This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2575837, IEEE Transactions on Mobile Computing

Code, Cache and Deliver on the Move: A Novel Caching Paradigm in Hyper-dense Small-cell Networks

Konstantinos Poularakis and Leandros Tassiulas

Abstract—Caching popular content files at small-cell base stations (SBSs) has emerged as a promising technique to meet the overwhelming growth in mobile data demand. Despite the plethora of work in this field, a specific aspect has been overlooked. It is assumed that all users remain stationary during data transfer and therefore a complete copy of the requested file can always be downloaded by the associated SBSs. In this work, we revisit the caching problem in realistic environments where moving users intermittently connect to multiple SBSs encountered at different times. Due to connection duration limits, users may download only parts of the requested files. Requests for files that failed to be delivered on time by the SBSs are redirected to the coexisting macro-cell. We introduce an optimization framework that models user movements via random walks on a Markov chain aimed at minimizing the load of the macro-cell. As the main contribution, we put forward a distributed caching paradigm that leverages user mobility predictions and innovative information-mixing methods based on the principle of network coding. Systematic experiments based on measured traces of human mobility patterns demonstrate that our approach can offload 65% more macro-cell traffic than existing caching schemes in realistic settings.

Index Terms—Content Caching, Human Mobility Predictability, Network Coding, Markov Chain Model, Small-cell Networks

1 Introduction

1.1 Motivation

We are witnessing an unprecedented worldwide growth of mobile data traffic that is expected to continue at an annual rate of 45 percent over the next years, reaching 30.5 exabytes per month by 2020 [2]. To manage this "data-tsunami", operators deploy small-cell base stations (SBSs), such as pico-cells and femto-cells, that work in conjunction with conventional macro-cell base stations (MBSs), forming the so-called Heterogeneous Cellular Networks (HCNs) [3]. SBSs increase area spectral efficiency and serve the users via short-range energy-prudent communication links.

Caching at SBSs popular content files for which recurring requests are expected has been recently proposed [4] in order to mitigate congestion in the backhaul links that connect the SBSs with the core network. Field trials of such systems revealed that this technique improves the user experienced delay, and at the same time reduces the network servicing cost [5]. Interestingly, the wireless network industry has already begun to commercialize systems that enable SBS caching, such as Data-at-the-Edge^{TM} from Altobridge [6], Nokia Siemens Networks' liquid application [7] and Saguna Networks' Open RAN platform [8]. The key problem in this context is to design the optimal caching policy, i.e., to determine which content files should be placed at each SBS so as to maximize the number of requests that are directly served by the caches. This is a very challenging problem that needs to take into consideration other basic features of the network in order to achieve the best possible outcome/performance. One of them is the mobility of the users, who may move out of an SBS coverage before the data transfer is finished.

Mobility mechanisms for ensuring continuous service of the users are included in the LTE release 8 [9], where the history of cell crossings is analyzed to estimate mobility behavior. Additional, more sophisticated mechanisms have been established to predict users' future connectivity with SBSs in [10], [11]. As the SBS deployment becomes denser [12], [13], mobile users will be more frequently handed off between SBSs. This transition can take place within a few minutes or less considering the typical range of SBS coverage areas; 10-20 meters for femto-cells and less than 200 meters for pico-cells [14]. For a user who is repeatedly moving in and out of the SBS coverage areas, the system can deliver *parts* of the requested file through different SBS caches that the user encounters. If certain file parts fail to be delivered on time by the SBSs, the request is redirected to the macro-cell network. The latter option raises scalability concerns, especially during peak usage hours.

Mobility affects the efficiency of the caching policies. For example, on a several blocks long shopping road, users may frequently move in and out of the coverage areas of two SBSs located a few blocks away one from

• *Konstantinos Poularakis is with the Electrical & Computer Engineering Department, University of Thessaly, Greece. Email: kopoular@uth.gr. Leandros Tassiulas is with the Electrical Engineering Department & Institute for Network Science, Yale University, USA. Email: leandros.tassiulas@yale.edu.* *Part of this work has been accepted to the proceedings of IEEE International Symposium on Information Theory (ISIT), 2013 [1].*

another. These users see a distributed cache that is the union of the two SBS caches. Therefore, there is no benefit from replicating the same file parts at both the SBSs. Disjoint file parts should be cached instead. Clearly, the caching policy should be designed with concerns on predictions about user mobility patterns. Such a consideration adds up to the complexity of the traditional caching problem where optimization is based solely on the anticipated content demand [4], [15]-[22].

1.2 Methodology and contributions

In this work, we revisit the caching problem in realistic HCN environments where moving users intermittently connect to multiple SBSs encountered at different times. The users perform requests for a finite collection of content files, with the respective expected user demand being known as in [4], [15]-[22]. Our methodology exploits predictions about users' future connectivity with SBSs which are achievable through analysis of previous time period mobility statistics [10], [11]. While such predictions are more common in public transportation networks (e.g., buses, trains, etc.), recent studies identified a 93% average predictability on human mobility patterns [23].

We consider *delayed data offloading* [24]. That is, each request is associated with a *deadline* and the user can download parts of the requested file by different encountered SBS caches until the deadline expires. In case that file parts are missing, the request is redirected to the MBS. We first show that the problem of deriving the caching policy that minimizes the MBS load is NP-Hard to approximate within any constant factor. Then, as a first attempt to overcome this difficulty, we present a *Mixed Integer Programming* formulation to find optimal centralized solutions using branch and bound techniques [25].

Following the intuition that the user's future position highly depends on the current one, we model user movements via *random walks on a Markov chain* [26]. Going one step further, we assign weights to the states of the chain representing the amount of "useful" data that a user can download by the SBS caches at each time instance. Here, by "useful" data, we refer to the parts of the requested file which were not previously downloaded by the user. Therefore, the total weight of a walk determines whether the request will be redirected to the MBS or not, and the framework enables the operator to minimize the load of the macro-cell network. Using large deviation inequalities specific to the Markov model, we derive a *distributed caching algorithm* that leverages mobility predictions to minimize the probability that a request reaches the macro-cell network. To better utilize the cache space, our scheme applies *Maximum Distance Separable (MDS) codes* [27], [28] to store at the SBSs encoded versions of the files instead of the raw data packets. In this sense, a file request is completely served when the total amount of (encoded) data downloaded by the user is at least equal to the size of the requested file. This facilitates analysis and can potentially increase the efficiency of content access [29], [30] compared to the traditional case that uncoded file segments are cached [31], [32], [33].

Using measured traces of human mobility patterns in wireless network environments and requests for Youtube videos, we investigate numerically the impact of various parameters on the efficiency of the caching decisions, such as the SBS cache sizes, the delay deadline, the density of SBS deployment and the transmission capacities of SBSs. We find that the proposed algorithm can offload up to 65% more traffic of the macro-cells than conventional caching algorithms in realistic settings. The technical contributions of this work can be summarized as follows:

- *Modeling.* We introduce the caching problem in HCNs compromising users moving in and out of the SBS coverage areas with the goal of minimizing the probability that a request reaches the macro-cell network. Our model considers realistic features such as the heterogeneity of the SBSs which may have different cache sizes and transmission capacities.
- *Complexity.* We prove the caching problem to be NP-Hard to approximate within any constant factor. The proof is based on a reduction from the Independent Set Decision Problem [34].
- *Centralized small-scale solution.* We formulate the caching problem as a Mixed Integer Programming (MIP) problem, and give a centralized solution using branch and bound techniques.
- *Distributed large-scale solution.* We introduce an optimization framework that relates the probability that a request reaches the macro-cell network to the total weight of a random walk in a Markov chain. Using large deviation inequalities, we derive a distributed caching algorithm that scales well with problem size.
- *Performance evaluation.* We use real traces of mobility patterns and requests for Youtube videos and show that our approach can offload up to 65% more traffic of the macro-cells than existing caching algorithms. We make our evaluation code publicly available online for the benefit of the research community.

The rest of the paper is organized as follows: Section 2 describes the system model and introduces formally the caching problem. In Section 3, we prove the intractability of the problem and present a centralized solution that is applicable for small problem sizes. A connection to the Markov chain model and a distributed solution that scales well with problem size are presented in Section 4. Section 5 presents our evaluation results, whereas we review our contribution compared to related works in Section 6. We conclude our work in Section 7.

2 Model and problem formulation

In this section, we describe the system model, we present a simple example that shows how mobility

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2575837, IEEE Transactions on Mobile Computing

affects the efficiency of the caching policies, and we formally define the optimization problem.

2.1 System model

We study the downlink operation of a heterogeneous cellular network like the one depicted in Figure 1. A set $\mathcal{N}$ of N small-cell base stations (SBSs) are deployed in a macro-cell operating in conjunction with the conventional macro-cell base station (MBS)[1] . Each SBS $n \in \mathcal{N}$ is equipped with a cache of size C_n (bytes). Although SBSs are often deployed in low-density to fill coverage holes of macro-cells, the density of their deployment is expected to increase in next years [12], [13]. Therefore, the coverage areas of the SBSs may overlap one another, and a user may be concurrently covered by multiple SBSs [4], [15], [18]. Mobile users may repeatedly move in and out of the SBS coverage areas, thus associating with different SBSs at different times.

To model user mobility, we introduce a set $\mathcal{L}$ of L highly visited locations in the macro-cell, e.g., busy shopping blocks, hotspots, crowded crossroads, etc. These locations can be extracted using clustering algorithms on the user mobility traces [35]. Each location $l \in \mathcal{L}$ may be covered by multiple SBSs, denoted by $\mathcal{N}_l \subseteq \mathcal{N}$. We then partition time into identical slots and allow users to move to different locations from slot to slot as in [26]. We assume that the operator leverages previous time period statistics to estimate user's location [10], [11]. In this sense, we denote with p_l the probability that a user is in location $l \in \mathcal{L}$. We also denote with $q_{ll'}$ the probability that a user moves from location l to location l', $\forall l, l' \in \mathcal{L}$ within a time slot. Intuitively, the probability $q_{ll'}$ will be higher for locations l and l' that are in close proximity one another.

The average mobile user demand for a set $\mathcal{F}$ of F popular content files and within a certain time period (e.g., a few hours or days) is assumed to be fixed and known in advance, as in [4] and [15]-[22]. For example, the demand can be learned by analyzing previous time statistics of user request patterns to infer on future events [36], [37]. Specifically, for a request generated in location l, we denote with λ_{lf} the probability that f is the requested file. This captures the preferences of the users in location l for content which may vary from location to location. For example, users on a shopping road may be particularly interested in fashion content, while users in proximity to stock market may frequently ask for financial reports. The size of file $f \in \mathcal{F}$ is denoted with $s_f > 0$ (bytes).

We consider *delayed data offloading* [24] and associate each request with a deadline d. That is, each request must be served within a specified time window of d slots by the encountered SBSs, or it will be redirected to the MBS. Clearly, a user visits d locations within the deadline. We refer to the sequence of visited locations as the *walk* of the user, i.e., $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$, where $w_i \in \mathcal{L}$ is the location visited at slot $i \in \{1, 2, \ldots, d\}$. Since a user may visit multiple times the same location, the walk $\boldsymbol{w}$ is a *multiset*, i.e., it possibly includes duplicate elements. Let us denote with $\mathcal{W}$ the set of all possible walks. Then, the probability that a user takes a walk $\boldsymbol{w} \in \mathcal{W}$ is given by:

$$r_{\boldsymbol{w}} = p_{w_1} \prod_{i=1}^{d-1} q_{w_i w_{i+1}} \tag{1}$$

1 . Our model can be directly extended for multiple macro-cells and MBSs.

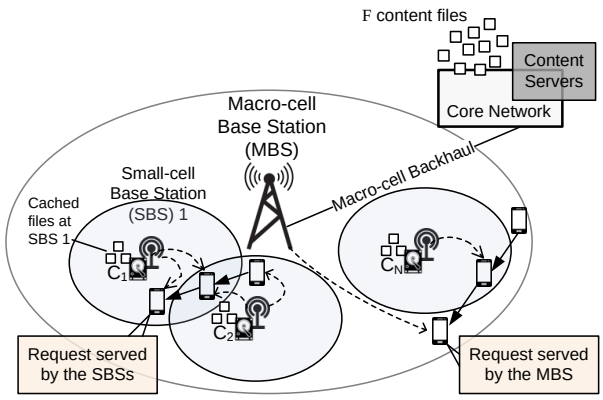


Fig. 1. Graphical illustration of the proposed model. Circles represent the coverage areas of the MBS and the SBSs. Solid and dashed arrows specify user trajectories and base station associations respectively.

We consider a massive content delivery scenario, e.g., in populated areas or during peak traffic hours. Hence, the bottleneck in content delivery is not the receiver antenna, but the limited transmission capacity of the SBSs. In this sense, we denote with $B_n \geq 0$ the average amount of data that SBS n can transmit to a user within a time slot. The different B_n values across the SBSs reflect the heterogeneity in terms of the deployed bandwidth and the average workload.

To better utilize the SBS cache space, we adopt a Maximum Distance Separable (MDS) code [27], [28] and store encoded versions of the files instead of the raw data packets. Particularly, a set of encoded segments for each file is generated. Then, we need to consider how many segments of a file to store at each cache. Successful file recovery occurs when the total amount of encoded data is at least equal to the size of the original (uncoded) file. Previous works [29], [30] revealed the benefits of MDS-based caching over conventional schemes that store uncoded file segments at the caches [31], [32], [33].

Our goal is to determine the content placement at each SBS to fully utilize the network resources, such as the limited cache sizes of the SBSs, and the limited contact duration with the users. Traditional caching schemes neglect user mobility and contact duration limits and store complete copies of the files at the caches [4], [15]-[22]. In the following, we show the inefficiency of such schemes using a simple example.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2575837, IEEE Transactions on Mobile Computing

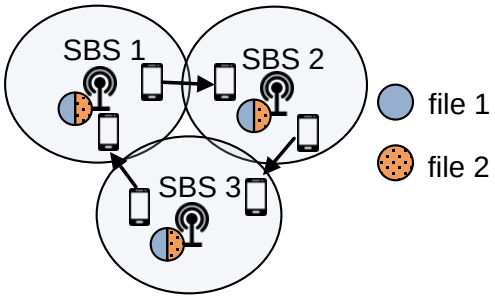


Fig. 2. An example with $N = 3$ SBSs, $F = 2$ unit-sized files and unit-sized caches. Mobile users contact uniformly at random two SBSs within $d = 2$ slots deadline. During each contact half unit of data can be transmitted.

2.2 Motivating example

Let us consider the scenario in Figure 2 with $N = 3$ SBSs each one equipped with a unit-sized cache. Within a deadline of $d = 2$ time slots, each mobile user contacts uniformly at random two SBSs as he moves. During each contact at most half unit of data can be transmitted due to the time slot duration limits. There exist also $F = 2$ equally-popular files each one of size one.

In a mobility-agnostic system, each user is assumed to be stationary and hence he can be served only by the local SBS. It is well known that placing the most popular files with respect to the local demand in each cache is optimal (in terms of MBS load) in this setting [4]. Since the two files are equally popular in our example, there is an indifference in caching them by the operator. Hence, the optimal caching policy would store a complete copy of either file 1 or file 2 at each SBS. If all SBSs store the same file, say file 1, then all the users can successfully download this file by the two encountered SBSs; half unit of data by each one of them. However, all the requests for file 2 will be redirected to MBS. In the other case that an SBS, say SBS 1, stores a different file (file 2), then only the users requesting file 1 contacting the two last SBSs would successfully download the requested file within the deadline. The rest users will be redirected to the MBS, since they will download at most half of the requested file by the encountered SBSs.

However, if we take into consideration the fact that the users move and access pairs of SBSs, then the optimal caching policy changes; it stores half unit of MDS-encoded data of each file at each SBS. In this case, no matter which file a user requests, he can download half of it by each encountered SBS. The downloaded encoded parts can be combined to recover the requested (uncoded) file. Hence, none of the requests will be redirected to MBS. This example, reveals the *inefficiency of caching schemes that neglect user mobility and contact duration limits.*

2.3 Problem formulation

We introduce the optimization variable $x_{nf} \in [0,1]$ to indicate the portion of encoded data of file f that is cached at SBS n. These variables constitute the caching policy of the operator:

$$\boldsymbol{x} = (x_{nf} \in [0,1] \ : \ \forall n \in \mathcal{N}, \ f \in \mathcal{F}) \tag{2}$$

A user may encounter the same SBS multiple times during his walk. We denote with $\mathcal{N}_{\boldsymbol{w}} \subseteq \mathcal{N}$ the set of SBSs encountered at least once during the walk $\boldsymbol{w}$, and $\alpha_{\boldsymbol{w}n}$ the exact number of appearances of SBS n in $\boldsymbol{w}$. Clearly, it is wasteful for a user to download the same data already downloaded by an SBS at previous contacts. Specifically, during the 1^{st} contact with SBS n, the *useful* portion of file f that can be downloaded is given by:

$$y_{nf}^1 = \min\{x_{nf}, \ \frac{B_n}{s_f}\} \tag{3}$$

i.e., it is upper bounded by the portion of cached data and the SBS transmission capacity. During the k^{th} contact with SBS n, where $k \in \{2,3,\ldots,\alpha_{\boldsymbol{w}n}\}$, the *useful* portion of file f is given by:

$$y_{nf}^k = \min\{x_{nf} - \sum_{t=1}^{k-1} y_{nf}^t, \ \frac{B_n}{s_f}\} \tag{4}$$

where we have subtracted from x_{nf} the portion of file f downloaded at the $k-1$ previous contacts.

A user request will reach the MBS if the total portion of the requested file that is downloaded by the SBSs is less than 1. To express the probability of this event, we note that, for a user taking walk $\boldsymbol{w}$ and requesting file f, the total portion of the file f downloaded by the encountered SBSs is equal to $\sum_{n\in\mathcal{N}_{\boldsymbol{w}}} \sum_{k=1}^{\alpha_{\boldsymbol{w}n}} y_{nf}^k$. Since the latter indirectly depends on the caching policy $\boldsymbol{x}$ (cf. equations (3), (4)), the probability that a user request reaches the MBS can be expressed as follows:

$$J(\boldsymbol{x}) = \sum_{\boldsymbol{w}\in\mathcal{W}} r_{\boldsymbol{w}} \sum_{f\in\mathcal{F}} \lambda_{w_1 f} \mathbf{1}_{\{\sum_{n\in\mathcal{N}_{\boldsymbol{w}}} \sum_{k=1}^{\alpha_{\boldsymbol{w}n}} y_{nf}^k < 1\}} \tag{5}$$

where $\mathbf{1}_{\{.\}}$ is the indicator function, i.e., $\mathbf{1}_{\{c\}} = 1$ iff condition c is true; otherwise $\mathbf{1}_{\{c\}} = 0$. Table 1 summarizes the key notations used throughout the paper.

The problem of deriving the caching policy that minimizes the probability that a request reaches the MBS can be expressed as follows:

$$\min_{\boldsymbol{x}} \ J(\boldsymbol{x}) \tag{6}$$

$$s.t. \ \sum_{f\in\mathcal{F}} s_f x_{nf} \leq C_n, \forall n \in \mathcal{N} \tag{7}$$

$$x_{nf} \in [0,1], \forall n \in \mathcal{N}, f \in \mathcal{F} \tag{8}$$

where constraints in (7) indicate that the total amount of data placed in a cache should not exceed its capacity. Inequalities in (8) indicate the non-negativeness of the optimization variables and that it would be wasteful to place to a cache more than one unit of a file. The above problem is difficult to solve due to it's non-convex nature and the high number of different walks that a user can take. Namely, there exist L^d such walks. In the next two sections, we prove the intractability of the problem and provide efficient solutions.

TABLE 1
Key Notations

Symbol	Physical Meaning
$\mathcal{N}$	Set of N SBSs
$\mathcal{L}$	Set of L locations in the macro-cell
$\mathcal{F}$	Set of F files
s_f	Size of file f
C_n	Cache capacity of SBS n
B_n	Maximum amount of data delivered by SBS n in a slot
$\mathcal{N}_l$	Subset of SBSs covering location l
p_l	Probability that a user is in location l
$q_{ll'}$	Probability that a user moves from location l to l'
λ_{lf}	Probability that a request in location l is for file f
d	Deadline for serving requests (slots)
$\boldsymbol{w}$	Multiset of d locations representing a user walk
$\mathcal{N}_{\boldsymbol{w}}$	Subset of SBSs encountered at least once in walk $\boldsymbol{w}$
$\alpha_{\boldsymbol{w}n}$	Number of appearances of SBS n in walk $\boldsymbol{w}$
$r_{\boldsymbol{w}}$	Probability of walk $\boldsymbol{w}$
x_{nf}	Portion of file f cached at SBS n
y_{nf}^k	Useful portion of file f at the k^{th} contact with SBS n
$J(\boldsymbol{x})$	Probability that a user request reaches MBS
$z_{\boldsymbol{w}f}$	Indicator of service by the MBS for file f and walk $\boldsymbol{w}$

3 Complexity and Centralized Small-Scale Solution

In this section, we formally prove the intractability of the caching problem and show how to formulate it as a *Mixed Integer Programming* (MIP) problem. This is important since there exist many commercial packages, such as CPLEX [25], that allow for efficient solution to such problems.

3.1 Complexity

As Lemma 1 states, the caching problem is NP-Hard to approximate within any constant factor.

Lemma 1. *It is NP-Hard to approximate the problem described in (6)-(8) within any constant factor.*

Proof. To prove lemma 1, we consider the corresponding decision problem, Caching Decision Problem (CDP):

Definition 1. CDP: *Given the values of the terms d, $r_{\boldsymbol{w}}$, $\mathcal{N}_{\boldsymbol{w}}$, $\alpha_{\boldsymbol{w}n}$, λ_{lf}, s_f, B_n and C_n defined in Table 1 and a number $Q > 0$, we ask: does there exist a caching policy $\boldsymbol{x}$, such that the value of the objective function in (6) is less or equal to Q and constraints (7)-(8) are satisfied?*

We will prove the NP-Hardness of CDP by reduction from the independent set decision problem (ISDP) [34].

Definition 2. ISDP: *Consider an undirected graph with a set $\mathcal{V}$ of V vertices and a set $\mathcal{E}$ of E edges. We ask: do there exist k vertices that are non-adjacent, i.e., for every two vertices there is no edge connecting the two?*

We will show that every instance of the ISDP problem can be expressed as a specific instance of the CDP problem. We construct this instance as follows:

We create V walks, one walk for each vertex $v \in \mathcal{V}$. We denote with $w(v)$ the walk corresponding to vertex v. For each walk, we also create a unit-sized file and a user that requests this file following this walk. Each walk includes a sequence of d SBSs without duplicates, where each SBS corresponds to a separate location. Each SBS can deliver arbitrarily large amount of data per slot, i.e., $B_n = +\infty$, $\forall n$. We also restrict the aggregate cache space of the SBSs in a walk $w(v)$ to be equal to 1, i.e., $\sum_{n \in \mathcal{N}_{w(v)}} C_n = 1$, $\forall v \in \mathcal{V}$. The important point is that we force every two walks $w(v_1)$ and $w(v_2)$ for which the vertices v_1 and v_2 are adjacent in the ISDP instance to share a common SBS.

If the MBS serves all the requests, then the objective function in (6) has a value equal to 1 (the worst case scenario). For each user that the operator manages to serve completely through local caching at the SBSs, the operator reduces this value by $1/V$. Hence, there will be a caching policy with value equal to $1 - k/V$ if there are k users that are served by the SBSs during their walks. Notice that the aggregate amount of data that the SBSs in a walk can cache is at most 1 and each user requests a separate file. Hence, the caching decisions should be disjoint with respect to the k walks. Since every walk corresponds to a separate vertex, this occurs when there are k vertices in the ISDP instance that are non-adjacent. Hence, the CDP instance is equivalent to the ISDP instance.

ISDP is NP-Hard to approximate within any constant factor [34]. Hence, the above reduction indicates the inapproximability of CDP as well and completes the proof of Lemma 1. □

3.2 MIP formulation

To obtain the MIP formulation of the caching problem, we express the y_{nf}^k terms, introduced in equations (3)-(4), as optimization variables and denote with $\boldsymbol{y}$ the respective vector:

$$\boldsymbol{y} = (y_{nf}^k \in [0,1] : n \in \mathcal{N}, f \in \mathcal{F}, k \in \{1, 2, \ldots, d\}) \quad (9)$$

We also introduce the *integer* optimization variables $\boldsymbol{z}$:

$$\boldsymbol{z} = (z_{\boldsymbol{w}f} \in \{0,1\} : \boldsymbol{w} \in \mathcal{W}, f \in \mathcal{F}) \quad (10)$$

Here, $z_{\boldsymbol{w}f}$ indicates whether file f will be delivered to a user taking a walk $\boldsymbol{w}$ by the MBS ($z_{\boldsymbol{w}f} = 1$) or not ($z_{\boldsymbol{w}f} = 0$).

Then, the MIP problem can be expressed as follows:

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}} \sum_{\boldsymbol{w} \in \mathcal{W}} r_{\boldsymbol{w}} \sum_{f \in \mathcal{F}} \lambda_{\boldsymbol{w}_1 f} z_{\boldsymbol{w}f} \quad (11)$$

$$\text{s.t. } (7) - (8)$$

$$y_{nf}^k \in [0, \frac{B_n}{s_f}], \ \forall n \in \mathcal{N}, f \in \mathcal{F}, k = 1, ..., d \quad (12)$$

$$\sum_{k=1}^{d} y_{nf}^k \leq x_{nf}, \ \forall n \in \mathcal{N}, f \in \mathcal{F} \quad (13)$$

$$z_{\boldsymbol{w}f} \geq 1 - \sum_{n \in \mathcal{N}_{\boldsymbol{w}}} \sum_{k=1}^{\alpha_{\boldsymbol{w}n}} y_{nf}^k, \ \forall \boldsymbol{w} \in \mathcal{W}, f \in \mathcal{F} \quad (14)$$

$$z_{\boldsymbol{w}f} \in \{0,1\}, \forall \boldsymbol{w} \in \mathcal{W}, f \in \mathcal{F} \quad (15)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2575837, IEEE Transactions on Mobile Computing

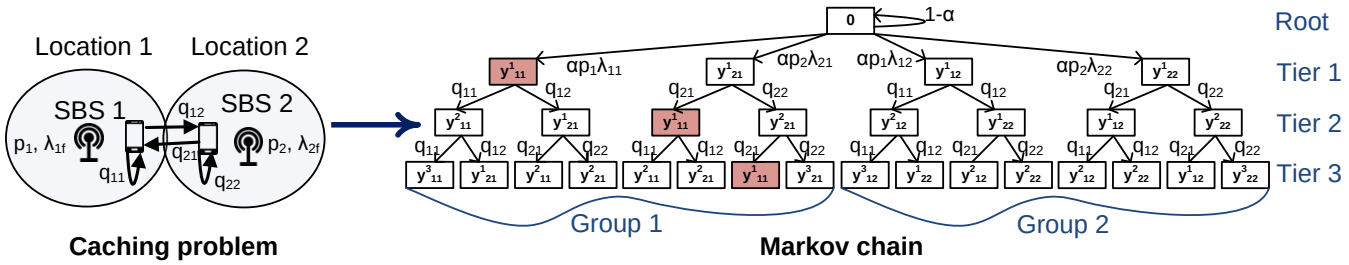


Fig. 3. An example of the Markov chain with $N = 2$ SBSs, $L = 2$ locations, $F = 2$ files and $d = 3$ slots. In the chain, rectangles denote states and state labels indicate the respective weights. Link labels specify the transition probabilities. To ease presentation, the links uniting tier-d states to the root are omitted.

Inequalities (12)-(13) stem of decomposition of inequalities (3)-(4). Inequality (14) restricts that $z_{\boldsymbol{w}f}$ will be 1 if $\sum_{n\in\mathcal{N}_{\boldsymbol{w}}}\sum_{k=1}^{\alpha_{\boldsymbol{w}n}} y_{nf}^{k} < 1$.

In practice, MIP problems can be solved only for small-scale instances, i.e., involving a few number of SBSs, locations and files. This is because the applied Branch & Bound methods perform implicit enumeration of the solution space, partitioning it into a search tree of exponential size. Also, a central entity is required to compute and communicate the solution to the SBSs. In the next section, we show how to derive an efficient distributed solution for arbitrarily large problem instances.

4 Distributed Large-Scale Solution

In this section, we establish a distributed solution for large-scale caching systems. We first show how the caching problem relates to the Markov chain model, then we establish an upper bound on the objective function in (6) using large deviation inequalities, and finally we propose a distributed algorithm that minimizes this upper bound.

4.1 Relation to the Markov chain model

We introduce a Markov chain with state space $\mathcal{S}$ and matrix M of transition probabilities. A sequence of states $(S_1, S_2, \ldots, S_t)$ indicates a t-step random walk on the chain starting from an initial distribution Φ on $\mathcal{S}$. States can be assigned weights based on a function $\Omega : \mathcal{S} \to [0, 1]$. In this case, the total weight of a walk $(S_1, S_2, \ldots, S_t)$ is equal to $\sum_{i=1}^{t} \Omega(S_i)$. Given an instance of the caching problem, we construct the corresponding Markov chain as follows.

State space $\mathcal{S}$. The state space consists of the following: (i) a *root* state indexed by 0, and (ii) a group of $\sum_{t=1}^{d} L^t$ *inner* states for each file $f \in \mathcal{F}$. The states of each group are partitioned into d tiers, where tier 1 contains the first L states, tier 2 the next L^2 states and so on. We denote with $\mathcal{F}_u \in \{1, 2, \ldots, F\}$ and $\mathcal{T}_u \in \{1, 2, \ldots, d\}$ the group and the tier of a state u respectively. Here, a state u belonging to tier $t \in \{1, 2, \ldots, d-1\}$ is the unique *parent* of L states in tier $t + 1$. We call the latter states as the *children* of state u and denote with $\mathcal{C}_u \subseteq \mathcal{S}$ the respective set. We also introduce the notation $\mathcal{L}_u \in \{1, 2, \ldots, L\}$ for a state u, where $\mathcal{L}_u = l$ if u is the l^{th} child of another inner state. Similarly, the root state is the unique parent of all the states in tier 1, L for each group. For a tier-1 state u that is the l^{th} child of the root in a particular group, we set $\mathcal{L}_u = l$.

Transition matrix M. The probability of transiting from state $u \in \mathcal{S}$ to $v \in \mathcal{S}$ is given by:

$$M_{uv} = \begin{cases} 1-\alpha, & if\ u = v = 0 \\ \alpha \cdot p_{\mathcal{L}_u} \cdot \lambda_{\mathcal{L}_u\mathcal{F}_u}, & if\ u = 0, \mathcal{T}_v = 1 \\ q_{\mathcal{L}_u\mathcal{L}_v}, & \text{if } 0 < \mathcal{T}_u < d, v \in \mathcal{C}_u \\ 1, & \text{if } \mathcal{T}_u = d, v = 0 \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

where $a \in (0, 1)$ is any constant. Therefore, during a walk, when at the root state, the chain can either stay in it or move to a tier-1 state at the next step. Then, the chain moves to a tier-2 state and so on. Having reached a tier-d state, the chain moves back to the root. Figure 3 illustrates an example for $N = 2$ SBSs, $L = 2$ locations, $F = 2$ files and $d = 3$ slots deadline. Here, each location corresponds to a single SBS coverage area.

Initial Probability Φ. The probability that a random walk starts from a state $u \in \mathcal{S}$ is given by:

$$\Phi(u) = \begin{cases} p_{\mathcal{L}_u} \cdot \lambda_{\mathcal{L}_u\mathcal{F}_u}, & \text{if } \mathcal{T}_u = 1 \\ 0, & \text{otherwise} \end{cases} \quad (17)$$

Therefore, every walk starts from a tier-1 state.

We can show that a user's walk in the macro-cell in the caching problem can be expressed as random walk on the above chain. Specifically, each inner state u represents a specific location $\mathcal{L}_u \in \mathcal{L}$ visited by a user requesting file $\mathcal{F}_u \in \mathcal{F}$ at time slot $\mathcal{T}_u \in \{1, 2, \ldots, d\}$. By construction, a random walk starting from a tier-1 state traverses $\mathcal{T}_u - 1$ states before reaching u. Equivalently, the user visits $\mathcal{T}_u - 1$ locations before reaching location $\mathcal{L}_u$. Denoting with $w(u)$ such a walk, the notation $\alpha_{w(u)n}$ specifies the number of appearances of SBS n in this walk. Based on Eqn. (3) and Eqn. (4), the *useful* portion of file $\mathcal{F}_u$ that the user downloads by an SBS $n \in \mathcal{L}_u$ at slot $\mathcal{T}_u$ is equal to $y_{n\mathcal{F}_u}^{\alpha_{w(u)n}}$. Then, the important point for this relation to hold is to define the state weights as follows:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2575837, IEEE Transactions on Mobile Computing

Weight function Ω. The weight of a state $u \in \mathcal{S}$ is given by:

$$\Omega(u) = \begin{cases} 0, & \text{if } u = 0 \\ \sum_{n \in \mathcal{N}_{\mathcal{L}_u}} y_{n\mathcal{F}_u}^{\alpha_{w(u)n}}, & \text{otherwise} \end{cases} \quad (18)$$

Hence, the total weight of a $d-$step walk on the chain, which we model with a random variable Y, specifies the total portion of the requested file that a user manages to download by the encountered SBSs within the delay deadline. If $Y \geq 1$, the amount of downloaded data suffices to recover the requested file; otherwise the request is redirected to the MBS. The objective function in (6) can be written as the probability that Y is lower than 1, i.e., $\Pr[Y < 1]$. In the following, we show how the presented framework can be used to optimize the caching policy.

4.2 Upper bound on the objective function

We start with the following large deviation inequality.

Lemma 2. *(Hoeffding's Inequality for Markov Chains [38]). Consider an ergodic Markov chain and a t-step random walk starting from an initial distribution ϕ with total weight Y. For any $\delta \in [0,1]$, there exists some constant c such that:*

$$\Pr[Y \leq (1-\delta)\mu t] \leq c||\phi||_\pi \exp\left\{ -\frac{\delta^2 \mu t}{72T} \right\} \quad (19)$$

where π is the stationary distribution, μ is the expected weight of the walk with respect to π, T is the mixing time, and $||\phi||_\pi$ indicates the $\pi-$norm of the vector ϕ.

Clearly, the Markov chain that we constructed in the previous subsection is *ergodic*, i.e., it is possible to go from every state to every state within a finite number of steps. Hence, we can use Lemma 2 to upper bound the objective function in (6), i.e., $\Pr[Y < 1]$. Specifically, for $t = d$, $\delta = 1 - 1/(\mu d)$ and $\mu d \geq 1$, and using the inequality $\Pr[Y < 1] \leq \Pr[Y \leq 1]$, we can show that:

$$\Pr[Y < 1] \leq c||\Phi||_\pi \exp\left\{ -\frac{\mu d + \frac{1}{\mu d} - 2}{72T} \right\} \quad (20)$$

Besides of the constant c, the value of which is given in [38], the initial probability distribution Φ defined in the previous subsection, and the delay deadline d, the upper bound depends on: (i) the stationary distribution π, (ii) the expected weight of a walk μ and (iii) the mixing time T. In the rest of this subsection, we formally define these values.

(i) Stationary distribution π. The stationary distribution π is a probability distribution vector on the states that is unchanged by the operation of transition matrix M on it, i.e.,

$$\pi = \pi M \quad (21)$$

Due to the special structure of this chain, we can compute the stationary probability for a state $u \neq 0$ as follows:

$$\pi(u) = \pi(0) \cdot M_{0,S_1} \cdot M_{S_1,S_2} \cdot \cdots \cdot M_{S_l,u} \quad (22)$$

where $(S_1, S_2, \ldots, S_l)$ denotes the intermediate states on the walk starting from the root until state u. Besides, since π is a probability distribution, it holds that:

$$\sum_{u \in \mathcal{S}} \pi(u) = 1 \quad (23)$$

By combining (22) and (23), we can show that:

$$\pi(0) = \frac{1}{1 + \alpha d} \quad (24)$$

(ii) Expected weight of a $d-$step walk μ. Let us first denote with $\Pr[y_{nf}^k]$ the probability with respect to π of reaching *any* state u with $n \in \mathcal{L}_u$, $f = \mathcal{F}_u$ and $\alpha_{w(u)n} = k$. For example, in Figure 3 there are three states with weight y_{11}^1 (the states corresponding to marked rectangles) and hence it should be:

$$\begin{aligned} \Pr[y_{11}^1] = {} & \pi(0)\alpha p_1 \lambda_{11} \\ & + \pi(0)\alpha p_2 \lambda_{21} q_{21} \\ & + \pi(0)\alpha p_2 \lambda_{21} q_{22} q_{21} \end{aligned} \quad (25)$$

Then, by definition, the expected weight of a $d-$step random walk is equal to:

$$\mu(\boldsymbol{y}) = \sum_{n=1}^{N} \sum_{f=1}^{F} \sum_{k=1}^{d} \Pr[y_{nf}^k] y_{nf}^k \quad (26)$$

where we have explicitly expressed μ as a function of $\boldsymbol{y}$ variables to capture this dependency.

(iii) Mixing time T. Following [38], we define $T = \min\{t : \max_q ||qM^t - \pi||_{TV} \leq \frac{1}{8}\}$. Here, q is an arbitrary initial distribution over $\mathcal{S}$. For two distributions q and q', it is: $||q - q'||_{TV} = \max_{A \subseteq \mathcal{S}} |\sum_{i \in A} q_i - \sum_{i \in A} q_i'|$.

4.3 Distributed algorithm

Since it is NP-Hard to directly minimize the objective function in (6), we take an alternate approach and minimize its upper bound in (20). From this approach, we obtain an approximate solution, together with a guaranteed upper bound on the achieved probability of requests routed to MBS.

Lemma 3 shows that minimizing this bound is equivalent to maximizing the expected weight $\mu(\boldsymbol{y})$.

Lemma 3. *Let* $g(\boldsymbol{y}) = c||\phi||_\pi \exp\left\{ -\frac{\mu(\boldsymbol{y})d + \frac{1}{\mu(\boldsymbol{y})d} - 2}{72T} \right\}$. *Then,* $\operatorname{argmin}_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{A}}\{g(\boldsymbol{y})\} = \operatorname{argmax}_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{A}}\{\mu(\boldsymbol{y})\}$, *where* $\mathcal{A} = \{\boldsymbol{x}, \boldsymbol{y} : x_{nf} \in [0,1], y_{nf}^k \in [0,1]\ \forall n \in \mathcal{N},\ f \in \mathcal{F},\ k \in \{1,2,\ldots,d\}$ *and constraints (7)-(8) and (12)-(13) are satisfied*$\}$.

Proof. Let $(\boldsymbol{x}^*, \boldsymbol{y}^*) = \operatorname{argmin}_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{A}}\{g(\boldsymbol{y})\}$, i.e., $g(\boldsymbol{y}^*) \leq g(\boldsymbol{y})\ \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A}$. Dividing by $c||\phi||_\pi$ and then taking the logarithm on both sides preserves the inequality as $c > 0$, $||\phi||_\pi > 0$ and $\log(.)$ is increasing function. Hence, we obtain:

$$-\frac{\mu(\boldsymbol{y}^*)d + \frac{1}{\mu(\boldsymbol{y}^*)d} - 2}{72T} \leq -\frac{\mu(\boldsymbol{y})d + \frac{1}{\mu(\boldsymbol{y})d} - 2}{72T} \quad (27)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2575837, IEEE Transactions on Mobile Computing

Dividing by $-72T \leq 0$ and then adding 2 on both sides yields:

$$\mu(\boldsymbol{y}^*)d + \frac{1}{\mu(\boldsymbol{y}^*)d} \geq \mu(\boldsymbol{y})d + \frac{1}{\mu(\boldsymbol{y})d} \tag{28}$$

However, it holds that: $\mu(\boldsymbol{y})d \geq 1$, resulting that: $\mu(\boldsymbol{y}^*)d \geq \mu(\boldsymbol{y})d \ \forall(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{A}$, which completes the proof. □

Based on Lemma 3, the caching problem becomes:

$$\max_{\boldsymbol{x},\boldsymbol{y}} \ \mu(\boldsymbol{y}) \tag{29}$$
$$s.t. \ (7), (8), (12), (13)$$

The above problem is more tractable than the original caching problem. By the structure of $\mu(\boldsymbol{y})$, that is described in Eqn. (26), and the above constraints we can show that the caching decisions at an SBS do not affect the rest. Hence, we can *decompose* this problem to N independent subproblems, one for each SBS, and solve them in a *distributed* manner. The subproblem for an SBS $n \in \mathcal{N}$, which we refer to as $\mathcal{P}_n$, can be expressed as:

$$\mathcal{P}_n: \ \max_{\boldsymbol{x}_n,\boldsymbol{y}_n} \ \sum_{f=1}^{F}\sum_{k=1}^{d} \Pr[y_{nf}^k] y_{nf}^k \tag{30}$$
$$s.t. \ \sum_{f\in\mathcal{F}} s_f x_{nf} \leq C_n \tag{31}$$
$$x_{nf} \in [0,1], \forall f \in \mathcal{F} \tag{32}$$
$$y_{nf}^k \in [0, \frac{B_n}{s_f}], \ \forall f \in \mathcal{F}, k = 1, ..., d \tag{33}$$
$$\sum_{k=1}^{d} y_{nf}^k \leq x_{nf}, \ \forall f \in \mathcal{F} \tag{34}$$

where $\boldsymbol{x}_n$ and $\boldsymbol{y}_n$ denote the variables in $\boldsymbol{x}$ and $\boldsymbol{y}$ for SBS n. The objective function and the constraints of this problem are linear with respect to the optimization variables. Hence, it can be efficiently solved using standard *linear optimization techniques* [39]. Going one step further, we show that $\mathcal{P}_n$ falls into a class of knapsack problems with *known solution structure*, alleviating the need for applying linear optimization techniques. Specifically, we will prove the following lemma.

Lemma 4. *The optimal solution of problem $\mathcal{P}_n$ can be computed in $O(F \cdot d \cdot \log(F \cdot d))$ time.*

Proof. Fractional knapsack problem asks for placing fractions of items of different values and weights in a knapsack of limited capacity in a way that maximizes the aggregate value of items placed in it [40]. The problem $\mathcal{P}_n$ can be translated to a *restricted* version of the fractional knapsack problem in which there exist $F \cdot d$ items, one item for each variable y_{nf}^k, $f \in \mathcal{F}$, $k = 1, 2, \ldots, d$ and a knapsack of capacity C_n. The value of the item corresponding to y_{nf}^k is $\Pr[y_{nf}^k]$ and it's weight is s_f. The item placement must also satisfy the following two restrictions: **Restriction 1**: At most B_n/s_f fraction of the item corresponding to variable y_{nf}^k can be placed in the knapsack, $\forall f, k$. **Restriction 2**: The total amount of items corresponding to variables y_{nf}^k, $k = 1, 2, \ldots, d$ placed in the knapsack must be less or equal to 1, $\forall f$.

Algorithm 1: Mobility-aware caching algorithm

Input : An instance of the $\mathcal{P}_n$ problem.
Output: The optimal solution $\boldsymbol{x}_n, \boldsymbol{y}_n$.

1 $value_{nf}^k \leftarrow \Pr[y_{nf}^k]$, $\forall f \in \mathcal{F}, k \in \{1, \ldots, d\}$;
2 $weight_{nf}^k \leftarrow s_f$, $\forall f \in \mathcal{F}, k \in \{1, \ldots, d\}$;
3 $y_{nf}^k \leftarrow 0, \ \forall f \in \mathcal{F}, k \in \{1, \ldots, d\}$;
4 $\mathcal{D} \leftarrow \mathcal{F} \times \{1, \ldots, d\}$;
5 **for** $i = 1, 2, \ldots, F \cdot d$ **do**
6 $\quad (f^*, k^*) \leftarrow \text{argmax}_{(f,k)\in\mathcal{D}} \frac{value_{nf}^k}{weight_{nf}^k}$;
7 $\quad y_{nf^*}^{k^*} \leftarrow \min\left\{\frac{B_n}{s_{f^*}}, \ 1 - \sum_{k=1}^{d} y_{nf^*}^k, \ C_n - \sum_{f\in\mathcal{F}}\sum_{k=1}^{d} y_{nf}^k\right\}$;
8 $\quad \mathcal{D} \leftarrow \mathcal{D} \setminus (f^*, k^*)$;
9 $\quad$ **if** $\sum_{f\in\mathcal{F}}\sum_{k=1}^{d} y_{nf}^k = C_n$ **then**
10 $\quad\quad$ break;
$\quad$ **end**
end
11 $x_{nf} = \sum_{k=1}^{d} y_{nf}^k, \ \forall f \in \mathcal{F}$;

The optimal solution of this knapsack-type problem can be attained by a simple scheme that iteratively places a fraction of the item with the highest ratio of value/weight in the knapsack until the knapsack becomes full [40]. At each iteration, the scheme ensures that the item placement satisfies the above two restrictions. The knapsack solution can be mapped to a solution to the problem $\mathcal{P}_n$ such that every y_{nf}^k variable takes as value the fraction of the associated item placed in the knapsack and x_{nf} takes as value the sum: $\sum_{k=1}^{d} y_{nf}^k$. This procedure is summarized in Algorithm 1.

Here, $value_{nf}^k$ and $weight_{nf}^k$ denote the value and weight of the item corresponding to variable y_{nf}^k respectively (lines 1-2). Operator $\times$ denotes the cartesian product of two sets. Collection $\mathcal{D}$ includes all the (f, k) pairs for which item y_{nf}^k has not been picked yet (line 4). At every iteration, Algorithm 1 picks the pair $(f^*, k^*) \in \mathcal{D}$ with the largest ratio value/weight (line 6). Then, a fraction of item $y_{nf^*}^{k^*}$ will be placed in the knapsack satisfying the two restrictions (line 7). The algorithm terminates when all the items are picked or the knapsack becomes full (line 9).

In the worst case, all the $F \cdot d$ items will be picked. Since the items are picked in decreasing order of their ratios of value/weight, sorting all these ratios is required. Hence, the complexity of Algorithm 1 will be $O(F \cdot d \cdot \log(F \cdot d))$ [41], which completes the proof. □

4.4 Implementation considerations

We need to emphasize that the overhead of cooperation among SBSs and MBS for serving the user requests may affect the efficiency of Algorithm 1. For example, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2575837, IEEE Transactions on Mobile Computing

operator may use techniques like network initiated offloading [42] to dynamically decide which SBS will serve each request taking into account the cached content. This may cause additional latency for signaling among SBSs and MBS [43] which can be to the detriment of the mobile users. An operator can estimate the overhead latency, e.g., by processing previous time period statistics. This information can be used to estimate the amount of data delivered in a slot to a user. The latter is captured in our model by the parameter B_n for each SBS n that is passed as input to Algorithm 1.

A second aspect that we need to consider is that several Video-on-Demand sites (YouTube, Netflix, Hulu, etc) make use of DASH-based [44] (or similar) adaptive streaming approaches in the late years. Optimizing the quality of streaming experience creates a far more difficult challenge for caching in SBSs since it involves new metrics such as start-up delay, video stalls, frame rate and spatial resolution. This is a different approach from our main objective, since the method we describe centers on offloading the macro-cell networks. The latter is particularly important during periods of peak traffic when scalability issues are raised. Even if an adaptive video streaming protocol is applied, we stress that our method requires only a small delay for streaming to start. This is captured by the delay deadline d. As we show in the next section, our approach achieves significant gains, in terms of reduction in macrocell's load, delaying video viewing by less than 1 minute. The latter is an often acceptable video start-up delay.

5 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithm using real traces of human mobility patterns and requests for Youtube videos. Overall, we find that moving from a conventional caching algorithm to one enhanced with mobility-awareness reduces the load of the macro-cell network, and this heavily depends on cache sizes, delay deadline and density of SBS deployment. In the best scenario, an operator can improve its bottom line by 65% delaying data transfer by 1 minute. In the rest of this section, we discuss these results in detail; we begin by describing the algorithms used in the later evaluations.

5.1 Algorithms

Throughout the evaluation, we compare the performance of three algorithms:

1) *Max-popularity*: Each SBS caches the locally most popular files independently from the others.
2) *Femtocaching [4]*: All users are assumed to be stationary during the evaluation and caching is performed based on the initial distribution of users in the cell. Particularly, the algorithm starts with all the caches being empty. Iteratively, it performs the placement of a file to a cache that minimizes the probability of requests served by MBS, i.e., $\sum_{l\in\mathcal{L}} p_l \sum_{f\in\mathcal{F}} \lambda_{lf} \mathbf{1}_{\{\sum_{n\in\mathcal{N}_l} x_{nf}<1\}}$. The procedure terminates when all the caches become full.
3) *Mobility-aware*: We apply Algorithm 1 to determine the caching policy for each SBS in a distributed manner.

The first two algorithms take caching decisions considering only user demand (i.e., λ_{lf} and p_l). The proposed algorithm considers also information about the users' motility patterns (i.e., $q_{ll'}$ values) and places MDS-encoded file segments at the SBSs instead of entire file copies.

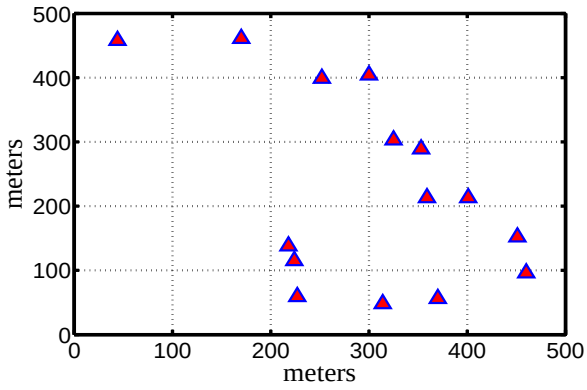


Fig. 4. A 500m$\times$ 500m area with 15 APs (triangles) in [45].

5.2 Mobility model

We evaluate the performance of the described schemes using the measured trace of mobility patterns released by the Wireless Topology Discovery project [45]. This trace contains information from approximately 275 PDA users for an 11 week period between September 22, 2002 and December 8, 2002. Specifically, each active user records every 20 seconds all the WiFi access points (APs) that are detected by its device. Due to the short distance between APs, a user may sense more than one APs at the same time. In total, more than 400 APs are detected. For each AP, its geographical location, described by a pair of (X,Y) values (measured in meters), is also recorded.

We focus on a certain subarea in [45] with dense AP deployment depicted in Figure 4. To facilitate presentation, we shifted the point (X,Y)=(1698270,259950) to the zero coordinates. This area includes $N = 15$ APs in total. In our evaluation, we substitute SBSs for WiFi APs as in [46]. This is a reasonable approximation for picocells, since the radius of the latter usually resembles that of the WiFi APs ($\sim$ 100 meters [14]). Importantly, due to the fact that operators typically keep the datasets of mobility across their base stations confidential, it would be difficult to acquire such information. In contrast, the trace we use is publicly available online. Hence, any caching algorithms that will be developed in the future can be directly compared with the proposed one under the same network settings.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2575837, IEEE Transactions on Mobile Computing

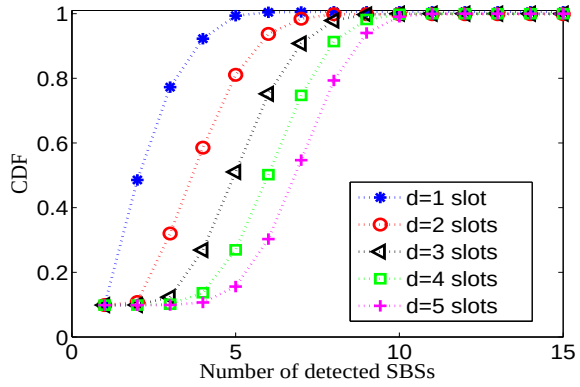


Fig. 5. Cumulative distribution function of the number of SBSs detected during a user walk in [45].

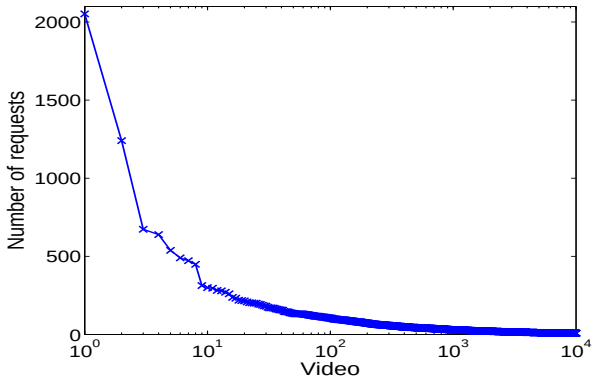


Fig. 6. Number of requests for Youtube videos in [47].

We focus on the busiest day, namely the day of 16 October 2002, and keep the peak time statistics, i.e., between 18pm and 23pm. For each subset of SBSs that concurrently cover a user we create a distinct location, which results into $L = 185$ locations in total. We also set the time slot duration to be 20 seconds. Then, for each location l, we set the p_l probability to be the portion of slots that users visit location l. In order to compute $q_{ll'}$, we divide the number of sequential visits to locations l and l' over the total number of visits to l. If a user becomes inactive by the end of a slot, we assume that he remains in the same location, and hence we increase the value of q_{ll}, where l is the location recorded last.

Intuitively, the benefits of applying a mobility-aware caching scheme instead of a conventional one are higher when users move rapidly in and out of the SBS coverage areas. Therefore, it is crucial to answer how often this occurs. Figure 5 aims to shed light on this question by showing the cumulative distribution function (CDF) of the number of SBSs detected by a user within a deadline of $d \in \{1, 2, 3, 4, 5\}$ slots. Here, for $d = 1$, all the requests are satisfied within a single slot, and hence all the users can download data only by the SBSs detected in this slot. However, for $d > 1$, the users can detect additional SBSs encountered in subsequent slots until the deadline expires. The average number of detected SBSs increases from 2.76 for $d = 1$ up to 6.87 for $d = 5$ slots. This is a drastic increase, bearing in mind that the extra time interval is only a few tens of seconds long. Clearly, all the SBSs that are detected during this extra time interval can be used to deliver content to the user if they have cached parts of the requested file. Hence, *even for a short delay deadline d, e.g., a few minutes, a mobility-aware caching scheme can potentially offload more traffic from the macro-cell network compared to a conventional scheme.*

5.3 Demand model

In order to model user demand, we use a measured trace of YouTube requests from a study performed at Amherst campus, University of Massachusetts, in 2008 [47]. The trace records for each request arising from the (wired) campus network, its exact time and a unique identifier of the requested video. We use the trace data for the two consecutive weeks starting from January 29th, 2008. The number of requests for the $10,000$ most popular videos is presented in Figure 6. In our evaluation, we consider the same video library ($F = 10,000$) and set the λ_{lf} probabilities for each location l based on the popularity of video f in the trace as in [4].

5.4 Evaluation results

Throughout the evaluation, we set all video files to be of size 40 MB, which is reasonable assuming a screen size 640×360, flash encoding and a few minutes ($3 - 4$ min) playback time. Unless otherwise specified, each SBS n is endowed with a cache of size $C_n = C \ \forall n \in \mathcal{N}$ that can store up to 10% of the entire video library size. To set the B_n values, we follow the real bandwidth measurements in [48], which report an average bit-rate between a user and an SBS equal to 8 mbps. Hence, we upper bound the amount of data that an SBS n can deliver to a user in a (20 seconds) slot by $B_n = 20MB$. The performance criterion we use is the probability that a request is served by the macro-cell network denoted with J (Eqn. (5)).

Before we proceed with the evaluation results, let us remark that for the algorithms' implementation we used the C language in the Visual Studio environment. The complete code we wrote is *publicly available* in [49]. Hence, all the presented results can be easily verified for correctness, while we believe this will encourage future experimentation with mobile data caching algorithms.

Impact of cache sizes: We first compare the probability that a request is served by the macro-cell network (J) achieved by the presented algorithms for different cache sizes. In the experiment in Figure 7(a), cache sizes span a wide range of values, starting from 5% to 50% of the entire file library size, reflecting different operator conditions. As expected, increasing the cache sizes reduces J for all algorithms, since more files become available for download from the SBSs. Femtocaching consistently

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2575837, IEEE Transactions on Mobile Computing

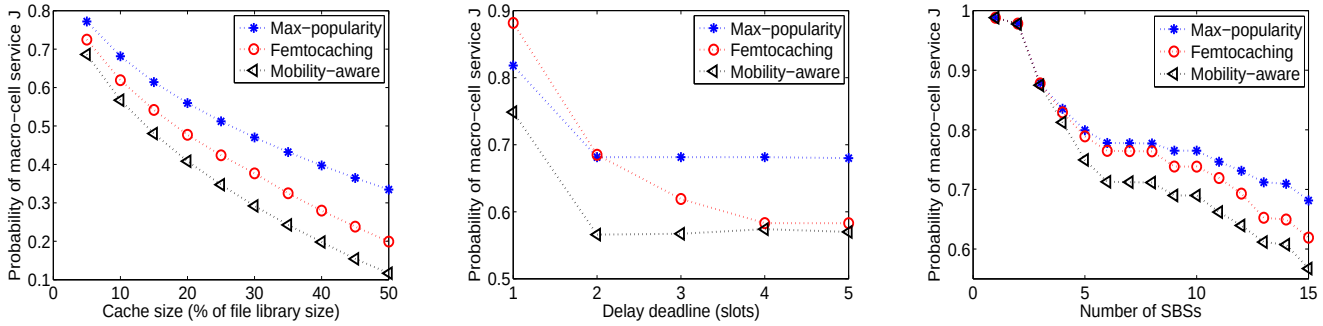


Fig. 7. Probability that a request is served by the macro-cell network for different values of: (a) the cache size per SBS C_n, (b) the delay deadline d and (c) the number of SBSs N.

outperforms Max-popularity algorithm. This can be explained from the fact that the latter simply stores the C_n most popular files at each SBS n. However, in dense SBS deployments, users often access multiple SBSs and each sees a distributed cache that is the union of the respective caches. Clearly, such users would benefit if some of the SBSs stored different (less popular) files, since they could find more files at the accessed SBSs. Femtocaching algorithm identifies such cases by considering the overlapping SBS coverage areas, captured by the $\mathcal{N}_l$ values.

Besides of the superiority of Femtocaching over Max-popularity, Figure 7(a) comments also on the inefficiency of the existing caching policies that are designed for static networks. Namely, in realistic environments where users move rapidly from SBS to SBS, mobility impacts the performance of the caching policy. Mobility-aware is the only algorithm among the three that exploits user mobility, captured by the $q_{ll'}$ values. In our experiment, *Mobility-aware performs markedly better than Max-popularity and Femtocaching, where the gains increase with cache sizes reaching* 65% *and* 41% *respectively*.

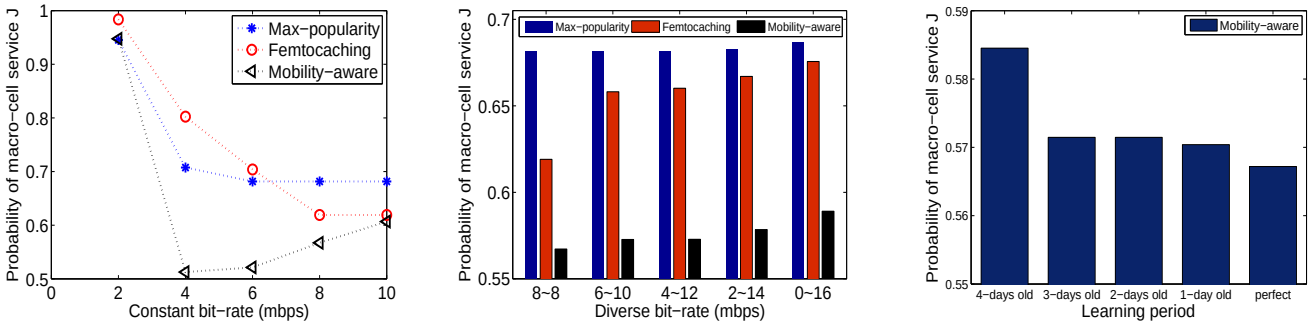
Impact of delay deadline: Figure 7(b) shows how the performance of the presented algorithms depends on the delay deadline d. In this experiment, the deadline varies within $\{1, 2, 3, 4, 5\}$ slots, i.e., $\{20, 40, 60, 80, 100\}$ seconds. This is an often acceptable video start-up delay. We observe that as d increases, probability J decreases for all the algorithms, since users have more contact opportunities with the SBSs. The performance of Max-popularity saturates at $d = 2$, since from this point and above users download all the most popular files that are fully replicated at the SBSs, but none of the rest files. On other hand, Femtocaching stores different files across the SBSs, and hence users can download more files as d increases. Among the three algorithms, Mobility-aware exploits better the contact opportunities, since for a sequence of SBSs that are frequently visited one after the other, coded parts of a file are spread to all the SBSs instead of storing complete file copies at some of them. This increases the number of potential sources from which a user can obtain data and can potentially decrease probability J. Specifically, we find that *Mobility-aware outperforms Max-popularity and Femtocaching for all values of* d*, where the gains can be up to* 16% *and* 15% *respectively*.

Impact of SBS density: We explore how the density of SBSs impacts the results in Figure 7(c). Specifically, we consider the topology depicted in Figure 4, but keep a randomly chosen subset of the SBSs. We observe that as the number of SBSs increases, the probability J decreases for all the algorithms, since users encounter more SBSs within the deadline. Mobility-aware consistently outperforms the rest algorithms, especially for high number of SBSs. For example, the gains over Max-popularity and Femtocaching are very close to zero when there exist five SBSs, but increase to 16% and 8% respectively when all the fifteen SBSs are considered. As a takeaway, *the superiority of Mobility-aware over the alternate algorithms that were examined, is more pronounced for dense SBS deployments*.

Impact of SBS bit-rate: We analyze the impact of the available bit-rate between SBSs and users on the algorithms' performance in Figure 8(a). Specifically, we vary the bit-rate from 2 to 10 mbps. As expected, increasing the bit-rate decreases the probability J achieved by Max-popularity and Femtocaching, since the SBSs can transfer more data during the contacts with users. For low bit-rates, i.e., when the system is in overloaded conditions, storing the most popular files at all SBSs offloads more the macro-cell than the Femtocaching algorithm does. In contrast, as the bit-rate increases, the need for diversing the cached content becomes more apparent and, hence, Femtocaching outperforms Max-popularity. Interestingly, *Mobility-aware outperforms the Max-popularity and Femtocaching algorithms for all bit-rates, a gap being up to* 27% *and* 36% *respectively*.

The numerical results presented so far assume constant bit-rate between SBSs and users. Nevertheless, the bit-rate often varies over time, e.g., due to variations in channel quality, temporal interference, congestion ef-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2575837, IEEE Transactions on Mobile Computing



(a) (b) (c)

Fig. 8. Probability that a request is served by the macro-cell network for (a) constant and (b) diverse bit-rate between SBSs and users. (c) The impact of learning period.

fects, etc. To capture the above dynamics of the wireless medium, we synthesize a variety of scenarios that differ in the way that the bit-rate is set. Specifically, the bit-rate value is randomly drawn from a range. We denote with $a \sim b$ the scenario when the bit-rate is randomly generated within $[a, b]$. Figure 8(b) compares the performance of the three presented algorithms for the scenarios that $a \sim b$ is $8 \sim 8$, $6 \sim 10$, $4 \sim 12$, $2 \sim 14$ and $0 \sim 16$ (mbps). We observe that the probability J increases more-or-less for all the algorithms as the bit-rate becomes more diverse. This can be explained by the fact that, for diverse bit-rates, some users are served with very low bit-rate by the SBSs, while others are served with higher bit-rate than they need. Interestingly enough, we notice that *the performance of Femtocaching is rather sensitive to the diversity of bit-rates, while that of the Max-popularity and Mobility-aware is quite stable*. Mobility-aware is always better than the rest algorithms, with the gains being up to 14.2% and 12.8% when compared to the Max-popularity and Femtocaching algorithm respectively.

Impact of imperfect knowledge of mobility: The numerical results presented so far assume perfect knowledge of user mobility behavior, i.e., the exact probability values p_l and $q_{ll'}$ $\forall l, l' \in \mathcal{L}$ are known. In practice though, these values are predicted by analyzing statistics of a previous time period (learning period). Clearly, the accuracy of such predictions impacts the efficiency of the Mobility-aware caching algorithm. Figure 8(c) aims to shed light on this issue by evaluating the performance of Mobility-aware on October 16^{th}, 2002 for different learning periods. Here, statistics are taken from each one of the previous four days; these are referred to as 1-, 2-, 3- and 4-days old. Interestingly, we see that the performance of Mobility-aware is rather stable within these days (less than 3% loss compared to the case of perfect knowledge), which indicates that the user mobility behavior changes slowly in time. This is very important as it shows that, *for an operator periodically tracking user movements in the small-cell network, substantial performance benefits can be realized from applying our algorithm*.

Finally, we emphasize that there may be cases when the user mobility behavior highly varies with time and cannot be learned from history data with high accuracy. To evaluate such cases, we repeat the experiments in Figure 7 (a) but perturb the probability values p_l and $q_{ll'}$ passed to the proposed algorithm with random noise of uniform distribution. If the actual probability value is v, the perturbed value ranges between v/ξ and $v \cdot \xi$, where $\xi \in \{1, 5\}$ is the noise factor. Clearly, the higher this factor is, the less accurate the estimates become. We find that the performance of the proposed algorithm degrades by up to 25%, with the worst case appearing for $\xi = 5$ and $C_n = 50\%$ of library size. On a positive note, our algorithm still outperforms Max-popularity and Femtocaching in all the experiments.

6 Related Work

Caching has been extensively studied in content distribution [50], IPTV [51], social [52] and conventional cellular networks [53]. However, this problem obtains an interesting new twist with the advent of small-cell networks since SBSs may have overlapping coverage areas and (unlike wireline networks) caches are not connected each other. The SBS caching problem has been studied from an optimization [4], [15], [16], an information theoretic [17], [18] and a game theoretic point of view [19]. The results span a wide range of techniques, such as discrete/convex optimization, facility location algorithms and matching games. The problem was reconsidered in [20] for the special case that videos encoded with quality scalability are cached at the SBSs. A mixed-timescale optimization of MIMO precoding and cache control was proposed in [21] to handle the scenario that SBSs cooperate when transmitting data to end-users. Finally, an architecture that performs caching at both the Core Network and the RAN (SBSs) along with a content-centric networking inspired caching algorithm was presented in [22]. *Our model differs from these works since they do not consider user mobility.*

Leveraging network coding for improving the efficiency of the caching policies has recently attracted considerable attention from academia. Leong et al. [27] studied the problem of deciding the amount of encoded data of a single file to store at each node in a network given a total storage budget constraint. Here, nodes fail independently one from the other and the goal is to maximize the probability of recovering the file from the non-failed nodes. A generalization for heterogeneous probabilities of failure was presented in [28]. Furthermore, recent work in [29] studied the above problem in a delay tolerant network setting where mobile users share the content cached at their caches when encounter each other within a given delay deadline. A similar study in vehicular networks was presented in [30]. However, *the methodologies used in all these works depend on the assumption that content access is independent across the caches.* Hence, they can only be used to analyze node failures or user encounters in delay-tolerant/vehicular networks. In contrast, *our work considers content access to follow a Markov chain model, which naturally represents user movements in a small-cell network.*

A mobility-aware caching scheme has already been presented in [54], but not for small-cell networks. Here, mobile users randomly connect to the leaf nodes in a cache hierarchy, where the cache space that was occupied by the content requested by a user is freed right after the user receives this content. This is a radically different model to ours, since it involves caches storing content for a short-term rather than long-term. The closest work to ours is that presented by Guan et al. in [46]. The authors assume that the *exact* trajectories are known a priori for a set of moving users and optimize SBS caching based on them. This seems to be an *over-optimistic* view of current mobility prediction mechanisms. In contrast, our algorithms take as input the *probabilities* of user movements from one location to another. Besides, the above two works make the simplifying assumption that as long as a user contacts a cache-node, the *complete* requested content can be downloaded. In contrast, we consider the realistic case that contact duration limits bottleneck content delivery.

This work generalizes the model presented in our initial study in [1] where the coverage areas of the SBSs were assumed to be non-overlapping and hence users could associate to at most one SBS at each slot. We need to emphasize though that SBSs are deployed today in low density and hence they rarely overlap. Also, if SBSs overlap, the interference problem will arise. Nevertheless, this generalization will be important in the future as the SBS density increases [12], [13] and progress in interference management is made. A second difference of this work is that the numerical analysis in [1] was based on (i) a mobility trace that keeps track of only the SBS that a user associates (rather than all the visible SBSs), and (ii) an artificial content popularity. Finally, with this paper we make our evaluation code publicly available for the benefit of the research community.

7 Conclusion

In this paper, we studied the impact of user mobility on content caching in small-cell networks aimed at minimizing the load of macro-cells. This is of major importance nowadays since more and more small-cells are deployed in macro-cells causing more frequent user hand-offs. To overcome the NP-Hardness nature of the revisited caching problem, we introduced a novel optimization framework based on a connection to the Markov chain model and evaluated its efficacy using traces of human mobility patterns and requests for Youtube videos. The results demonstrated that, for an operator having at his disposal a few days-old statistics of user mobility, this framework can serve as an important tool for offloading the macro-cell network.

References

[1] K. Poularakis and L. Tassiulas, "Exploiting User Mobility for Wireless Content Delivery", IEEE International Symposium on Information Theory (ISIT), pp. 1017-1021, 2013.
[2] Ericsson, "Mobility Report: On the Pulse of Networked Society", June 2015.
[3] J. G. Andrews, "Seven ways that hetnets are a cellular paradigm shift", IEEE Communications Magazine, vol. 51, no. 3, pp. 136-144, 2013.
[4] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch and G. Caire, "FemtoCaching: Wireless Video Content Delivery through Distributed Caching Helpers", IEEE Conference on Computer Communications (Infocom), pp. 1107-1115, 2012.
[5] Intel, "Rethinking the Small Cell Business Model", 2011.
[6] Mobile Europe, "Altobridge debuts intel-based network edge small cells caching solution", June 2013.
[7] Light Reading, "NSN Adds ChinaCache Smarts to Liquid Applications", 2014, http://www.lightreading.com/mobile/4g-lte/-nsn-adds-chinacache-smarts-to-liquid-applications-/d/d-id/708280.
[8] Saguna, "Saguna Open-RAN", 2015, http://www.saguna.net/products/saguna-open-ran.
[9] H. Holma and A. Toskala, "LTE-Advanced: 3GPP Solution for IMT-Advanced", Wiley, October 2012.
[10] A. J. Nicholson and B. D. Noble, "Breadcrumbs: Forecasting mobile connectivity", ACM International Conference on Mobile Computing and Networking (MobiCom), pp. 46-57, 2008.
[11] J. Pang, B. Greenstein, M. Kaminsky, D. McCoy, and S. Seshan, "Wifi- reports: Improving wireless network selection with collaboration", IEEE Transactions on Mobile Computing, vol. 9, no. 12, pp. 1713-1731, 2010.
[12] Y. Zhu, Z. Zhang, Z. Marzi, C. Nelson, U. Madhow, B.Y. Zhao, H. Zheng, "Demystifying 60GHz Outdoor Picocells", ACM International Conference on Mobile Computing and Networking (MobiCom), pp. 5-16, 2014.
[13] Qualcomm, "Enabling Hyper-Dense Small Cell Deployments with UltraSON", https://www.qualcomm.com/documents/enabling-hyper-dense-small-cell-deployments-ultrason
[14] JPL's Wireless Communication Reference Website, Cell Sizes, http://www.wirelesscommunication.nl/reference/chaptr04/cellplan/cellsize.htm
[15] 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", IEEE Conference on Computer Communications (Infocom), pp. 936-944, 2015.
[16] K. Poularakis, G. Iosifidis, L. Tassiulas, "Approximation Algorithms for Mobile Data Caching in Small Cell Networks", IEEE Transactions on Communications, vol 62, no. 10, pp. 3665-3677, 2014.
[17] M.A. Maddah-Ali, U. Niesen, "Fundamental Limits of Caching", IEEE Transactions on Information Theory, vol 60, no. 5, pp. 2856-2867, 2014.

[18] J. Hachem, N. Karamchandani, and S. Diggavi, "Content Caching and Delivery over Heterogeneous Wireless Networks", IEEE Conference on Computer Communications (Infocom), pp. 756-764, 2015.
[19] K. Hamidouche, W. Saad and M. Debbah, "Many-to-Many Matching Games for Proactive Social-Caching in Wireless Small Cell Networks", International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (Wiopt), pp. 569-574, 2014.
[20] K. Poularakis, G. Iosifidis, A. Argyriou, L. Tassiulas, "Video Delivery over Heterogeneous Cellular Networks: Optimizing Cost and Performance", IEEE Conference on Computer Communications (Infocom), pp. 1078 - 1086, 2014.
[21] A. Liu and V. K. N. Lau, "Mixed-timescale precoding and cache control in cached MIMO interference network", IEEE Transactions on Signal Processing, vol. 61, no. 24, pp. 6320-6332, 2013.
[22] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the Air: Exploiting Content Caching and Delivery Techniques for 5G Systems", IEEE Communications Magazine, vol. 52, no. 2, pp. 131-139, 2014.
[23] C. Song, Z. Qu, N. Blumm, and A. Barabasi, "Limits of predictability in human mobility", Science, vol. 327, pp. 1018-1021, 2010.
[24] K. Lee, J. Lee, Y. Yi, I. Rhee, S. Chong, "Mobile Data Offloading: How Much Can WiFi Deliver?", IEEE/ACM Transactions on Networking, vol. 21, no. 2, pp. 536-550, 2012.
[25] CPLEX: Linear Programming Solver. [Online]. Available: http://www.ilog.com.
[26] S. Gambs, M. Killijian, M. Cortez, "Next Place Prediction using Mobility Markov Chains", Workshop on Measurement, Privacy, and Mobility, pp. 3:1-3:6, 2012.
[27] D. Leong, A. G. Dimakis, and T. Ho, "Distributed storage allocations", IEEE Transactions on Information Theory, vol. 58, no. 7, pp. 4733-4752, 2012.
[28] V. Ntranos, G. Caire, A. Dimakis, "Allocations for Heterogenous Distributed Storage", IEEE International Symposium on Information Theory (ISIT), pp. 2761-2765, 2012.
[29] X. Zhuo, Q. Li, W. Gao, G. Cao, Y. Dai, "Contact Duration Aware Data Replication in Delay Tolerant Networks", IEEE International Conference on Network Protocols (ICNP), pp. 236-245, 2011.
[30] M. Sathiamoorthy, A.G. Dimakis, B. Krishnamachari, Fan Bai, "Distributed Storage Codes Reduce Latency in Vehicular Networks", IEEE Conference on Computer Communications (Infocom), pp. 2646-2650, 2012.
[31] R.Tewari, H. Vin, A. Dan, D. Sitaram, "Resource based caching for web servers", SPIE/ACM Conference on Multimedia Computing and Networking, pp. 191-204, 1998.
[32] S. Sen, J. Rexford, D. Towsley, "Proxy prefix caching for multimedia streams", IEEE Conference on Computer Communications (Infocom), pp. 1310-1319, 1999.
[33] K. Wu, P. Yu, J. Wolf, "Segment based proxy caching of multimedia streams", International Conference on World Wide Web (WWW), pp. 36-44, 2001.
[34] M. Garey, D. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. Freeman & Comp., San Francisco, 1979.
[35] D. Ashbrook, T. Starner, "Learning Significant Locations and Predicting User Movement with GPS", International Symposium on Wearable Computers (ISWC), pp. 101-108, 2002.
[36] E. Bastug, J. L. Guenego, and M. Debbah, "Proactive Small Cell Networks", International Conference on Telecommunications (ICT), pp. 1-5, 2013.
[37] P. Blasco and D. Gunduz, "Learning-Based Optimization of Cache Content in a Small Cell Base Station", IEEE International Conference on Communications (ICC), pp. 1897-1903, 2014.
[38] K. Chung, H. Lam, Z. Liu, M. Mitzenmacher, "Chernoff-Hoeffding Bounds for Markov Chains: Generalized and Simplified", International Symposium on Theoretical Aspects of Computer Science (STACS), pp. 124-135, 2012.
[39] D. Bertsimas and J. N. Tsitsiklis, "Introduction to Linear Optimization", Belmont, MA: Athena Science, 1997.
[40] G. Dantzig, "Discrete-Variable Extremum Problems", Operations Research Vol. 5, No. 2, pp. 266-288, 1957.
[41] Michael J. Quinn, Parallel computing (2nd ed.): theory and practice, McGraw-Hill, Inc., New York, NY, 1994.
[42] 3GPP Specification, "Access Network Discovery and Selection Function Management Object", http://www.3gpp.org/dynareport/24312.htm
[43] P. Xia, Jo Han-Shin, J.G. Andrews, "Fundamentals of Inter-Cell Overhead Signaling in Heterogeneous Cellular Networks", IEEE Journal of Selected Topics in Signal Processing, pp. 257-269, 2011.
[44] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet", IEEE MultiMedia, vol. 18, no. 4, pp. 62-67, 2011.
[45] M. McNett and G. M. Voelker, "Access and mobility of wireless pda users", ACM SIGMOBILE Mobile Computing and Communications Review, vol. 9, no. 2, pp. 40-55, Apr. 2005.
[46] Y. Guan, Y. Xiao, H. Feng, C-C. Shen, L. J. Cimini Jr. "MobiCacher: Mobility-Aware Content Caching in Small-Cell Networks", IEEE Global Communications Conference, pp. 4537-4542, 2014.
[47] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch global, cache local: YouTube network traffic at a campus network-measurements and implications", SPIE/ACM Multimedia Computing and Networking Conference (MMCN), 2008.
[48] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 20142019, White Paper, February 2015.
[49] K. Poularakis and L. Tassiulas, Publicly available code, https://www.dropbox.com/s/whc24gcfmmti76j/tmccode.rar?dl=0
[50] S. Borst, V. Gupta, and A. Walid, "Distributed Caching Algorithms for Content Distribution Network", IEEE Conference on Computer Communications (Infocom), pp. 1-9, 2010.
[51] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative Hierarchical Caching with Dynamic Request Routing for Massive Content Distribution", IEEE Conference on Computer Communications (Infocom), pp. 2444-2452, 2012.
[52] M. Taghizadeh, K. Micinski, C. Ofria, E. Torng, S. Biswas, "Distributed Cooperative Caching in Social Wireless Networks", IEEE Transactions on Mobile Computing, vol. 12, no. 6, pp. 1037-1053, 2013.
[53] J. Dai, F. Liu, Bo Li, B. Li, J. Liu, "Collaborative Caching in Wireless Video Streaming Through Resource Auctions", IEEE Journal on Selected Areas in Communications, vol. 30, no. 2, pp. 458-466, 2012.
[54] V.A. Siris, X. Vasilakos, G. C. Polyzos, "Efficient Proactive Caching for Supporting Seamless Mobility", arXiv:1404.4754, 2014.

Konstantinos Poularakis obtained the Diploma and the M.S. degree in Computer & Communications Engineering from University of Thessaly, Greece, in 2011 and 2013, respectively. Currently, he is a Ph.D. candidate in the Dept. of Electrical & Computer Engineering, University of Thessaly, Greece. He has been honored with several awards and scholarship during his studies, from sources including the Greek State Scholarships foundation (IKY), the Center for Research and Technology Hellas (CERTH) and the "Alexander S. Onassis Public Benefit Foundation". In spring 2014 he was a Research Intern at the Technicolor Paris Research Lab. His research interests lie in the broad area of network optimization with emphasis on mobile data caching and network coding.

Leandros Tassiulas (S'89-M'91-SM'06-F'07), the John C. Malone Professor of Electrical Engineering at Yale University, obtained the Diploma in Electrical Engineering from the Aristotelian University of Thessaloniki, Greece in 1987, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Maryland, College Park, in 1989 and 1991, respectively. He has held positions as Assistant Professor at Polytechnic University New York (1991-1995), Assistant and Associate Professor University of Maryland College Park (1995-2001), and Professor at University of Ioannina (1999-2001) and University of Thessaly (2002-2015), Greece. His research interests are in the field of computer and communication networks with emphasis on fundamental mathematical models, architectures and protocols of wireless systems, sensor networks, high-speed internet and satellite communications. Dr. Tassiulas is a Fellow of IEEE. He received a National Science Foundation (NSF) Research Initiation Award in 1992, an NSF CAREER Award in 1995 an Office of Naval Research, Young Investigator Award in 1997 and a Bodosaki Foundation award in 1999. He also received the INFOCOM 1994 best paper award, the INFOCOM 2007 achievement award, and the IEEE 2016 Koji Kobayashi Computers and Communication Award.