS which can store up to $M_s = 800$ contents. Bandwidth cost increases as we move from the small to moderate femtocell and to cellular BS; recalling the cost functions from the previous section, the respective values of $R_s$ are 5, 2.5 and 1.

In Figure 4 we first plot the performance achieved by the joint-price signaling scheme along with the performance of the two heuristic schemes. Again, we observe that the joint-price scheme can achieve a large range of performance values, which is in contrast to the heuristic schemes whose perfor-

mance is fixed at either optimizing bandwidth cost or cache-miss probability, without regarding for the other metric.

In the thus far implemented joint-price schemes, we have been assigning requests to download chunks *strictly* from the set of min-priced servers. We now relax this strict consideration by regarding any BS, whose price is within a *tolerance value* of $\tau$ from the minimum price, as *eligible* to serve chunks for an incoming request (provided as before the eligible BSs are connected to the request's location); chunks are however downloaded by sorting the eligible BSs in the increasing order of their prices. In Figure 4 we have depicted the performance of the joint-price scheme for different values of $\tau$, where the earlier strict min-price approach now corresponds to $\tau = 0$; the case where all connected BSs (irrespective of their prices) are eligible corresponds to $\tau = \infty$.

We observe that as $\tau$ increases the trade-off curves shift downwards, implying that it is possible to achieve a lower cache-miss probability for a given target bandwidth cost. However, the range of trade-off that is possible reduces as $\tau$ increases. For instance, using $\tau = 100$ it is not possible to reduce the bandwidth cost to less than 40; In fact, $\tau = \infty$ case achieves the lowest cache-miss probability, however at the expense of being constrained to operate at a higher range of bandwidth cost. This shift in performance is because, as $\tau$ increases, more BSs become eligible to serve a request so that the cache-miss probability decreases; however, more BSs are now loaded, resulting in a bandwidth scarce system. Thus, further trade-off in performance can be achieved by suitably choosing the value of $\tau$.

In summary, the joint-price scheme, in conjunction with the CLRU policy for cache management, can be a practical candidate for request assignments in futuristic cellular networks where a dense deployment of heterogeneous femtocell base-stations are expected. The cache-miss cost $b$, along with the tolerance value $\tau$, can be used to serve as "tunable knobs" to trade-off one metric for the other.

# 5. CONCLUSION

We proposed an optimization framework to study the problem of cost minimization in cellular cache networks. Towards this direction, we proposed an online algorithm for caching and request assignments which is shown to be optimal and stable in a limiting regime that is obtained by scaling the arrival rates and the content chunking. Based on the online algorithm, we proposed a light-weight joint-price scheme for request assignment that can be used in conjunction with the LRU cache management strategy. Through simulations we found that our joint-price based request assignment strategy outperforms the common practices of routing purely based on either load or content availability. Our proposed joint-price routing mechanisms are thus an appealing candidate combining sound theoretical guarantees with good experimental performance while being simple to implement.

# 6. REFERENCES


[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014-2019. February 2015.

[2] N. Abedini and S. Shakkottai. Content Caching and Scheduling in Wireless Networks With Elastic and Inelastic Traffic. *IEEE/ACM Transactions on Networking*, 2013.

[3] M. Amble, P. Parag, S. Shakkottai, and L. Ying. Content-Aware Caching and Traffic Management in Content Distribution Networks. In *INFOCOM '11*.

[4] I. Baev, R. Rajaraman, and C. Swamy. Approximation Algorithms for Data Placement Problems. *SIAM J. Comput.*, 38(4):1411–1429, August 2008.

[5] G. Bianchi, A. Detti, A. Caponi, and N. Blefari Melazzi. Check Before Storing: What is the Performance Price of Content Integrity Verification in LRU Caching? *SIGCOMM CCR*, July 2013.

[6] T. Bonald, M. Jonckheere, and A. Proutiére. Insensitive Load Balancing. In *ACM SIGMETRICS /PERFORMANCE '04*.

[7] S. Borst, V. Gupta, and A. Walid. Distributed Caching Algorithms for Content Distribution Networks. In *INFOCOM '10*.

[8] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li. Collaborative Hierarchical Caching with Dynamic Request Routing for Massive Content Distribution. In *INFOCOM '12*.

[9] A. Dan and D. Towsley. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes. In *ACM SIGMETRICS '90*.

[10] 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 *to appear at INFOCOM 2015*.

[11] C. Fricker, P. Robert, and J. Roberts. A Versatile and Accurate Approximation for LRU Cache Performance. In *Proceedings of the 24th International Teletraffic Congress, ITC '12*.

[12] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire. FemtoCaching: Wireless Video Content Delivery Through Distributed Caching Helpers. In *INFOCOM '12*.

[13] E. Jaho, M. Karaliopoulos, and I. Stavrakakis. Social Similarity Favors Cooperation: The Distributed Content Replication Case. *IEEE Transactions on Parallel and Distributed Systems*, March 2013.

[14] W. Jiang, S. Ioannidis, L. Massoulié, and F. Picconi. Orchestrating Massively Distributed CDNs. In *ACM CoNEXT '12*.

[15] M. Jonckheere and J. Virtamo. Optimal Insensitive Routing and Bandwidth Sharing in Simple Data Networks. In *ACM SIGMETRICS '05*.

[16] M. Leconte, M. Lelarge, and L. Massoulié. Bipartite Graph Structures for Efficient Balancing of Heterogeneous Loads. In *ACM SIGMETRICS/ PERFORMANCE '12*.

[17] K. P. Naveen, L. Massoulie, E. Baccelli, A. Carneiro Viana, and D. Towsley. On the Interaction between Content Caching and Request Assignment in Cellular Cache Networks. Research Report RR-8707, INRIA Saclay, March 2015.

[18] K. Poularakis and L. Tassiulas. Exploiting User Mobility for Wireless Content Delivery. In *IEEE International Symposium on Information Theory Proceedings, ISIT '13*.

[19] B. Tan and L. Massoulié. Optimal Content Placement for Peer-to-Peer Video-on-Demand Systems. *IEEE/ACM Transactions on Networking*, April 2013.