In-Network Caching and Content Placement in Cooperative Small Cell Networks

Francesco Pantisano[1], Mehdi Bennis[2], Walid Saad[3], and Mérouane Debbah[4]

[1]JRC - Joint Research Centre, European Commission, Ispra, Italy, email: francesco.pantisano@jrc.ec.europa.eu
[2]CWC - Centre for Wireless Communications, Oulu, Finland, email: bennis@ee.oulu.fi
[3]Wireless@VT, Bradley Department of Electrical and Computer Engineering, Blacksburg, VA, USA, email: walids@vt.edu
[4] Mathematical and Algorithmic Sciences Lab, Huawei France R&D, Paris, France, email: merouane.debbah@huawei.com

***Abstract*—Anticipating multimedia file requests via caching at the small cell base stations (SBSs) has emerged as a promising technique for enhancing the quality-of-service (QoS) of cellular user equipments (UEs). Nevertheless, in traditional caching approaches, files are retrieved from evolved packet core (EPC) of the network, and without coordination among SBSs, which introduces new challenges of content duplication and unbalanced traffic load on the backhaul. In this paper, we propose a collaborative framework in which the SBSs can access files from the caches of other SBSs within the same network domain (i.e., connected to the same service gateway). We design a cost model for the content retrieval of contents across SBSs and from the EPC, and we propose a cost-aware decentralized algorithm, based on which the SBSs devise individual caching strategies (i.e., which files to cache, and from where). Simulation results show that the proposed cooperative caching scheme yields significant gains in terms of in-network content availability, reaching up to 21% improvement compared to a traditional approaches based on geographical distribution of the UEs.**

I. Introduction

Meeting the stringent quality-of-service (QoS) requirements of emerging multimedia services has led to the introduction of novel decentralized wireless cellular architectures, such as those based on the concept of small cell base stations (SBSs), such as picocells, microcells or femtocells. SBS deployments promise to deliver high QoS, at low operational costs [1], yet, in order to reap those benefits, a number of technical challenges must be addressed, notably in the field of load balancing and limited backhaul capacity [2].

To overcome the backhaul capacity limitations, state-of-the-art SBS architectures propose the integration of data storage units and offloading techniques, based on data *caching*. Caching has been originally proposed in content distribution networks for enhancing data locality, i.e., by content replication at strategic nodes of the network (e.g., proxy servers, gateways), while balancing the network traffic during off-peak intervals [3]. In essence, by decoupling the time instant in which a file content is downloaded, from the one in which it is delivered to a UE, an SBS can boost the QoS experienced by its users while saving backhaul resources.

Caching for SBS networks entails a number of technical challenges. Notably, retrieving files from the respective content providers incurs a cost in terms of backhaul bandwidth utilization. Such a cost is typically different for each SBS, based on their current backhaul bandwidth availability. Also, due to the limited storage capacity of each memory units, each SBS is required to implement a decision policy – so as to select which files to cache – and a replacement policy – for updating the cache composition, when the disk space is full. The complexity of such operations is exacerbated by the fact that each SBS performs caching operations independently and without coordination, which introduces issues of content duplication and unbalanced backhaul load. In summary, implementing caching at SBS level needs to carefully account for the cost for retrieving the files to cache, balance the resulting traffic load increase on the backhaul, and to manage the content duplication.

A promising alternative to address the above challenges is to allow the SBSs to implement cooperative caching mechanisms, such as *in-network caching* techniques [3], [4]. The rationale behind cooperative caching is that if two SBSs are aware of their respective cache composition, they can pool resources, and coordinate future requests. In-network caching is a specific case in which the cooperative SBSs are connected to the same service gateway, which defines a network domain, as depicted in Figure 1. Exchanging information among SBSs in the same network domain does not involve higher level network elements, such as the packet gateways in the evolved packet core (EPC) of the network. As a result, in-network caching enable the SBSs to devise cooperative caching policy, pool resources and increase the hit-ratio (i.e., the probability that a file requested by a UE is found in the cache of its serving SBS), while harnessing the backhaul congestion at EPC network elements.

The main contribution of this paper is to propose a novel, decentralized caching strategy for coordinating the content retrieval in cooperative small cell networks. The proposed solution enables the SBSs to make individual decisions on *which* SBS they can cooperate with – based on their cache composition – and *how* to update the respective caches, based on the cost of file retrieval, and their respective backhaul bandwidth availability. To solve this problem, we formulate a cost model for retrieving files from the EPC or from neighboring SBSs. Based on this cost model, we propose a heuristic algorithm and we compute its performance lower-bound. Simulation results show that, in the proposed cooperative caching approach, the SBSs overcome the backhaul capacity limitations and improve the UE's QoS delivery of traditional UE-SBS associations, yielding gains of up to 21%.

The research leading to this paper has been partly supported by the Celtic-Plus project SHARING (proj. C2012/1-8), the U.S. National Science Foundation under grants CNS-1460316, CNS-1460333, CNS-1443914, CNS-1446621, and AST-1443913, and the ERC Starting Grant 305123 MORE.

5GU 2014, November 26-27, Levi, Finland

DOI 10.4108/icst.5gu.2014.258230

The rest of this paper is organized as follows. In Section II, we present the related work and our contributions. In Section III, we introduce the system model and formulate the in-network cooperative caching problem. In Section IV, we present a decomposition of the original problem and propose an algorithm that implements the cooperative caching in a distributed way. Simulation results are analyzed in Section V. Finally, conclusions are drawn in Section VI.

II. Related Work and Contribution

In the existing literature on small cell networks, a number of works have delved into the aspects of caching. As the efficiency of caching depends on the ability of each SBS to intelligently select which files to store, i.e., to enhance the hit-ratio of content requests, several works have aimed at maximizing the hit-ratio, by adequately dimensioning the SBSs' caches [4], or based on the mobility estimation of UEs [5]. The file selection and content placement, i.e., the most suitable SBSs in which the files should be cached have been addressed in [3], [6], based on the file popularity distribution. In this respect, additional information on the spatial and social links across the users has been shown to further improve the QoS by tracking the data content popularity at the SBSs [7], [8].

With concern to the methodologies, caching optimization frameworks have been proposed in [3], [6], [9] but an underlying assumption is that the network optimization is pursued by taking individual decision at each SBS. The aforementioned works deploy cost-unaware performance indicators (e.g., hit-ratio or number of hops), which do not account for the cost of file retrieval. Moreover, such works assume decentralized, uncooperative caching operations at each SBS. Unlike those works, we propose a cost-aware caching mechanism in which the SBSs can cooperate so as to pool their caches and optimize the file selection, while accounting for a cost of file retrieval. As the SBS network architecture includes communication interfaces among SBSs (i.e. X2, interface) and between SBSs and gateways (i.e., S1 interface), 3GPP specification has proposed local IP access (LIPA) and selective IP traffic offload (SIPTO) as effective solutions for enabling coordinated policies among neighboring SBSs, notably in the context of proactive caching [10]. Our work is aligned to this research direction, and in summary, our main contributions are:

- We design an in-network caching mechanism for SBSs, which accounts for both storage and backhaul capacity limitations.
- We evaluate the worst-case performance of the proposed solution and discuss possible upgrades.
- Main caching performance indicators are evaluated and compared with those of other benchmark caching schemes.

III. System Model

Consider the *downlink* transmission of a single orthogonal frequency division multiple access (OFDMA) macro-cell. In this network, M mobile UEs and N SBSs are deployed, respectively denoted by the sets $\mathcal{M} = \{1, ..., M\}$ and $\mathcal{N} = \{1, ..., N\}$. Let $\mathcal{L}_i$ denote the set of UEs serviced by SBS i. The SBSs are connected to the core network via a backhaul of capacity B. Each UE m requests a set of files $\mathcal{F}_m = \{1, ..., F_m\}$, $\mathcal{F}_m \subset \mathcal{F}$. For simplicity, we assume that all files have the same size s. The request probability associated to each file $f \in \mathcal{F}$, with popularity rank k across the UEs in $\mathcal{M}$, is assumed to follow a Zipf distribution with skewness parameter ψ [11]. Thus, each UE m requests file f with probability $\rho_{m,f} = \frac{f^{-\psi}}{\sum_x^{|\mathcal{F}|} \frac{1}{x}^{\psi}}$, $x \in \mathcal{F}$, which we assume it is known.

In order to accommodate the UE's traffic requests, each SBS allocates a backhaul bandwidth $B_{i,m}$ to each UE $m \in \mathcal{L}_i$, such that $\sum_{m \in \mathcal{L}_i} B_{i,m} \leq B$. When a file requested by UE $m \in \mathcal{L}_i$ is not locally available at its SBS, it is downloaded from the content provider, through the backhaul at constant rate $B_{i,m}$. Due to the fact that the number of serviced UEs $|\mathcal{L}_i|$ varies at each small cell, each SBS i experiences a different backhaul bandwidth availability, defined as:

$$B_i = B - \sum_{m \in \mathcal{L}_i} B_{i,m}. \qquad (1)$$

Each SBS is equipped with a data storage unit having a capacity of K_i bytes and the set of files that are locally stored at SBS i is denoted by $\mathcal{D}_i = \{1, \ldots, D_i\}$. This caching procedure can continue until the storage capacity K_i is exhausted. Upon reaching the maximum storage capacity K_i, the least popular files are systematically dropped to accommodate new file entries, while verifying the storage capacity constraint:

$$D_i \cdot s \leq K_i \;\; [bits]. \qquad (2)$$

In traditional networks, the files requested by an UE that are not available in the cache of its serving SBS are downloaded from a content provider via the backhaul. Clearly, this operation entails a number of challenges, such as the verification of delay or data rate constraints, which are greatly affected by the backhaul traffic congestion. Moreover, files stored in remote content providers are generally processed by a number of packet gateways, each one introducing a delay, whose estimation is difficult. As a result, a caching approach which maximizes the hit-ratio cannot ensure high QoS delivery, and it can compromise the QoS of uncached file requests. To alleviate this issue, an alternative is to allow the SBSs to retrieve files also from the caches of other SBSs in the same network domain (i.e., from other SBSs which are connected to the same network gateway) other than traditional content providers, as depicted in Figure 1. Each service gateway is connected to a smaller set of SBSs, defining a network domain. The file exchange across SBS of the same domain only involves the service gateway of that domain, which limits the delay and does not affect traffic congestion in the EPC. In this work, we consider that, within each network domain, the SBSs can form cooperative groups, called *coalitions*, and jointly decide on which files to cache. Hence, given the file requests of each UE, each coalition is formed so as to exploit cache diversification and increase the hit-ratio.

We now introduce some preliminary definitions. Based on the file popularity distribution and the caching policy at each SBS, it is possible to determine the probability $\pi^L_{m,i,f}$ that a file f, requested by UE m, is available at a local SBS i [6]. Hence, the probability that a UE m requests an uncached file $f \notin \mathcal{D}_i$ is given by $\pi^M_{m,i,f} = 1 - \pi^L_{m,i,f}$. In-network caching mechanisms add a third scenario to the above two. A file which is requested

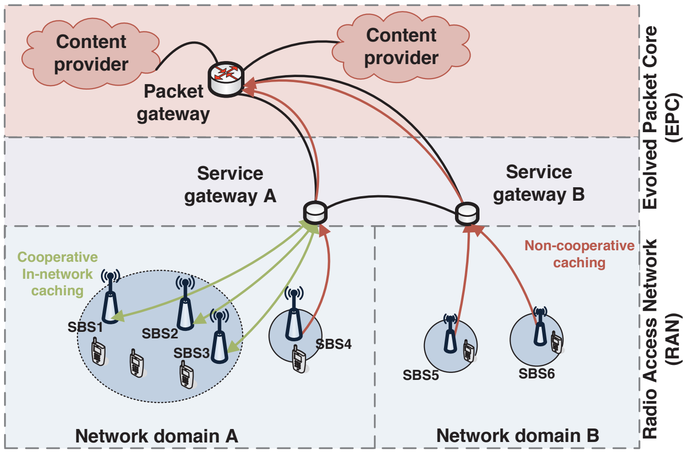

Fig. 1: Network architecture and in-network caching. The network is partitioned into two network domains. In domain A, SBS $\{1, 2, 3\}$ have pooled their caches, based on the requirements of their respective UEs. SBS 4 and $5, 6$ cache their files from the EPC. Accordingly, network domains A and B are respectively partitioned into $\{(SBS_1, SBS2, SBS_3), (SBS_4)\}$ and $\{(SBS_5), (SBS_6)\}$.

at SBS i, and is thereby unavailable, can be found in the cache of SBS j (in the same network domain) with probability $\pi^R_{m,i,f}$. In summary, the three probabilities that a file requested by UE m, serviced by SBS i is locally cached, or cached at another cooperative SBS j, or unavailable are respectively expressed by:

$$\pi^L_{m,i,f} = \text{Prob}\{f \in \mathcal{D}_i | f \in \mathcal{F}_m, m \in \mathcal{L}_i\}, \quad (3)$$

$$\pi^R_{m,i,f} = \text{Prob}\{f \in \mathcal{D}_j | f \in \mathcal{F}_m, m \in \mathcal{L}_i, \{i, j\} \in S\}, \quad (4)$$

$$\pi^M_{m,i,f} = 1 - \pi^L_{m,i,f} - \pi^R_{m,i,f}. \quad (5)$$

In order to formulate an optimization problem for the proposed in-network caching architecture, we introduce a cost model for the file acquisition. Specifically, when a copy of file f is available at the local cache of an SBS i, the cost for SBS i to acquire file f is equal to zero. Instead, retrieving a file that is available at another SBS j in the same coalition $\mathcal{S}$ has a cost $c_{m,i,f} = d^R(\mathcal{S})$ that depends on the number of cooperating SBSs. In fact, for the proposed in-network caching mechanism, we assume that such a cost depends on the number of requests managed by the service gateway, and thus, the cost grows linearly with $|\mathcal{S}|$. Finally, when a file is unavailable both locally and within a coalition, the cost for its acquisition is d^M, where $d^M > d^R(\mathcal{S})$. Namely, the cost of retrieving files from the cache of another SBS within the network domain (e.g., serviced by the same network gateway) is smaller than fetching data from the content provider, in which the number of network gateways and the routing path may yield intolerable delays. In summary, the cost for obtaining file $f \in \mathcal{F}_m$, of UE m, serviced by SBS i in coalition $\mathcal{S}$ is:

$$c_{m,i,f} = \begin{cases} 0, & f \in \mathcal{D}_i, \\ d^R(\mathcal{S}), & f \in \mathcal{D}_j,\ \{i, j\} \in \mathcal{S}, \\ d^M, & f \notin \mathcal{D}_i,\ \forall i \in \mathcal{S}. \end{cases} \quad (6)$$

Given the above cost model, the overall network cost for delivering the files requested by the UEs to their respective SBSs is:

$$\begin{aligned} &\sum_{i\in\mathcal{N}} \sum_{m\in\mathcal{M}} \sum_{f\in\mathcal{F}_m} c_{m,i,f} \\ &= \sum_{i\in\mathcal{N}} \sum_{m\in\mathcal{M}} \sum_{f\in\mathcal{F}_m} d^R(\mathcal{S})\pi^R_{m,i,f} + d^M \pi^M_{m,i,f} \\ &= \sum_{i\in\mathcal{N}} \sum_{m\in\mathcal{M}} \sum_{f\in\mathcal{F}_m} d^R(\mathcal{S})\pi^R_{m,i,f} + d^M (1 - \pi^L_{m,i,f} - \pi^R_{m,i,f}) \\ &= \sum_{i\in\mathcal{N}} \sum_{m\in\mathcal{M}} \sum_{f\in\mathcal{F}_m} d^M (1 - \pi^L_{m,i,f}) - (d^M - d^R(\mathcal{S}))\pi^R_{m,i,f}. \end{aligned} \quad (7)$$

Note that the first term in (7), i.e., $\sum_{i\in\mathcal{N}} \sum_{m\in\mathcal{M}} \sum_{f\in\mathcal{F}_m} d^M (1 - \pi^L_{m,i,f})$ represents the cost for retrieving the missing files by each single SBS, in a non-cooperative approach. For example, a traditional approach for minimizing this cost is by caching the files which maximize the hit-ratio, as shown in [6]. In this work, we focus on the second term, which we define *cooperative caching utility* U, that represents the cost that can be *saved* by cooperative caching management of the SBSs.

In such a setting, we aim at finding a *network partition*, i.e., a collection of disjoint, nonempty coalitions, $P(\mathcal{N}) : \{\mathcal{S}_1, \mathcal{S}_2, \dots\} \in \mathcal{N}$, and an file selection strategy $\mathcal{D}_i$ at each SBS, that maximize the utility U, by considering the limitations on the backhaul capacity and the storage capabilities at each SBS. Essentially, this yields the following optimization problem:

$$\underset{P(\mathcal{N}), \mathcal{D}_i}{\arg\max}\ U = \sum_{i\in\mathcal{N}} \sum_{m\in\mathcal{M}} \sum_{f\in\mathcal{F}_m} (d^M - d^R(\mathcal{S}))\pi^R_{m,i,f} \quad (8)$$

$$\text{s.t.,} \quad D_i \cdot s \leq K_i, \quad (9)$$

$$B_i > 0, \ \forall i \in \mathcal{N} \quad (10)$$

In terms of complexity, the optimization problem in (8–10) is NP-complete. Even by relaxing some of the constraints, the exponential complexity makes a centralized approach intractable, notably in small cell networks in which the number of UEs and SBSs can considerably grow. The problem complexity coupled with the need for self-organizing solutions in small cells mandate distributed approaches in which the SBSs can autonomously decide on in-network caching policy, based on available backhaul bandwidth and storage capacity, as we will present in the following section.

IV. Proposed Algorithm for Cluster Formation

For the sake of mathematical tractability, we decompose the original problem into sub-problems, each one focusing on a smaller set of SBSs $\mathcal{R} \subset \mathcal{N}$ in the same network domain (i.e., connected to the same service gateway). In each sub-problem, we aim at finding a partition $P(\mathcal{R}) : \{S_1, S_2, \dots\} \in \mathcal{R}$, in which the SBS $i \in \mathcal{R}$ join the coalition $\mathcal{S}$ in order to maximize the coalitional caching utility $U_{\mathcal{S}}$, as defined in the following optimization problem:

$$\underset{P(\mathcal{R}), \mathcal{D}_i}{\arg\max}\ U_{\mathcal{S}} = \sum_{i\in\mathcal{S}} \sum_{m\in\mathcal{L}_i} \sum_{f\in\mathcal{F}_m \cap \mathcal{D}_i} \left(d^M - d^R(\mathcal{S})\right) \quad (11)$$

Algorithm 1: Proposed Algorithm for In-Network Cache Management

```
Data: $\mathcal{L}_i$, $\rho_{m,f}$, $\Pi_{\mathcal{N}} = \mathcal{N}$, $\mathcal{F}_m$.
Phase I - SBS Discovery;
• $\mathcal{S} = \{i\}$;
• Each SBS $i$ discovers the SBSs in its network domain $\mathcal{R}$;
• Each SBS $i$ sorts the found SBSs $j$ by their residual bandwidth $B_j$;
Phase II - Cooperative Cache Management;
for $m \in \mathcal{L}_i$ do
    • $j \leftarrow \arg\max_{j} \sum_{f\in\mathcal{F}_m\cap\mathcal{D}_i} \left(d^M - d^R(\{i,j\})\right)$;
    • $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$;
    while $|\mathcal{D}_i| \leq K_i$ and $B_i > 0$ do
        • $j \leftarrow \arg\max B_j$ ;
        • $f \leftarrow \arg\max_{f} \sum_{i\in\mathcal{S}} \sum_{m\in\mathcal{L}_i} \sum_{f\in\mathcal{F}_m\cap\mathcal{D}_i} \left(d^M - d^R(\mathcal{S})\right)$;
        • $\mathcal{D}_j \leftarrow \mathcal{D}_j \cup \{f\}$;
    end
end
Result: $\mathcal{S}, \mathcal{D}_i, i \in \mathcal{S}$.
```

$$\text{s.t.,} \quad D_i \cdot s \leq K_i, \tag{12}$$

$$B_i > 0, \ \forall i \in \mathcal{S}. \tag{13}$$

In order to find a solution for the problem in (11), we propose Algorithm 1. Algorithm 1 consists of two main phases: SBS discovery and cooperative cache management. Initially, each SBS is servicing a set of UEs $\mathcal{L}_i$ and has cached a set of files $\mathcal{D}_i$ (which can be also be an empty set or randomly formed). Then, each SBS i discovers the SBSs $j \in \mathcal{R}$ in the same network domain, using standard techniques such as in [2]. Next, each SBS sorts the neighboring SBSs in $\mathcal{R}$ by their backhaul bandwidth availability. In the second phase, SBS i sends a proposal to the SBS $j \in \mathcal{R}$ which can provide most of the files originally requested by the UE $m \in \mathcal{L}_i$. The cost $d^R(\mathcal{S})$ is updated accordingly. With respect to the content selection, the SBS with the largest available backhaul bandwidth is appointed to the selection of the files which maximize the cooperative caching utility of $\mathcal{S}$ as in (11). Next, the SBSs periodically update their respective utilities according to the current cache composition. Therefore, based on the requests of the UEs $m \in \mathcal{L}_i$, the outcome of Algorithm 1 is a set of cooperative SBSs $\mathcal{S}$, and the composition of their respective caches $\mathcal{D}_i$.

V. Algorithm Properties

Proposition 1. *The worst-case performance $\tilde{U}_{\mathcal{S}}$ of Algorithm 1, with respect to the optimal solution U^* of the problem in (8), is such that: $\frac{U^*}{S} \leq \tilde{U}_{\mathcal{S}}$, where S is the maximum number of iterations of Algorithm 1 required at each SBS.*

Proof. Let $\tilde{U}_{\mathcal{S}}$ and $\tilde{\mathcal{D}}_i$ respectively denote the utility obtained though our proposed algorithm, and the set of files selected at each SBS in a coalition $\mathcal{S}$. Accordingly, the utility in (11) can be expressed in terms of the files $f \in \tilde{\mathcal{D}}_i$ that are stored at each SBS $i \in \mathcal{S}$:

$$\tilde{U}_{\mathcal{S}} = \sum_{i\in\mathcal{S}} \sum_{m\in\mathcal{L}_i} \Big\{ \sum_{f\in\mathcal{F}_m\cap\tilde{\mathcal{D}}_i} d^M - d^R(\mathcal{S}) \Big\}. \tag{14}$$

We now introduce a binary variable $I_{f,i}$ whose value is 1 when an file $f \in \mathcal{F}_m$ is available at SBS i, and 0 otherwise. Accordingly, we can rewrite the utility $\tilde{U}_{\mathcal{S}}$ as:

$$\tilde{U}_{\mathcal{S}} = \sum_{m\in\mathcal{L}_i} \sum_{f\in\mathcal{F}_m} \Big\{ (d^M - d^R(\mathcal{S})) \cdot \min\Big(1, \sum_{i\in\mathcal{S}} I_{f,i}\Big) \Big\} \tag{15}$$

Now, let us denote the optimal collaborative caching utility by U^* and use $\{I^*_{f,i}\}$ to denote the optimal selection of files to be cached at each SBS, and $\mathcal{S}^*$ the respective coalition, leading to the utility maximization in (11). In order to show the worst-case performance of our algorithm, we need to demonstrate that $\frac{U^*}{\tilde{U}_{\mathcal{S}}} \leq S$, where $S = |\mathcal{S}|$ denotes the maximum number of SBSs that SBS i has to interrogate in order to obtain a file $f \in \mathcal{F}_m$ which it has not been cached. By noting that, $\min(1, \sum_x^N a_x) \geq \frac{1}{N}\sum_x^N \min(1, a_x)$, for a generic binary variable a_x, we can rewrite (15) as:

$$\begin{aligned}\tilde{U}_{\mathcal{S}} &\geq \sum_{m\in\mathcal{L}_i} \sum_{f\in\mathcal{F}_m} \Big\{ \big(d^M - d^R(\mathcal{S})\big) \cdot \frac{1}{S} \sum_{i\in\mathcal{S}} \min(1, I_{f,i}) \Big\} \\ &= \frac{1}{S} \sum_{i\in\mathcal{S}} \sum_{m\in\mathcal{L}_i} \sum_{f\in\mathcal{F}_m} \big\{ \big(d^M - d^R(\mathcal{S})\big) \cdot \min(1, I_{f,i}) \big\}. \end{aligned} \tag{16}$$

Note that $\sum_{i\in\mathcal{S}}\sum_{m\in\mathcal{L}_i}\sum_{f\in\mathcal{F}_m}\{\big((d^M - d^R(\mathcal{S})\big) \cdot \min(1, I_{f,i})\}$ in (16) is equivalent to (11), when the latter is formulated using binary variables $I_{f,i}$. Since $\{I^*_{f,i}\}$ are chosen to maximize (8) as opposed to (11), the following inequality holds:

$$\begin{aligned} &\sum_{i\in\mathcal{S}} \sum_{m\in\mathcal{L}_i} \sum_{f\in\mathcal{F}_m} \{(d^M - d^R(\mathcal{S}) \cdot \min(1, I_{f,i})\} \geq \\ &\sum_{i\in\mathcal{S}} \sum_{m\in\mathcal{M}} \sum_{f\in\mathcal{F}_m} \{(d^M - d^R(\mathcal{S}) \cdot \min(1, I^*_{f,j})\}. \end{aligned} \tag{17}$$

Combining the results in (16) and (17) yields:

$$\begin{aligned}\tilde{U}_{\mathcal{S}} &\geq \frac{1}{S} \sum_{j\in\mathcal{S}} \sum_{m\in\mathcal{L}_i} \sum_{f\in\mathcal{F}_m} \{(d^M - d^R(\mathcal{S}) \cdot \min(1, I^*_{f,j})\} \\ &= \frac{1}{S} \sum_{m\in\mathcal{L}_i} \sum_{f\in\mathcal{F}_m} \Big\{ (d^M - d^R(\mathcal{S})) \cdot \sum_{j\in\mathcal{S}} \min(1, I^*_{f,j}) \Big\} \\ &\geq \frac{1}{S} \sum_{m\in\mathcal{L}_i} \sum_{f\in\mathcal{F}_m} \Big\{ (d^M - d^R(\mathcal{S})) \cdot \min\Big(1, \sum_{j\in\mathcal{S}} I^*_{f,j}\Big) \Big\} \\ &= \frac{1}{S} U^*. \end{aligned} \tag{18}$$

in which the last inequality holds as $\sum_x^N \min a_x \geq \min\left(1, \sum_x^N a_x\right)$, where a_x is a binary variable. Combining (18) with the definition of utility in (15), it can be noted that $\tilde{U}_{\mathcal{S}} \geq \frac{1}{S} U^*$, or equivalently, $\frac{U^*}{S} \leq \tilde{U}_{\mathcal{S}}$. □

Proposition 2. *The complexity of Algorithm 1 is polynomial and scales in the order of $O(S \cdot D_i \cdot |\mathcal{L}_i|)$.*

Proof. The number of iterations depends solely depends on the number of potential cooperative partners, the number of file requests for each serviced UE (one instance per UE) and the number of cached files at each SBS. □

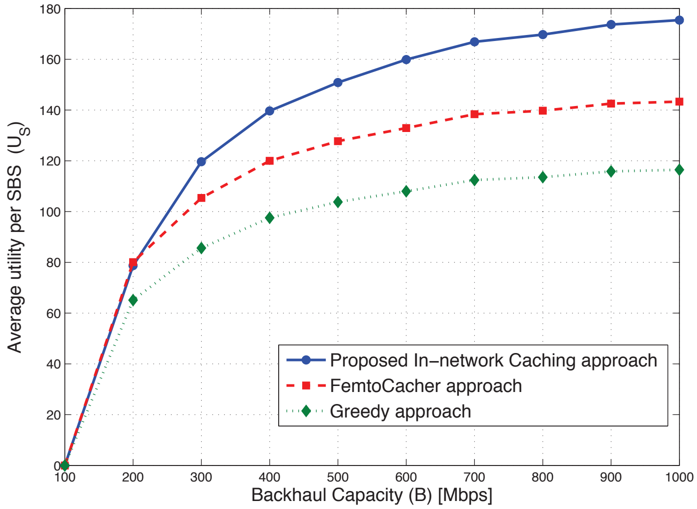

Fig. 2: Average utility U_S per SBS vs backhaul bandwidth. $|\mathcal{L}_i| = 2$ UEs, $S = 3$ SBS, $K_i = 5$ TB.

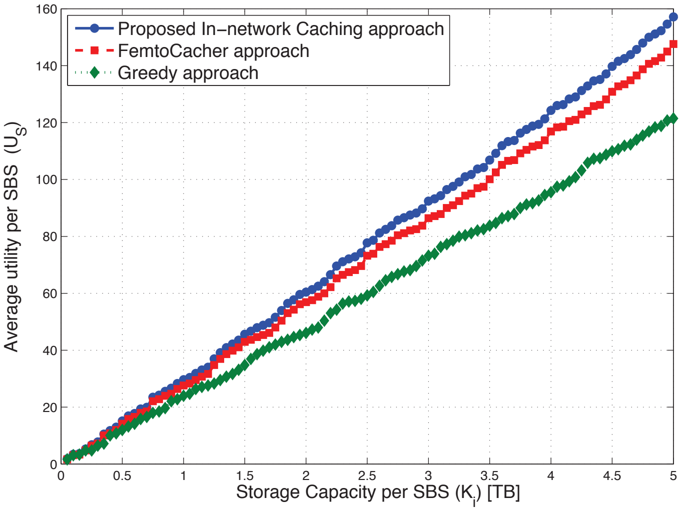

Fig. 3: Average utility U_S per SBS vs storage capacity per SBS K_i. $|\mathcal{L}_i| = 3$ UEs, $S = 3$ SBS, $B = 1$ Gbps.

VI. Performance Evaluation

For our simulations, we considered a single macro-cell of 500×500 m. The macro-cell also represents one network domain $\mathcal{R}$ composed by $|\mathcal{R}| = [0, 200]$ SBSs, uniformly deployed, each one servicing $|\mathcal{L}_i| = [1, 5]$ UEs. The backhaul bandwidth assigned to each UE is $B_{i,m} = 200$ Mbps. For caching purposes, each SBS has a storage capacity chosen from an interval $K_i = [0.1, 5]$ TB. At each SBS, only complete files can be cached, and the size of each file in $\mathcal{F}$ is $s = 10$ MB. The file requests follow a Zipf distribution with parameter $\psi = 1.12$. Each UE m requests $|\mathcal{F}_m| = 100$ files, out of a set of $|\mathcal{F}| = 10^4$ files. The backhaul capacity is equal for all SBSs and it is chosen from an interval $B = [0.1, 1]$ Gbps. In the proposed cost model, the cost of retrieving a missing file from the EPC is $d^M = 10$, while the cost for in-network retrievals is $d^R(\mathcal{S}) = 2 \cdot S$. Prior to the performance evaluation, the SBSs' caches are composed by randomly selected files.

For comparison purposes, we consider two popular schemes from the recent literature. The first is the *FemtoCacher* solution proposed in [12], in which the caching operations are carried out in a cooperative fashion, based on the geographical distribution of the UEs. The second solution is a cooperative greedy approach, in which the SBSs can still access other SBSs' caches within the network domain, but such caches are composed without coordination among the SBSs, by selecting the most popular files (i.e., based on the Zipf distribution).

Figure 2 shows the average utility per SBS as a function of the backhaul bandwidth B, in a network domain with $|\mathcal{R}| = 180$ SBSs, and a storage capacity of $K_i = 5$ TB. Figure 2 shows that, for smaller bandwidth availability ($B < 200$ Mbps) the studied approaches perform similarly, due to the fact that, in the considered simulation setting, most of the backhaul bandwidth is allocated to the UEs' requests, while little is left for cooperative caching (while in-network caching operations need to verify condition (10)). Conversely, for larger backhaul capacities, (i.e., $B > 200$ Mbps), the proposed Algorithm 1 reaps the cooperative gains by balancing instantaneous traffic requests and file caching from in-network SBSs. For example, Figure 2 shows that the performance gains of Algorithm 1 reaches up to 21% and 45% with respect to the FemtoCacher and greedy approaches, respectively, for SBSs with backhaul capacity of $B = 800$ Mbps. Finally, for larger backhaul capacities, the gains stemming from caching tend to saturate since larger cooperative sets $\mathcal{S}$ of SBSs start to form and the cost of retrieving files from the EPC or from other SBSs in $\mathcal{R}$ becomes similar. Therefore, Figure 2 demonstrates that the proposed approach yields significant utility gains by exploiting in-network content availability.

Figure 3 shows the average utility per SBS as a function of the storage capacity K_i, in a network with cooperative groups of maximum size $S = 3$ SBSs, and a backhaul capacity of $B_i = 1$ Mbps. Figure 3 shows the utility U_S is proportional to the probability of finding the UE's files in the serving SBS' cache (i.e., the hit-ratio), which grows for large storage capacities K_i. However, while the Femtocacher only seeks cooperation among the SBSs in the vicinity of the active UEs, Algorithm 1 enables the SBSs to access the cache of any other SBS in the same network domain $\mathcal{R}$. Also note that, unlike Figure 2, the gains stemming from a larger storage capacity do not saturate, for the considered simulation settings. This is due to the fact that the composition of the SBSs' caches is constantly updated. In this respect, while the Femtocacher and the greedy solution systematically drop the least requested files at each SBS, Algorithm 1 drops the least requested files within the cooperative set $\mathcal{S}$, which maintains the cache consistency among cooperative SBSs. For example, the performance gap between the proposed approach and the FemtoCacher solution is 9%, for SBSs with a storage capacity of $K_i = 2$ Mbps and storage units of 3 TB. In summary, Figure 2 and Figure 3 demonstrate that the proposed Algorithm 1 yields additional gains by sharing resources among SBSs and intelligently differentiating the composition of the SBSs' caches, within the same network domain.

Figure 4 shows the average utility per SBS as a function of the file library size $|\mathcal{F}|$, normalized to the storage capacity at each SBS K_i, in a network with $N = 120$ SBSs, and $M = 120$ UEs.

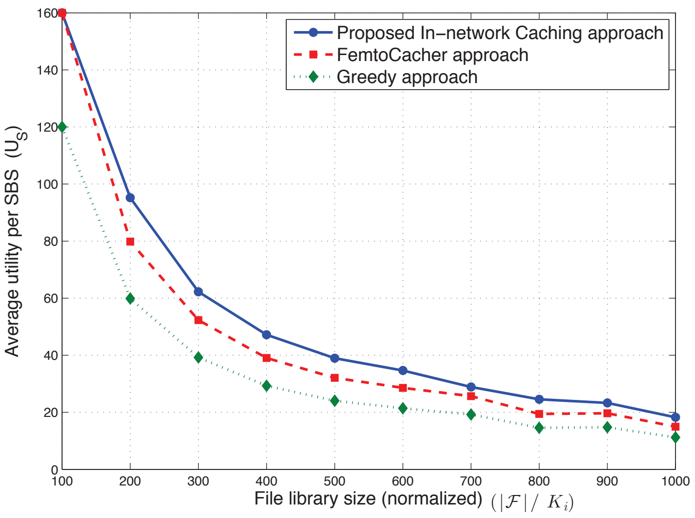

Fig. 4: Average utility U_S per SBS vs file library size (normalized to the caching capacity per SBS). $|\mathcal{L}_i| = 3$ UEs, $B = 0.5$ Mbps, $K_i = 5$ TB.

Figure 4 shows that, for all the considered approaches, the utility U_S decreases for large file libraries $\mathcal{F}$, since U_S directly depends on the hit-ratio. Nevertheless, pooling the caches of in-network SBSs allows to extend set of available files, which yields a maximum performance gain of 23% and 46%, respectively over the Femtocacher and the greedy approach.

In Figure 5, we show the average number of algorithm iterations (Phase II of Algorithm 1) required at each SBS, as a function of the number of SBSs in the network. Figure 5 demonstrates that the complexity of Algorithm 1 depends on the number of SBSs and the UEs they respectively service. For instance, the average number of algorithm iterations is 17, for a set of $\mathcal{L}_i = 2$ UEs per SBS, while it grows up to 20 for a larger set of UEs of $\mathcal{L}_i = 6$ SBSs. In summary, Figure 5 verifies that the complexity of the proposed Algorithm 1 is polynomial and that it converges to a stable solution in a reasonable number of iterations.

VII. Conclusions and Future Work

In this paper, we have presented a novel decentralized solution for in-network caching management and content placement, in small cell networks. We have proposed a cost-model which differentiates between the cases of retrieving files from the EPC and from SBSs connected to the same service gateway. Based on this cost model, we have proposed an algorithm that enables each SBS to select the which contents to cache, by accounting for the composition of the cache at other cooperative SBSs in the same network domain. We have demonstrated that the performance of the proposed algorithm is lower-bounded and evaluated the cooperation gains. Simulation results have shown that, by exploiting local files availability at the SBSs, the proposed cache-based solution enables the SBSs to overcome the limitations of a congested backhaul, and yield significant gains in terms of data delivered to the UEs, reaching up to 23%, with respect to other cooperative caching solutions which only focus on geographical distribution of UEs and SBSs.

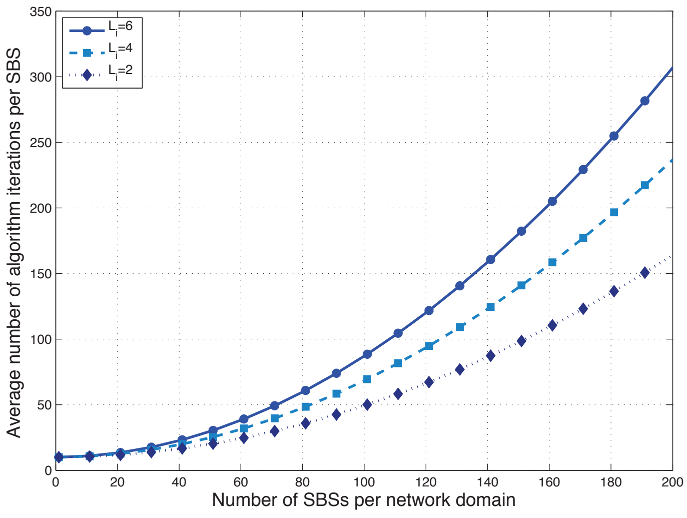

Fig. 5: Average number of algorithm iterations vs number of SBSs in the network domain $|\mathcal{R}|$. $|\mathcal{L}_i| = 2, 4, 6$ UEs, $K_i = 5$ TB, $B = 1$ Gbps.

In this paper, we have assumed static UEs and known popularity distribution of the requested files. Accounting for user mobility will affect the UE-SBS association and the scheduling decisions at each SBS. Similarly, the distribution of file requests can account for the user context (i.e., indoor/outdoor user, using a tablet or a laptop) and the history of past requests. As a result, we plan to further investigate the above aspects and incorporate them into an extended optimization framework.

References

[1] J. Andrews, H. Claussen, M. Dohler, S. Rangan, and M. Reed, "Femtocells: Past, present, and future," *IEEE Journal on Sel. Areas in Comm.*, vol. 30, no. 3, pp. 497 –508, Apr. 2012.

[2] T. Q. S. Quek, G. de la Roche, I. Guvenc, and M. Kountouris, *Small Cell Networks: Deployment, PHY Techniques, and Resource Management*. New York, USA: Cambridge University Press, Sept. 2012.

[3] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5G systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, February 2014.

[4] J. Erman, A. Gerber, M. Hajiaghayi, D. Pei, S. Sen, and O. Spatscheck, "To cache or not to cache: The 3G case," *IEEE Internet Computing*, vol. 15, no. 2, pp. 27–34, March 2011.

[5] F. Pantisano, M. Bennis, W. Saad, and M. Debbah, "Cache-aware user association in backhaul-constrained small cell networks," in *In Proc. of Int'l Symp. of Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2014, pp. 37–42.

[6] E. Bastug, J.-L. Guenego, and M. Debbah, "Proactive small cell networks," in *In Proc. of Int'l Conf. on Telecommunications (ICT)*, May 2013, pp. 1–5.

[7] E. Bastug, M. Bennis, and M. Debbah, "Social and spatial proactive caching for mobile data offloading," in *In Proc. of IEEE ICC Workshop on Small Cell and 5G Networks (SMALLNETS)*, June 2014, p. (to appear).

[8] K. Hamodouche, W. Saad, and M. Debbah, "Many-to-many matching games for proactive social caching in small cell networks," in *In Proc. of 12th Int'l Symp. on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2014, pp. 1–5.

[9] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, Aug 2014.

[10] 3GPP, "Local IP access and selected IP traffic offload (LIPA- SIPTO)," *3GPP Technical Specification (TS 23.829)*, Mar. 2011.

[11] N. Golrezaei, A. Dimakis, and A. Molisch, "Wireless device-to-device communications with distributed caching," in *In Proc. of IEEE Int'l Symp. on Information Theory Proceedings (ISIT)*, July 2012, pp. 2781–2785.

[12] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *In Proc. of IEEE INFOCOM*, March 2012, pp. 1107–1115.