Caching as a Service: Small-Cell Caching Mechanism Design for Service Providers

Zhiwen Hu, *Student Member, IEEE*, Zijie Zheng, *Student Member, IEEE*, Tao Wang, *Senior Member, IEEE*, Lingyang Song, *Senior Member, IEEE*, and Xiaoming Li

***Abstract*— Wireless network virtualization has been well recognized as a way to improve the flexibility of wireless networks by decoupling the functionality of the system and implementing infrastructure and spectrum as services. Recent studies have shown that caching provides a better performance to serve the content requests from mobile users. In this paper, we propose that caching can be applied as a service in mobile networks, i.e., different service providers (SPs) cache their contents in the storage of wireless facilities that are owned by mobile network operators. Specifically, we focus on the scenario of small-cell networks, where cache-enabled small-cell base stations are the facilities to cache contents. To deal with the competition for storage among multiple SPs, we design a mechanism based on multi-object auctions, where the time-dependent feature of system parameters and the frequency of content replacement are both taken into account. Simulation results show that our solution leads to a satisfactory outcome.**

***Index Terms*— Wireless network virtualization, small-cell caching, multi-object auction, matching.**

I. Introduction

WIRELESS network virtualization has been proposed in recent years to improve the flexibility of traditional wireless networks against the tremendous growth of diversified online services [2]. Similar to the traditional wired network virtualization [3], wireless network virtualization separates wireless networks into physical infrastructures and online services [4]. In wireless networks, the parties that operate the wireless physical infrastructures are called mobile network operators (MNOs), and the parties that provide online services for users are called service providers (SPs). SPs can typically create their own virtual networks to serve their users by aggregating resources from MNOs, where the *resources* usually have a broad scope, ranging from the spectrum, the infrastructure, to the air interface [5]. With the help of virtualization, multiple heterogeneous virtual networks that dynamically composed by different SPs can coexist together in isolation from each other [4]. Therefore, once the system is properly designed, wireless network virtualization can maximize the system utilization, facilitate the updating of existed services and alleviate the difficulty of applying new ones [6].

Since the services provided by SPs depend on the resources that allocated to them, resource allocation becomes one of the important issues, i.e., how to effectively allocate the limited resources to different SPs [2]. In most early studies, spectrum was considered as the most basic kind of resource in wireless network virtualization. The authors in [7] and [8] discussed the spectrum allocation problem in both time domain and frequency domain, and the works in [9] and [10] dealt with the competition for spectrum among SPs by using game theory. Apart from the spectrum, another kind of important resource that being considered in previous works was the infrastructure, such as the wireless building premises, RF antennas, and network routers, etc [2]. Several studies showed the ongoing trends of the virtualization of wireless infrastructures [11], [12]. Moreover, the combination of spectrum and infrastructure sharing was proposed as *full network sharing*, which was detailedly classified in [13].

However, there are still potential resources that are not discussed in wireless network virtualization, such as the storages of wireless facilities [14]. The storage-enabled wireless facilities were proposed in [15]–[22], where the pre-cached contents in small-cell base stations (SBSs) can bring better system performance, showing an effective way to deal with the low-speed backhaul link of SBSs [23]. This proposal was first given in [15], where a sub-optimal strategy of caching content was provided. Based on this, authors of [16]–[18] considered a more detailed physical layer model. Other studies in [19] and [20] further discussed the network layer coding technique. Moreover, the works in [21] and [22] also took into account the social ties and the mobilities of users respectively.

Although small-cell caching has been discussed from many aspects, few existing studies focus on the decision layer, where multiple SPs may exist. Since all the SPs only intend to better serve their own users by caching their own contents to reduce the average delay, they are likely to compete for limited caching storages. Thus, a proper mechanism should be designed to deal with the competition among SPs and

Manuscript received June 20, 2015; revised November 10, 2015 and March 17, 2016; accepted July 16, 2016. Date of publication July 27, 2016; date of current version October 7, 2016. This work was supported in part by the National 973 project under Grant 2013CB336700 and in part by the National Nature Science Foundation of China under Grant U1301255 and Grant 61370056. This paper was presented at the IEEE International Conference on Computer Communications, Hong Kong, April 2015 [1]. The associate editor coordinating the review of this paper and approving it for publication was M. Li.

The authors are with the School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China (e-mail: zhiwen.hu@pku.edu.cn; zijie.zheng@pku.edu.cn; wangtao@pku.edu.cn; lingyang.song@pku.edu.cn; lxm@pku.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2016.2594199

guarantee the overall performance at the same time. To solve the aforementioned problem, we propose to design an effective caching mechanism, which enables *caching as a service* in wireless networks. Specifically, the storages of wireless facilities can be virtualized and shared among multiple SPs, and these SPs can utilize the storages as caching spaces to cache their own contents for their users. With the help of caching, the average delay of content requests from users can be lowered, such that the quality-of-service can be improved.

Without loss of generality, in this paper, we focus on a scenario where small-cell base stations are the facilities that used to cache contents [15]. We formulate the multiple SPs' small-cell caching problem by taking into account the overlapping among SBSs. Since SPs have to compete for the caching storages on behalf of their own contents, a nature solution is to apply auctions [24], where each SP has to evaluate its contents and bid for caching storages. We propose our own mechanism based on multi-object auctions, where the mechanism organizes a serial of multi-object auctions to complete the caching scheme. Each multi-object auction can be solved by the market matching algorithm [25], which takes valuations as input and takes allocation results and prices as output. Considering that the system parameters are time-dependent, storages of SBSs may change contents to adapt to the variation, which also burdens the backhaul link of SBSs. To cope with this problem, we also present a novel approach to reduce the frequency of content replacement. Simulation results have shown the effectiveness of our solution.

The main contributions of our work are listed below:

1) We come up with a novel approach that caching can be applied as a service in the mobile networks with the help of wireless network virtualization, where each SP has to pay for the storages of the infrastructure that owned by MNOs.
2) We focus on the small-cell caching scenario and formulate the caching problem with the objective to minimize average delay, where the overlapping among SBSs and the competition among SPs are considered.
3) By using multi-object auctions in our mechanism, we provide a sub-optimal solution and also find a way to reduce the frequency of content replacement between adjacent hours.

The rest of our paper is organized as follows. Section II presents our system model of small-cell caching. Section III provides the problem formulation and the theoretical analysis on the system parameters. Section IV introduces our auction mechanism. Section V shows the simulation results which prove the effectiveness of our solution and testify our theoretical analysis. Finally, we conclude our paper in Section VI.

II. System Model

In this paper, we study a small-cell network, which involves I SBSs in an area and L SPs that provide different contents for users, as shown in Fig. 1. We use SBS_i to denote the i^{th} SBS, and SP_l to denote the l^{th} service provider, where $1 \leq i \leq I$ and $1 \leq l \leq L$. These SPs intend to cache their own contents into SBSs, where the storage capacity of SBS_i is given by H_i.

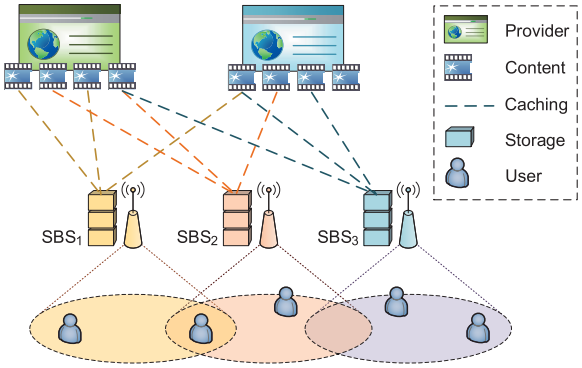


Fig. 1. System model for small-cell caching among multiple providers. For each request from any user, the delay can be lowered as long as the requested content is cached in a nearby SBS.

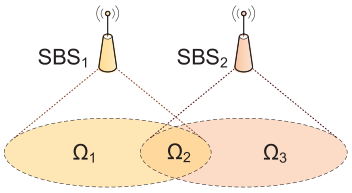


Fig. 2. A demonstration of coverage regions of two overlapping SBSs, where three simplest regions are included.

For the rest part of this section, we model our system in four aspects: the coverage region of SBSs, the distribution of users, the contents of SPs, and the traffic latency of content requests.

A. Coverage Regions

We assume that the SBSs are distributed in a 2-dimensional area, and the coverage regions of SBSs overlap with each other, as shown in Fig. 2. We define a region as a *simplest region* if it is not crossed by any curves in such a figure. In our model, simplest regions are denoted by Ω_j, $1 \leq j \leq J$, where J is the total number of simplest regions. Since Ω_j is covered by a set of SBSs, we use $\mathcal{F}_j$ to denote the set of SBSs that cover Ω_j. For the example shown in Fig. 2, we have $\mathcal{F}_1 = \{1\}$, $\mathcal{F}_2 = \{1, 2\}$, $\mathcal{F}_3 = \{2\}$.

B. Users' Distribution

The distribution of mobile users is described by user density, which can be space-dependent as well as time-dependent. And it can be estimated by some statistical methods [26] with the help of the data collected by SBSs. In order to better reflect the time-dependent characteristics, we use t to represent a specific time slot, where $t \geq 1$. And without loss of generality, we assume the length of a time slot is an hour.[1] We use $u^t(\boldsymbol{x})$ to denote the average user density at the t^{th} hour at location $\boldsymbol{x}$, where $\boldsymbol{x}$ is a two dimensional vector in space.

[1]One hour's length is a reasonable choice for caching replacement, because one hour can be precise enough to describe the overall variation of user density and content popularity. Although shorter time slot might be a better choice, the key point of our paper is not to choose the best time slot but to solve the problem with a given length of time slot.

Thus, the average user number at the t^{th} hour in the region Ω_j can be given by

$$U_j^t = \iint_{\Omega_j} u^t(\boldsymbol{x}) d\boldsymbol{x}. \tag{1}$$

The average user number at the t^{th} hour under SBS_i is

$$U^{t,i} = \sum_{j|i\in\mathcal{F}_j} \iint_{\Omega_j} u^t(\boldsymbol{x}) d\boldsymbol{x}. \tag{2}$$

And similarly, the total user number at the t^{th} hour can be calculated as:

$$U_{sum}^t = \sum_{j=1}^{J} \iint_{\Omega_j} u^t(\boldsymbol{x}) d\boldsymbol{x}. \tag{3}$$

C. Contents of SPs

We assume that SPs possess different sets of contents, and the contents may have different sizes. The k^{th} content of SP_l is denoted by $C_{l,k}$, and the size of $C_{l,k}$ is denoted by $S_{l,k}$, where $1 \le k \le K_l$ and K_l is the number of contents possessed by SP_l. At the t^{th} hour, the possibility of $C_{l,k}$ being requested by each single user is described by its *popularity*, denoted by $\phi_{l,k}^t$. And we also have

$$\phi_{sum}^t = \sum_{l,k} \phi_{l,k}^t, \tag{4}$$

where ϕ_{sum}^t is not necessary to be normalized to one, since each user can request several contents in an hour. A greater ϕ_{sum}^t implies more requests from users in an hour. Note that the trend of the variation of content popularity can also be predicted with some learning mechanisms [27].

Since contents can be divided into sequential blocks for caching, we use $C_{l,k,n}$ to denote the n^{th} block of $C_{l,k}$, and $S_{l,k,n}$ to denote the size of it, where $1 \le n \le N_{l,k}$, and $N_{l,k}$ is the number of blocks that $C_{l,k}$ is divided. Note that this content division procedure does not change users' requesting probability, therefore, all the blocks from the same original content $C_{l,k}$ have the same possibility to be requested, i.e., they share the same popularity $\phi_{l,k}^t$. Also notice that if one block is being requested, other blocks from the same content are also being requested at the same time, thus their requesting possibility is inter-dependent.

D. Traffic Latency

If the content requested by a certain user is cached in one of his nearby SBSs, then the request can be served by this SBS, which leads to a lower delay. Otherwise, one of the nearby SBSs can serve the user by setting up backhaul connections to the core network and downloading the content from the server. Therefore, the delay model of backhaul-link (from SBSs to SPs) and the delay model of downlink (from SBSs to users) should be constructed.

Here, we assume that the delay of backhaul-link θ_{back}^t is proportional to U_{sum}^t, since the load of both the backhaul network and the core network mainly depends on the total number of connected users. And for SBS_i, we assume that the delay of its downlink $\theta_{down}^{t,i}$ is proportional to $U^{t,i}$, i.e., the number of user that SBS_i covers.[2] So we have

$$\theta_{back}^t = \beta_1 \cdot U_{sum}^t, \tag{5}$$

$$\theta_{down}^{t,i} = \beta_2 \cdot U^{t,i}. \tag{6}$$

For a specific user that covered by several SBSs, it will choose a SBS with the lowest delay to download its desired content (since the content may only be cached in a few of these SBSs). In the "choosing" procedure, additional delay may be induced, and we assume this kind of delay depends on the number of available SBSs for the user. And we give the following definition:

$$\theta_c^j = \beta_3 \cdot |\mathcal{F}_j|, \tag{7}$$

where $|\mathcal{F}_j|$ is the number of SBSs by which Ω_j is covered. The more SBSs cover a user, the more time will be spent to choose the best downloading SBS. And we call θ_c^t as "choosing delay" later in our paper.

III. Problem Formulation and Analysis

In this section, we first formulate the problem and provide the objective function, then analyse the impact of three system parameters, which are the total number of the contents, the average storage capacity of SBSs, and the overlapping percentage.

A. Problem Formulation

We first use Γ to denote the allocation matrix, the definition of its elements is given below:

$$\gamma_{l,k,n}^{t,i} = \begin{cases} 1, & \text{if } C_{l,k,n} \text{ is cached in } \text{SBS}_i \text{ at } t, \\ 0, & \text{if } C_{l,k,n} \text{ is not cached in } \text{SBS}_i \text{ at } t. \end{cases} \tag{8}$$

For a user in the region Ω_j, if he requests $C_{l,k}$ at t, the delay can be calculated as:

$$\theta_{l,k}^{t,j} = \sum_{n=1}^{N_{l,k}} \frac{S_{l,k,n}}{S_{l,k}} \min_{i|i\in\mathcal{F}_j} \left[\theta_{down}^{t,i} + (1-\gamma_{l,k,n}^{t,i})\theta_{back}^t\right] + \theta_c^j, \tag{9}$$

where $\frac{S_{l,k,n}}{S_{l,k}}$ is the weight of the n^{th} block of $C_{l,k}$, and the average delay of requesting a specific content should be the weighted summation of the delay of requesting its blocks (which conforms to user experience).

Our main objective is to minimize the average delay of content requests from users at each hour by properly designing the allocation of caching storages. Based on (1), (3), (4) and (9), the average delay at the t^{th} hour can be written as:

$$D(t) = \frac{1}{\phi_{sum}^t U_{sum}^t} \sum_{l,k,j} \theta_{l,k}^{t,j} \cdot U_j^t \cdot \phi_{l,k}^t. \tag{10}$$

[2]Based on the result of [28], the transmission delay is mainly influenced by the number of connected users, and this relationship can be approximately regarded as a linear one. On the other hand, a specific user in the overlapping area of two SBSs can contribute incremental delay to both SBSs, no matter which SBS it is connected to. This is because SBSs may use the same bandwidth and this user takes up a specific channel of both SBSs. Therefore, $\theta_{down}^{t,i} = \beta_2 \cdot U^{t,i}$ is a reasonable formulation.

Finally, we give the objective function and its constraint as:

$$\min_{\Gamma} D(t), \quad \forall t, \tag{11}$$

$$s.t. \sum_{l,k,n} \gamma_{l,k,n}^{t,i} \cdot S_{l,k,n} \leq H_i, \quad \forall i, \ \forall t. \tag{12}$$

This problem is hard to optimize, even a much simpler version of this problem given in [15] is also proved to be NP-hard by reducing to a k-Disjoint Set Cover Problem [29]. The mechanism given in Section III is a sub-optimal solution based on optimizing a sequence of sub-problems.

B. Theoretical Analysis of System Parameters

In this subsection, we analyse some of the parameters which can affect the performance of the system at each certain hour. Since the competition for limited caching storages among SPs is the core issue, the total number of the contents to be cached and the storage capacity of SBSs are the two most concerns. Besides, the degree of overlapping among SBSs can also affect the outcome, which was never quantitatively discussed in early works. Therefore, we analyse three parameters here: the total number of the contents, denoted by K; the average storage capacity of SBSs, denoted by H; and the overlapping percentage, denoted by O. We define them as:

$$K = \sum_{l=1}^{L} K_l, \tag{13}$$

$$H = \sum_{i=1}^{I} H_i \Big/ I, \tag{14}$$

$$O = \Big[\sum_{i=1}^{I} A_i - A_{total} \Big] \Big/ A_{total}, \tag{15}$$

where A_i is the area of the coverage region of SBS_i, and A_{total} is the total area of SBSs' coverage regions. Due to the overlap of SBSs, we have $\sum_{i=1}^{I} A_i \geq A_{total}$, which means $O \geq 0$. The system performance is mainly reflected and measured by the average delay given in (10). Here, we provide three propositions on the influence of these parameters and proof them respectively.

Proposition 1: With a certain distribution of user density and content popularity, the total number of the contents K has a positive correlation with the average delay $D(t)$.

Proof: Suppose that there are initially K contents in the system and we denote the set of these content as $\mathcal{C}$. The average delay can be calculated after allocation, and we have

$$D(t) = \frac{1}{\phi_{sum}^{t} U_{sum}^{t}} \sum_{l,k,j} \theta_{l,k}^{t,j} U_j^t \phi_{l,k}^t = \frac{1}{\phi_{sum}^{t} U_{sum}^{t}} D(t, \mathcal{C}), \tag{16}$$

where $D(t, \mathcal{C})$ represents the un-normalized total delay of requesting contents in $\mathcal{C}$.

When additional set of contents $\mathcal{C}'$ with the same popularity distribution is added, supposing $|\mathcal{C}'| = K' = xK$ and $x > 0$, we can provide $\phi_{sum}^{t\prime} = (1+x)\phi_{sum}^{t}$, because the distribution of content popularity are fixed. The new caching result makes the average delay change in the form as below:

$$D'(t) = \frac{1}{\phi_{sum}^{t\prime} U_{sum}^{t}} \cdot \big[D'(t, \mathcal{C}) + D'(t, \mathcal{C}') \big] \tag{17}$$

where the $D'(t, \mathcal{C})$ represents the un-normalized total delay of requesting original contents, and $D'(t, \mathcal{C}')$ represents the un-normalized total delay of requesting newly added contents.

Due to the competition brought by additional contents, some of the original contents are evicted from the caching storage, which leads to $D(t, \mathcal{C}) < D'(t, \mathcal{C})$. And due to the same popularity distribution of $\mathcal{C}$ and $\mathcal{C}'$, the proportion that contents from $\mathcal{C}$ are cached and the proportion that contents from $\mathcal{C}'$ are cached are similar. Since $D(t, \mathcal{C})$ is the un-normalized delay, we have

$$D'(t, \mathcal{C}') : D'(t, \mathcal{C}) = |\mathcal{C}'| : |\mathcal{C}| = x. \tag{18}$$

Based on the expressions above, we can deduce that

$$\begin{aligned} D'(t) &= \frac{1}{\phi_{sum}^{t\prime} U_{sum}^{t}} \cdot \big[D'(t, \mathcal{C}) + D'(t, \mathcal{C}') \big] \\ &= \frac{1}{\phi_{sum}^{t\prime} U_{sum}^{t}} \cdot (1+x) \cdot D'(t, \mathcal{C}) \\ &> \frac{1}{\phi_{sum}^{t\prime} U_{sum}^{t}} \cdot (1+x) \cdot D(t, \mathcal{C}) \\ &= \frac{1}{\phi_{sum}^{t} U_{sum}^{t}} \cdot D(t, \mathcal{C}) = D(t). \end{aligned}$$

This result can also be intuitively comprehended that the increase in content number leads to the decrease in caching percentage, making the caching system less efficient. ■

Proposition 2: With fixed distributions of users, SBSs and content popularity, supposing that the storage capacities of SBSs are the same, then average storage capacity H has a negative correlation with average delay $D(t)$.

Proof: Assume that the average delay with storage capacity H is $D(t)$, and the average delay with storage capacity H' is $D'(t)$, where $H' > H$. Since the caching result of H' can be derived from the given caching result of H, additional contents can be added to the caching storages, which directly makes $\theta_{l,k}^{t,j\prime} \leq \theta_{l,k}^{t,j}$. Note that at least one set of l, k leads to $\theta_{l,k}^{t,j\prime} < \theta_{l,k}^{t,j}$, thus we have $D'(t) < D(t)$. ■

Unlike the analysis on number of contents or storage capacity, the influence of overlapping is abstruse due to the complicated geographic distribution of SBSs. We have to first assume that SBSs with fixed coverage radius are uniformly distributed in a cellular grid, where we control the overlapping percentage by making the cellular grid denser or sparser. An illustration is shown in Fig. 3. To further simplify the problem to be analyzed, we only consider a special case where the parameter β_3 in the equation (7) equals to zero, i.e., the choosing delay is ignored.

Proposition 3: In an approximately infinite cellular grid where SBSs with fixed coverage radius are uniformly distributed, given the constraint that 1) users density is uniform, 2) no coverage regions of four or more SBSs exist, and 3) the choosing delay can be ignored, the average delay based

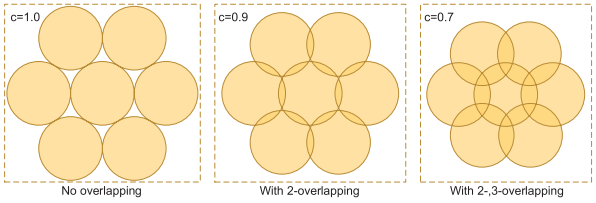


Fig. 3. A top view of the overlapping regions of SBSs with radius R. The distance of two adjacent SBSs is $2Rc$ where c is the compress factor. We let $c \in [1/\sqrt{3}, 1]$ to make sure that overlapping regions of four or more SBSs don't exist.

on a fixed caching result decreases when the overlapping percentage increases.

Proof: The detailed proof of this proposition is given in the Appendix. ■

These three propositions can be verified by our simulation results provided in section V. From the theoretical analysis above, we can have a rough idea of how well the caching can benefit the system under different circumstances. Although the total number of contents are not under control in a real world, we can still achieve a better result by enlarging the storage capacity of SBSs. Besides, the performance can be improved if the overlapping percentage of SBSs is greater, as long as the choosing delay is ignorable.

IV. Auction Mechanism Design

In this section, we propose an auction-based mechanism to solve the caching problem. In this mechanism, the caching scheme for each hour is determined by a series of *multi-object auctions*, where the objects are the storages of SBSs that owned by MNOs, and the bidders are the SPs who possess different sets of contents. Since the contents have different sizes, it is difficult to apply auctions directly based on the original contents. Therefore, MNOs should announce a *standard caching size* S in the auctions. With this standard, the storages of SBSs are divided into multiple blocks with size of S, and the contents of each SP are transformed into S-sized independent content blocks. In addition, we also propose *additional prices* are charged to properly reduce the frequency of content replacement between hours.

For the rest of this section, we first provide the setup of multi-object auctions at each hour, then introduce the market matching algorithm for each auction, and finally we discuss some properties of our mechanism. The whole procedure of our mechanism is shown in Algorithm 1.

A. Multi-Object Auction Setup

In this subsection, we first provide a method of transforming contents into equal-sized content blocks, then demonstrate the auctions at each hour and the valuations without additional prices, and finally take addition prices into account and provide the final valuations.

1) Transforming Contents Into Equal-Sized Content Blocks: Since the caching problem we've formulated is similar to the classical knapsack problem [30], the transforming procedure that we propose is inspired by one of the greedy algorithm.

Algorithm 1 The Proposed Auction-Based Caching Mechanism

```
begin
    MNOs announce the standard caching size S;
    Auctions setup at t = 1;
    while true do
        Each SP transforms its own contents to form
        multiple S-sized content blocks;
        for j from 1 to max_i{H_i/S} do
            The j^th storage blocks in all SBSs are regarded
            as objects;
            SPs estimate the utility of caching each of their
            content blocks to each of SBSs;
            Create the valuation matrix based on current
            allocations and additional prices;
            Run the market matching algorithm to complete
            one single multi-object auction;
        end
        Let t = t + 1, continue to determine the caching
        result in the next hour;
    end
end
```

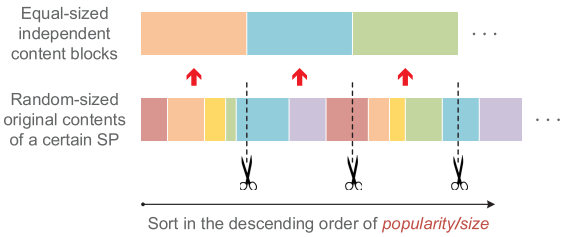


Fig. 4. The method of transforming random-sized original contents of a certain SP into equal-sized independent content blocks.

For a given SP, we sort all of its contents in the descending order of *popularity to size ratio*, and put them together to form a one-dimensional long "data ribbon", as shown in Fig. 4. And based on the given standard caching size S, we cut this "data ribbon" from the left side into multiple S-sized content blocks. Here we ignore the minor problem that whether the length of the "data ribbon" can be divisible by S, since the most right side usually consists of low-popularity contents and they have little impact to the caching performance.

Due to the huge number of contents in reality, we recommend that S is set greater than the largest original content, in which way the computational complexity can be reduced to some extent. And as a result, each content is divided into no more than two content blocks. Notice that the newly formed content blocks are independent from each other, which means that no two content blocks share a common slice of data.

Here, we use $B_{l,r}^t$ to denote the r^{th} content block of SP_l at the t^{th} hour. And the popularity of $B_{l,r}^t$ can be calculated as $\varphi_{l,r}^t = \sum_k \eta_{l,r,k}^t \cdot \phi_{l,k}^t$, where $0 \le \eta_{l,r,k}^t \le 1$, representing the percentage that the original content $C_{l,k}$ is contained in the content block $B_{l,r}^t$. This equation is essentially to linearly

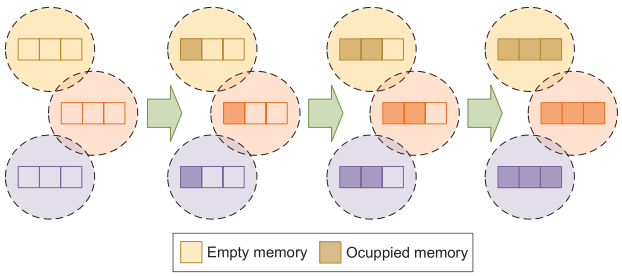


Fig. 5. The mechanism for each hour's caching, where a dashed circle indicates the coverage region of a SBS.

add up the weighted popularity of the contained contents, which conforms to the linear additive formulation given in equation (9).

In the multi-object auctions, we only consider the content blocks as the whole caching objects, and we use $\gamma_{l,r}^{t,i}$ to represent the allocation matrix of content blocks. Note that this denotation differs from $\gamma_{l,k,n}^{t,i}$ in equation (8), which stands for the allocation of original contents.

2) A Serial of Multi-Object Auctions for Each Hour: The caching problem for each hour is solved by holding a serial of multi-object auctions. Specifically, we auction for $\max\{\frac{H_i}{S}\}$ times, where the j^{th} memory block in all the SBSs are auctioned off in the j^{th} auction. This process is essentially to auction the storages of all SBSs concurrently with multiple steps, as shown in Fig. 5. In each multi-object auction, SPs play the roles of bidders and storages of SBSs play the roles of objects. After the each auction, each SP obtains a certain amount of caching spaces in each SBSs. Then each SP can place its contents into SBSs according to the caching result (can be done automatically by its server).

However, before each auction, SPs have to estimate the utility of caching each of their content blocks to each of the SBSs and bid for them. Based on equations (9) and (10), we give the expression to calculate the marginal utility of caching each content block into each SBS during the auction procedure as:

$$V_{l,r}^{t,i} = \sum_{j|i\in A_j} -\Delta\theta_{l,r}^{t,j} \cdot U_j^t \cdot \varphi_{l,r}^t, \tag{19}$$

where $-\Delta\theta_{l,r}^{t,j}$ is the average decrease of latency for the users in Ω_j requesting contents in $B_{l,r}^t$.

3) Additional Prices for Content Replacement Between Hours: Since content popularity and user density are time variant, caching results in different hours may differ a lot. When a SBS changes its caching contents, additional instantaneous traffic load burdens the backhaul. Regardless the specific technique to replace or pre-cache contents, we define λ^t as the *replacement percentage* at the t^{th} hour to indicate this additional load, given by

$$\lambda^t = \frac{1}{I}\sum_{i=1}^{I}\frac{1}{B_i}\sum_{l,r} S\cdot(1-\varepsilon_{l,r}^{t,i})\cdot\gamma_{l,r}^{t,i}, \tag{20}$$

where $\varepsilon_{l,r}^{t,i}$ represents the percentage of data in $B_{l,r}^t$ that was cached in SBS_i at $t-1$. Therefore, λ^t shows the average percentage of the storages of SBSs that are replaced.

In order to reduce λ^t, additional prices are charged for replacing the original contents if the new ones weren't cached in this SBS in the last hour. Here, we design the additional price as $\Delta p_{l,r}^{t,i} = \omega\cdot(1-\varepsilon_{l,r}^{t,i})\cdot\theta_{back}^t$, which indicates heavier traffic needs higher additional prices to limit the replacement percentage. And the constant ω is defined as *additional price coefficient*.

The introduction of additional price results in an adjustment to the valuations given above. For $B_{l,r}^t$, the valuation is calculated by $V_{l,r}^{t,i} - \Delta p_{l,r}^{t,i}$. This is because once $B_{l,r}^t$ obtains the caching storage in SBS_i, it will further pay $\Delta p_{l,r}^{t,i}$ because of additional price.

To simplify the denotations later in this section, we use C_n to represent the n^{th} content block (among all the content blocks from all the SPs), where $1 \le n \le N$, and use D_m to represent the m^{th} storage block (i.e., the storage block that provided by SBS_m), where $1 \le m \le M$. The valuation of C_n for D_m is denoted by $v_{n,m}$, which can be calculated by the corresponding expression of $V_{l,r}^{t,i}$ and $\Delta p_{l,r}^{t,i}$, given by $v_{n,m} = V_{l,r}^{t,i} - \Delta p_{l,r}^{t,i}$.

B. Market Matching Algorithm

To solve a single multi-object auction, we provide the *market matching algorithm*, which is originated from [25] and is able to match the content blocks and the storage blocks with maximum total utility.[3] For writing simplicity, we use the word "content" instead of "content block" and use "storage" instead of "storage block" in the rest of this subsection.

To be brief, this algorithm takes the valuations as the input, uses bipartite graph to get a perfect matching between contents and storages,[4] and outputs the allocation results and the prices of storages. This algorithm can be described by 7 steps as follows. Step 1 and Step 2 introduce the initialization process, Step 3 to Step 6 provide the iteration process to find a perfect matching between contents and storages, and Step 7 provides the outcome of the algorithm.

Step 1: Given N contents and M storages ($N > M$), add $N - M$ virtual storages.

This step is to equalize the number of contents and the number of storages, which is a necessary condition for the following steps. Due to the unworthiness of virtual storages, the valuations of them are confined as zero and the contents that obtains a virtual objects actually obtains nothing. And for clearer writing, we assume $N = M$ later in the algorithm introduction.

[3]The original algorithm satisfies the VCG principle [31], where bidders' best strategy is to bid truthfully. However, here we regard each content as the corresponding bidder. Since different contents may belong to the same SP, "truth-telling" cannot be proved to be the best strategy. But we are able to prove that: 1) If a SP doesn't know the valuations of others, then the average utility is excepted to decrease when it randomly varies its bids from the true valuations. 2) If a SP knows the valuations of others, the only better strategy is to decrease some of its bids, with the constraint of not affecting the auction results. Anyway, the caching efficiency will not be degraded, and it's reasonable to adopt "truthful bidding" in our mechanism.

[4]Since the algorithm is based on matching, one content cannot get more than one storage in each auction. Therefore a problem may arise if $I > \min\{H_i\}$, i.e., a content with great popularity is unable to be cached in every SBSs. Hence, we assume $I \le \min\{H_i\}$, which can be satisfied in most real-world situations because the storages are usually large enough.

Step 2: Initialize the prices of all storages as zero, i.e., $p_m = 0$, $\forall m \in [1, M]$.

The price of a storage represents the money that has to be paid by the SP whose content obtains this storage. And these prices will gradually increase in the process of the algorithm.

Definition 1: In a bipartite graph where a set of content nodes $\mathcal{C}$ is connected to a set of storage nodes $\mathcal{D}$, the edge between C_n and D_m exists if and only if $(v_{n,m} - p_m)$ is the largest for any m with a fixed n, then the bipartite graph is called a preferred-storage graph.

Step 3: Based on the valuations and the prices, a preferred-storage graph can be built. To put it simple, the preferred-storage graph shows which are the most preferred storages of each content. For C_n, if the profit of acquiring D_m is highest, then there will be an edge between C_n and D_m. Note that for each content, there may be serval most preferred storages.

Definition 2: Given a graph $G = (V, E)$, a matching *is a subset edges of E such that no two edges in this subset share a same vertex.*

Definition 3: Given a graph and a matching of it, an alternating path *is a serial of consecutively connected edges such that these edges are alternately contained or not contained in the matching. And an alternating path is an* augmenting path *if and only if the two end-vertices in the alternating path are unmatched.*

Step 4: In the preferred-storage graph, we use augmenting paths to expand the matching until no augmenting paths can be found.

The classical breadth-first-search (BFS) algorithm [32] is applied to find augmenting paths started from any of an unmatched content node. Given a matching $\mathcal{M}$ and a certain augmenting path, we denoted all the edges in the augmenting path as $\mathcal{E}$. The edges both in $\mathcal{E}$ and $\mathcal{M}$ are denoted as $\mathcal{E}_1$, and the edges in $\mathcal{E}$ but not in $\mathcal{M}$ are denoted as $\mathcal{E}_2$. By adding $\mathcal{E}_2$ to the matching and deleting $\mathcal{E}_1$ from the matching, a greater matching can be formed, because $|\mathcal{E}_2| = |\mathcal{E}_1| + 1$. The matching achieves maximum when there are no augmenting paths can be found.

Step 5: Based on the matching in the last step, if all the nodes are matched, i.e., the maximum matching is a perfect matching, then jump to the step 7. Otherwise, a constricted set can be found, which forbids us to get a perfect matching. The constricted set is defined as below:

Definition 4: In a preferred-storage graph, given $\mathcal{C}'$ as a subset of the content nodes, denote the directly connected storage nodes as set $\mathcal{D}'$. If $|\mathcal{C}'| > |\mathcal{D}'|$, then $\{\mathcal{C}', \mathcal{D}'\}$ forms a constricted set.

Intuitively, the constricted set cannot form a perfect matching in itself because the number of storages is larger. Therefore, the whole bipartite graph fails to form a perfect matching if a constricted set is contained. Reference [33] shows that the equivalence condition of a bipartite graph having a perfect matching is that there are no constricted sets. The process of searching for a constricted set is simple. When the algorithm fails to find an augmenting path during BFS, the nodes that being visited by BFS automatically form a constricted set [33]. Two examples of perfect matching and constricted set are shown in Fig. 6.

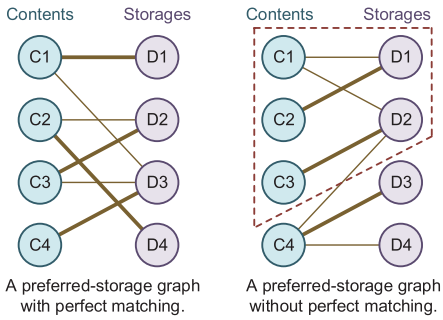


Fig. 6. Two preferred-storage graphs. The left one has a perfect matching, while the right one is confined by a constricted set shown in the dashed box.

Algorithm 2 Market Matching Algorithm for Each Multi-Object Auction

Input: Valuation matrix $V_{N\times M}$ (N: contents, M: storages, $N > M$).
Output: Allocation matrix $\Gamma_{N\times M}$ and price vector P_M.
begin
  Expand the valuation matrix from $V_{N\times M}$ to $V_{N\times N}$ with zeros (add virtual storages);
  Initialize the price vector P_N as zero;
  while *true* **do**
    Build a preferred-storage graph $G(\mathcal{C}, \mathcal{D}, \mathcal{E})$ based on $V_{N\times N}$ and P_N;
    Find a maximum matching M in this preferred-storage graph;
    if *M is a perfect matching* **then**
      Break the while loop;
    else
      Find a constricted set $(\mathcal{C}', \mathcal{D}')$ in $G(\mathcal{C}, \mathcal{D}, \mathcal{E})$;
      Find the minimum price δp, which can change G once added to storages in $\mathcal{D}'$;
      Let $p_n = p_n + \delta p$ for all $D_n \in \mathcal{D}'$;
      Let $p_n = p_n - \min\{p_n\}$ for all $1 \le n \le N$;
    end
  end
  The matrix $\Gamma_{N\times M}$ shows the caching result;
end

Step 6: Once a constricted set $\{\mathcal{C}', \mathcal{D}'\}$ is found, the algorithm raises the prices of storages in $\mathcal{D}'$ uniformly, until at least one content changes its preferred-storages so that a new preferred-storage graph can be built. If $\min\{p_m = \delta > 0\}$, let $p_m = p_m - \delta$ for all m. This step is necessary to keep the price as the lowest market-clearing price to obey the VCG principle [31]. After that, the algorithm goes back to step 3 and continues to build a new preferred-storage graph.

Step 7: The algorithm ends. The matching shows the allocation between contents and storages, and the price vector represents the final trading prices.

As a summary, an overview of the whole algorithm is given in Algorithm 2.

C. Properties of the Mechanism

In this subsection, we first discuss the influence of additional prices, then prove the convergence of the algorithm, and finally calculate its complexity.

TABLE I
SIMULATION PARAMETERS

Popularity parameter μ and σ	1 and 0.5
Popularity parameter a, b and t_0	$\mathcal{U}[0,3]$, $\mathcal{U}[4,12]$, and $\mathcal{U}[-75,25]$
Average user density $u(t)$ in 24 hours respectively ($10^{-5}m^{-2}$)	380,210,110,110,140,200, 300,650,1100,1260,1400,1570, 1530,1370,1310,1250,900,800, 940,1100,1200,1070,610,450
Delay parameter β_1, β_2, β_3 (ms)	1, 5, from 0 to 400
Number of SBSs I	24
Radius of SBSs R (m)	50
Total content number K	from 10000 to 20000
Overlapping percentage O	from 20% to 100%
Size of contents $S_{l,k}$ (GB)	$\mathcal{U}[0.1,1]$
Average storage capacity H (GB)	from 0 to 1000
Size of content blocks S (GB)	20
Additional price coefficient ω	from 0 to 4
Quantification accuracy α	from 10 to 1000

1) Influence of Additional Prices: The proposal of additional prices is able to limit the frequency of content replacement, especially at busy hours. Although the advantage of setting additional prices is non-trivial as shown in section V, the disadvantage still exists. When ω is too high, the caching result may not change between adjacent hours, which leads to an inefficient performance due to the time-dependent content popularity and user density. Therefore, choosing proper ω for the system is actually a tradeoff between the load of content replacement and the effectiveness of caching within each hour.

2) Convergence of the Market Matching Algorithm:

Proposition 4: Given that the valuations are presented by decimals with finite precision and finite upper-bound, the algorithm has convergence.

Proof: We define *the content's potential profit*, P_c^n as the maximum profit that content C_n can currently obtain from any one of the storages, and define *the storage's potential profit* P_s^m as the price of the storage D_m. The sum of all the potential profit of contents and storages P_{sum} represents the current maximum possible social welfare. Note that the existence of constricted set $\{\mathcal{C}', \mathcal{D}'\}$ makes it unable to satisfy all the $C_n \in \mathcal{C}'$ obtaining their profits, so P_{sum} may be exaggerated. At the beginning of the algorithm, $P_{sum}^{beg} \geq 0$ because $P_c^n \geq 0$ and $P_s^m \geq 0$ for any n and m. And at the end of the algorithm, we have $0 \leq P_{sum}^{end} \leq P_{sum}^{beg}$ because a possible social welfare is lower than an exaggerated one. In the algorithm, once the minimum price is above zero, we reduce the prices of all storages. This step doesn't change P_{sum} because $N = M$ and the total decrease of P_s^m equals to that of P_c^m. But when to raise the prices of storages in a constricted set $\{\mathcal{C}', \mathcal{D}'\}$, P_{sum} decreases by ΔP because $|\mathcal{C}'| > |\mathcal{D}'|$. Since $\Delta P > 0$ and $\leq P_{sum}^{beg} - P_{sum}^{end}$ is finite, P_{sum} can finally decrease to P_{sum}^{end} after certain amounts of iterations. ■

3) Complexity of the Market Matching Algorithm: The proof of its convergence shows that the complexity of this algorithm depends on the precision of valuations. We define the quantification accuracy as $\alpha = V/D$, where V is the maximum possible valuation and D is the minimum division of valuations. For the case that maximum possible valuation is 100 and minimum division of value is 0.1, we have $\alpha = 1000$.

Proposition 5: Given N as the total number of content blocks, the complexity of the market matching algorithm is $O(\alpha N^4)$.

Proof: The algorithm requires no more than $\alpha \cdot N$ times of iterations to get a perfect matching, since $0 \leq P_{sum}^{end} \leq P_{sum}^{beg} \leq VN$ and $\Delta P \geq D$. At the beginning of each iteration, a preferred-storage graph is constructed in the complexity of $O(N^2)$. Then, less than N times of BFS are executed, which takes $O(N^3)$. Finally, a constricted set is found and prices are changed in $O(N^2)$. Therefore, the algorithm takes $O(N^3)$ in each iteration, implying that the whole algorithm is $O(\alpha N^4)$. ■

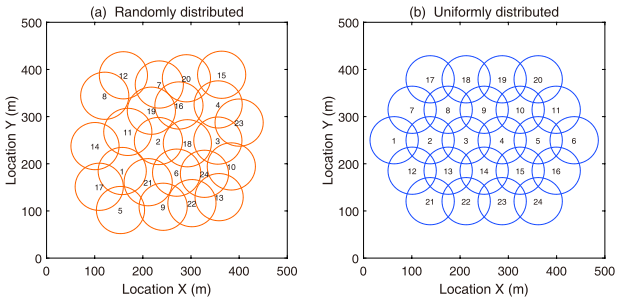


Fig. 7. Two SBSs-distribution settings. The left figure shows an example of the uniformly randomly generated distribution of 20 SBSs. The right figure shows a fixed uniform distribution of 24 SBSs.

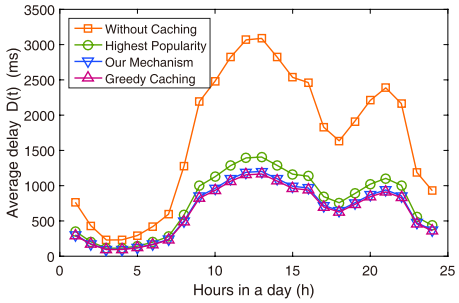


Fig. 8. Average delay profile in one day, where $O = 54\%$, $K = 10000$, $H = 1000GB$, $\beta_3 = 0ms$ and $\omega = 0$.

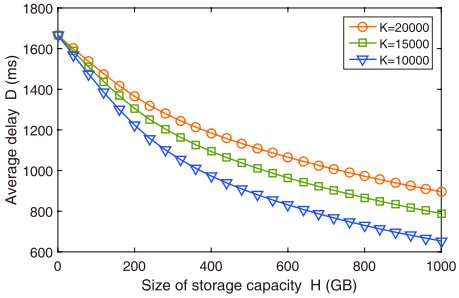


Fig. 9. Average delay D verses storage capacity H, with $K = 20000$, 15000, 10000, respectively.

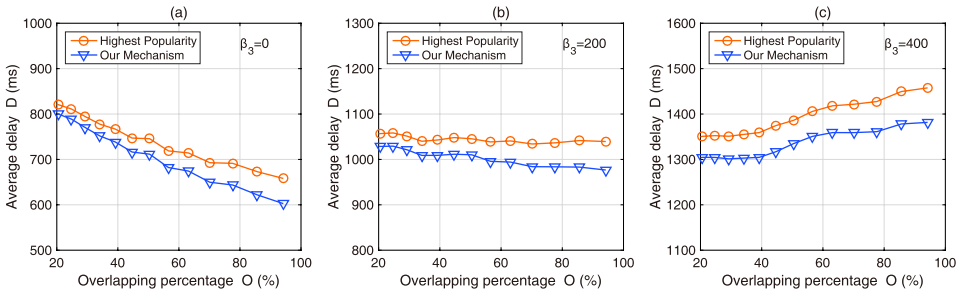


Fig. 10. Average delay D of different caching stratergies verses overlapping percentage O, where $\beta_3 = 0ms, 200ms, 400ms$, respectively.

As shown in our simulations, the practical complexity of this algorithm is not as high as $O(\alpha N^4)$. Moreover, we can set the standard caching size S greater, to adapt to the enormous number of contents.

V. Simulation Results

In this section, we simulate the performance of the proposed mechanism, the impact of system parameters, and the influence of the quantification accuracy. The simulation parameters are set in the first subsection, the simulation results and discussions are provided in the second subsection.

A. Simulation Parameters

Without loss of generality, we set $H_i = H$ for all $1 \leq i \leq I$. According to [34], the traffic loads in different days have the similar profile, so we set the variation of average user density $u(t)$ in a similar way, as shown in Table I. At each hour, the user density of each region conforms to Poisson distribution with mean value of $u(t)$.

Since the popularity of $C_{l,kx}$ is time-dependent, we assume that it has a similar time-evolutionary profile with the log-normal probability density function. This assumption accords with the study of [35] in characterizing the slow fading popularity of contents from time domain. And it also guarantees that the popularity distribution of large amount of contents at any given time conforms to Zipf-like distribution [36]. The log-normal probability density function is given by

$$f(x) = \begin{cases} \dfrac{1}{\sqrt{2\pi}\sigma x} \exp\left[\dfrac{-(\ln x - \mu)^2}{2\sigma^2}\right], & x > 0, \\ 0, & x \leq 0, \end{cases} \tag{21}$$

where the parameter μ and σ can be properly selected. Note that the maximum popularity and lifespan of different contents can be distinct from each other, therefore, we add extra parameters to the original function as $\phi^t_{l,k} = af(\frac{t-t_0}{b})$, where a determines the maximum popularity, b represents the lifespan, and t_0 is the time when $C_{l,k}$ is uploaded.

The SBSs are set in two different ways, as shown in Fig. 7, the random distribution and the uniform distribution. Here, we make sure that no more than three SBSs overlaps with each other, which is both for reality and simplicity.[5]

[5]The goal of deploying SBSs is to provide higher data transmission rate and larger coverage area, so it is unwise to put too many SBSs within a small region. In addition, the property we discussed in Proposition 3 intuitively has the same trend when we extend the area of overlapping from 3 to four or more, thus this constraint does not lose any generality.

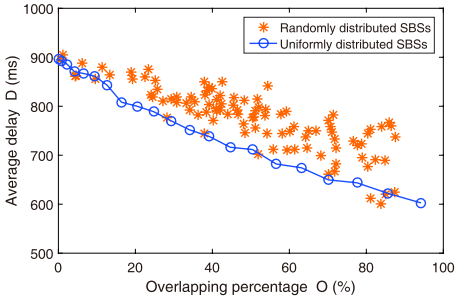


Fig. 11. Average delay D verses overlapping percentage O, for uniformly distributed and randomly distributed SBSs.

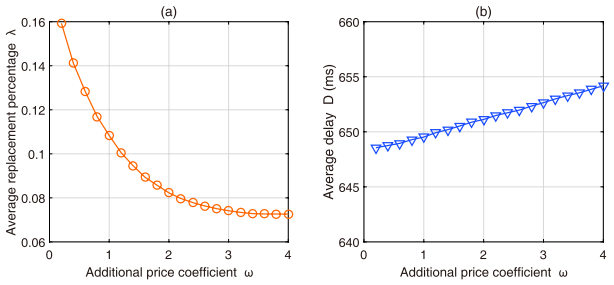


Fig. 12. The influence of additional price coefficient ω on average replacement percentage λ and average delay D.

Table I shows a more detailed list of parameters, where $\mathcal{U}[P, Q]$ means that the probability density is uniformly distributed from P to Q.

B. Simulation Results and Discussions

We first simulate a one-day situation to show how the average delay changes in a day, as given in Fig. 8, where $O = 54\%$, $K = 10000$, $H = 1000GB$, $\beta_3 = 0ms$, and $\omega = 0$. The uppermost curve shows how the average delay changes during a day without caching according to the predefined values in Table I. And the other three curves shows the average delay with different caching strategies. *Highest popularity* means caching the most popular contents in each SBS. *Greedy caching* comes from the algorithm proposed in [15], which allocates only one content block in each round (while our algorithm allocates I content blocks in each round). The outcome of our mechanism and that of the greedy caching algorithm are quite similar, and both surpass *Highest popularity*. Although the performance of our mechanism and that of *Greedy caching*

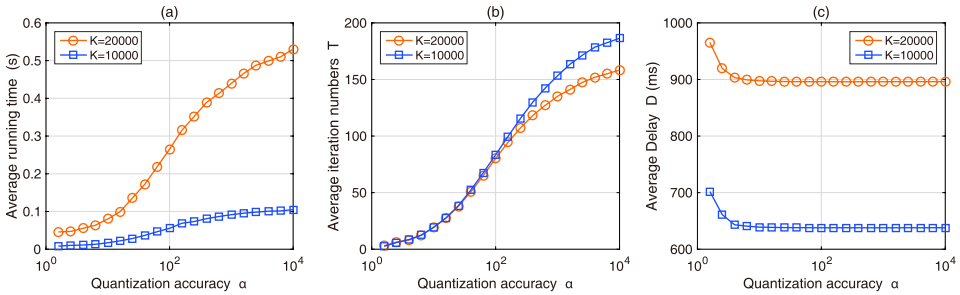


Fig. 13. The influence of quantization accuracy α on average iteration times η and average delay ratio D, with $K = 10000$ and $K = 20000$, respectively, where $O = 34\%$, $B = 100$.

are similar, much less rounds of allocation are needed with our mechanism, which implies that we can reduce the costs brought by communications among SPs and MNOs (since synchronization among them and submission of valuations are needed in each auction). Therefore, we can conclude that our mechanism is a suitable solution for small-cell caching.

The impact of H and K is shown in Fig. 9, where we set $O = 54\%$, $\beta_3 = 0ms$, $\omega = 0$, and let $K = 20000, 15000, 10000$ respectively. Here we use D to denote the average delay in 24 hours, given by $D = \frac{1}{24}\sum_{t=1}^{24} D(t)$, where $D(t)$ is the average delay of each hour. In this figure, we can see that when H gets greater, D decreases but the change rate of D decreases as well. So the same amount of storage makes greater difference in a low-capacity situation. It can also be observed that a greater number of contents K makes it more difficult to achieve low latency. Therefore, the simulation on the impact of K and H agrees with Propositions 1 and 2 in Section III.

The impact of O is shown in Fig. 10, where $H = 1000GB$, $K = 10000$, and $\beta_3 = 0ms, 200ms, 400ms$, respectively. Here we control O of uniformly distributed SBSs by multiplying the coordinates of SBSs with a constant, which is detailedly described in the Appendix. From all of the three subplots, we can see that the advantage of our mechanism over *highest popularity* becomes greater if O is higher. In Fig. 10 (a), where the choosing delay β_3 is ignorable, we find that D decreases with O, which agrees with Proposition 3. However, in Fig. 10 (b), where β_3 is set as $200ms$, the curve of D is becomes flat. And finally in Fig. 10 (c), where $\beta_3 = 400ms$, the correlation of O and D changes to positive instead of negative. These results imply that, the delay in overlapping regions can make a great difference on the average outcome. The caching efficiency can be further improved if we can shorten the "SBS choosing" procedure of mobile users in a practical cellular system.

In Fig. 11, we also provide the influence of O in the case of randomly distributed SBSs, with $\beta_3 = 0$. Since the number of possible random distributions of SBSs is infinite, and the distributions of SBSs and O have no one-to-one correspondence, we use the uniformly distributed SBSs as the benchmark to observe the outcome of randomly distributed SBSs. Here, 100 random cases are generated, and the results are presented by the star points on the figure. It can be observed that, for a certain O, the caching performance is not fixed. But roughly speaking, the correlation of O and D is similar to that of the uniform distribution, and the correlation coefficient in this simulation is around -0.8. We can also conclude that the floating range of D depends on O: A greater O brings D more uncertainty. And since the line of uniformly distributed SBSs separates most of the star points to the upward side, we can regard the uniform distribution as an effective way to deploy SBSs.

Then, we analyse the impact of ω, where we define $\lambda = \sum_{t=1}^{24} \lambda^t$. In Fig. 12, the relation of ω and λ as well as the relation of ω and D are given respectively, with $H = 1000GB$, $N = 10000$ and $O = 54\%$. It indicates that a higher additional price coefficient leads to a lower replacement percentage, but results in a higher average delay. When $\omega < 2$, λ decreases sharply but D increases slowly. This implies that a proper choice of ω can greatly reduce the load brought by content replacement, with only a trivial cost of average delay.

Finally, we analyse the influence of quantification accuracy α, on the time complexity of the algorithm and on the average delay of caching. Fig. 13 (a) shows the average running time (by seconds) of the market matching algorithm to complete each round of auction, where two curves with $K = 10000$ and $K = 20000$ are given. And in Fig. 13 (b), we count the average iteration number T of this algorithm, i.e., the times of rebuilding preferred-storage graphs to achieve perfect matching. It can be observed that, α does not contribute to T linearly as the theoretical analysis given by Proposition 5 ($T = O(\alpha N)$). What's more, the curve of $K = 20000$ is even below the curve of $K = 10000$, which indicates that $T = O(\alpha N)$ is an over estimated upper bound. Thus, the practical complexity of running this algorithm can be far below $O(\alpha N^4)$. To give another aspect of the impact of α, Fig. 13. (c) shows how α influences the system performance. It can be observed that when $\alpha > 10$, the two curves become almost flat and converged to certain values. Therefore, this algorithm can guarantee its efficiency even the quantification accuracy is not high enough, which can further reduce the practical complexity of executing the algorithm.

VI. Conclusions

In this paper, we proposed that caching can be applied as a service provided by SPs in mobile networks. We focused on the small-cell caching scenario and formulate the caching problem as how to minimize the average delay in consideration

of the competition among SPs for caching storages. In the theoretical analysis, we found that average delay has a positive correlation with total content number and a negative correlation with average storage capacity. In addition, overlapping percentage can also benefit the performance, as long as the choosing delay can be ignored. To solve the caching problem, we designed a mechanism that based on multi-object auctions, where the convergence of the algorithm can be guaranteed as long as the valuations are presented with finite precision. Simulation results testified the theoretical analysis and also showed that our solution leads to a better system performance, e.g., the average delay is reduced by 50% when $K = 10000$, $H = 1000GB$, $O = 54\%$. With a suitable setting of additional price, our mechanism can greatly reduce the load brought by content replacement. Moreover, the practical time complexity of executing the algorithm was revealed to be far below the theoretical upper bound.

APPENDIX

Proposition 3: In an approximately infinite cellular grid where SBSs with fixed coverage radius are uniformly distributed, given the constraint that 1) user density is uniform, 2) no coverage regions of four or more SBSs exist, and 3) the choosing delay can be ignored, the average delay based on a fixed caching result decreases when the overlapping percentage increases.

Proof: We assume that the coverage radius of a SBS is R and the distance of two adjacent SBSs is $2Rc$, where c is the *compress factor* which has a negative correlation with the overlapping percentage O. A smaller c indicates a smaller cellular grid and results in a greater O. To satisfy the constraint that no coverage regions of four or more SBSs exist, we let $1/\sqrt{3} < c < 1$. In the rest part of this proof, we discuss the influence of c instead of the influence of O.

Based on equation (5) and the assumption that user density is uniform, θ^t_{back} is only proportional to the area of the total coverage region of SBSs. When the parameter c gradually decreases from 1 to $1/\sqrt{3}$, we can deduce that θ^t_{back} is also decreased. Therefore, θ^t_{back} has a positive correlation with c, and we rewrite θ^t_{back} as $\theta^t_{back}(c)$ to express this inter-dependent character, where we have

$$\frac{\partial \theta^t_{back}(c)}{\partial c} > 0, \quad 1/\sqrt{3} < c < 1. \tag{22}$$

The average delay can be seen as the function of $\theta^t_{back}(c)$ and c, given by

$$D(t) = D\big(\theta^t_{back}(c), c\big), \quad 1/\sqrt{3} < c < 1. \tag{23}$$

Based on equations (9) and (10), we can give that

$$\frac{\partial D\big(\theta^t_{back}(c), c\big)}{\partial \theta^t_{back}(c)} \geq 0, \quad 1/\sqrt{3} < c < 1. \tag{24}$$

For any $1/\sqrt{3} < c_1 < c_2 < 1$, we can get $\theta^t_{back}(c_1) < \theta^t_{back}(c_2)$ by using inequality (22) and get $D\big(\theta^t_{back}(c_1), c_1\big) \leq D\big(\theta^t_{back}(c_2), c_1\big)$ by further using inequality (24). If we had the condition that $D\big(\theta^t_{back}(c_2), c_1\big) \leq D\big(\theta^t_{back}(c_2), c_2\big)$, then the conclusion $D\big(\theta^t_{back}(c_1), c_1\big) \leq D\big(\theta^t_{back}(c_2), c_2\big)$ could be obtained and we would have found the monotonicity between $D(t)$ and c. Therefore, in the rest part of this section, we prove that, with constant θ^t_{back}, there is a positive correlation between c and $D(t)$.

Based on equation (6) and the assumption of uniform user density, we can see that $\theta^{t,j}_{down}$ doesn't change with c. Along with the condition of fixed θ^t_{back} and the condition that choosing delay is zero, based on equation (9), we conclude that the delay of requesting a certain content is a fixed value, independent of the location of users. Since the caching allocation is also fixed, the only thing that influences $D(t)$ is the coverage percentage of any subset of SBSs in the whole region. Therefore we focus on how A_s/A_t changes with c, where A_s is the area of a subset of SBSs and A_t is the total area of all SBSs.

To simplify the expressions of A_s and A_t, we introduce another variable θ, where $c = \cos\theta$ and $\theta \in (0, \arccos\frac{1}{\sqrt{3}})$. Note that $\partial c/\partial\theta < 0$ in its definition domain. So our objective is to prove $\partial(A_s/A_t)/\partial\theta > 0$, which can also be denoted as $f'(\theta) > 0$, where $f(\theta) = A_s/A_t$ is the area percentage function.

As the demonstration of the top view shown in the Fig. 3, we need to consider two different situations in the definition domain of θ. The first one only involves 2-overlapping regions where $0 < \theta < \frac{\pi}{6}$, while the other one involves both 2-overlapping regions and 3-overlapping regions at the same time where $\frac{\pi}{6} < \theta < \arccos\frac{1}{\sqrt{3}}$. For better reading, we denote $\arccos\frac{1}{\sqrt{3}}$ as θ_u.

We denote the area of the coverage region of one SBS as A_1, the area of 2-overlapping region as A_2, and the area of 3-overlapping region as A_3. Their expressions can be simplified as:

$$A_1 = \pi R^2, \quad \theta \in (0, \theta_u),$$
$$A_2 = 2R^2(\theta - \cos\theta\sin\theta), \quad \theta \in (0, \theta_u),$$
$$A_3 = \begin{cases} 0, & \theta \in (0, \frac{\pi}{6}), \\ 3R^2(\theta - \frac{\pi}{6}) + \sqrt{3}R^2\cos^2\theta - 3R^2\sin\theta\cos\theta, & \\ & \theta \in (\frac{\pi}{6}, \theta_u). \end{cases}$$

Now we use A_1, A_2, and A_3 to express A_s and A_t. For A_t as the total area, we assume that the number of hexagons in the cellular grid is approximately infinite so that the influence of its boundary can be ignored. To get the proportion of the numbers of A_1, A_2, and A_3, we find a minimum repeated unit in the infinite grid. This process is the same as finding a cell in the molecular structure of graphite, leading $A_1{:}A_2{:}A_3 = 1{:}3{:}2$. Thus we have

$$A_t = M_t \cdot (A_1 - 3A_2 + 2A_3),$$

where M_t is a large integer to represent the total number of SBSs. In this way, A_t is expressed by its equivalent average coverage area of a single SBS.

For A_s as the area of a subset of SBSs, the proportions of A_2, and A_3 in A_s is less than those in A_t, because hexagons at the boundary have less overlapping regions. Hence, we have

$$A_s = M_s \cdot (A_1 - xA_2 + yA_3),$$

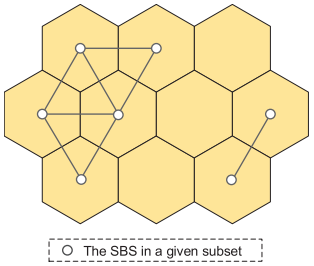


Fig. 14. The transformation from a real situation to a planar graph, where adjacent SBSs in the subset are connected by 2-overlapping regions. The given case indicates that $V = 7$, $F = 3$, $E = 8$ and $C = 2$.

where $0 \le x < 3$, $0 \le y < 2$ and M_s is the number of the SBSs in the given subset. Another constraint of x and y is introduced later.

Since A_3 has different expressions in different situations, we discuss $\theta \in (0, \frac{\pi}{6})$ and $\theta \in (\frac{\pi}{6}, \arccos\frac{1}{\sqrt{3}})$ respectively. The former case only involves 2-overlapping regions but the latter case involves 3-overlapping regions as well as 2-overlapping regions, as shown in the Fig. 3.

Case 1: $\theta \in \left(0, \frac{\pi}{6}\right)$.

$$f'(\theta) = \frac{M_s}{M_t}\left(\frac{A_1 - xA_2}{A_1 - 3A_2}\right)' = \frac{M_s}{M_t}\frac{(3-x)A_1A_2'}{(A_1 - 3A_2)^2}. \tag{25}$$

Since $A_2' = 2R^2(1 - \cos 2\theta) > 0$, $3 - x > 0$ and $A_1 - 3A_2 > 0$, we can easily get $f'(\theta) > 0$.

Case 2: $\theta \in \left(\frac{\pi}{6}, \arccos\frac{1}{\sqrt{3}}\right)$.

$$\begin{aligned} f'(\theta) &= \frac{M_s}{M_t}\left(\frac{A_1 - xA_2 + yA_3}{A_1 - 3A_2 + 2A_3}\right)' \\ &= \frac{M_s}{M_t}\frac{\sqrt{3}\sin\theta}{6\cos^3\theta}\Big[2\pi - y\pi + 6y\theta - 4x\theta\Big] \\ &= g(\theta)\Big[(2-y)(\pi - 2\theta) + 4\theta(1 + y - x)\Big], \end{aligned} \tag{26}$$

where $g(\theta) = \frac{M_s}{M_t}\frac{\sqrt{3}\sin\theta}{6\cos^3\theta}$.

In this case, we need another constraint of x and y to complete the proof. Given a subset of SBSs, we regard each SBS as a vertex in the planar graph. Each 2-overlapping region is an edge between two adjacent SBSs, and each 3-overlapping region is a face enclosed by three adjacent 2-overlapping regions. This abstraction process is illustrated in the Fig. 14.

The Eular Formula in the planar graph [32] is given by

$$V + (F + 1) - E = C + 1, \tag{27}$$

where V is the number of vertexes, F is the number of faces, E is the number of edges and C is the number of connected subgraphs. Thus, the instance in the Fig. 14 implies $V = 7$, $F = 3$, $E = 8$ and $C = 2$. And for any given subset of SBSs, $C \ge 0$ can be satisfied.

We apply the Eular Formula on our model as:

$$\begin{aligned} &(V + F - E) = C \ge 0 \\ \leftrightarrow\ &(V + F - E)/V \ge 0 \\ \leftrightarrow\ &1 + y - x > 0. \end{aligned}$$

Since $y < 2$ and $\theta \in (0, \pi/2)$, based on the equation (26), we finally have $f'(\theta) > 0$.

Conclusion: In both cases the area percentage function $f(\theta)$ has a positive correlation with θ respectively. When $\theta = \frac{\pi}{6}$, S_3 remains its continuality, implying that $f(\theta)$ is continuous. Thus $f(\theta)$ increases with θ in $(0, \arccos\frac{1}{\sqrt{3}})$. Therefore, the coverage percentage of any given subset of SBSs in the whole area decreases when c gets greater in $(\frac{1}{\sqrt{3}}, 1)$.

As a result, the necessary condition $D\left(\theta^t_{back}(c_2), c_1\right) \le D\left(\theta^t_{back}(c_2), c_2\right)$ mentioned above can be obtained and the positive correlation between $D(t)$ and c can be proved. ■

REFERENCES

[1] Z. Hu, Z. Zheng, T. Wang, and L. Song, "Small-cell caching mechanism for multi-service providers," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Hong Kong, Apr./May 2015, pp. 61–62.

[2] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 358–380, 1st Quart., 2014.

[3] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, Apr. 2010.

[4] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "NVS: A substrate for virtualizing wireless resources in cellular networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1333–1346, Oct. 2012.

[5] A. Belbekkouche, M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1114–1128, 4th Quart., 2012.

[6] H. Wen, P. K. Tiwary, and T. Le-Ngoc, *Wireless Virtualization*. New York, NY, USA: Springer, Aug. 2013.

[7] M. Yang, Y. Li, L. Zeng, D. Jin, and L. Su, "Karnaugh-map like online embedding algorithm of wireless virtualization," in *Proc. Int. Symp. Wireless Pers. Multimedia Commun. (WPMC)*, Taipei, Taiwan, Sep. 2012, pp. 594–598.

[8] X. Zhang, Y. Li, D. Jin, L. Su, L. Zeng, and P. Hui, "Efficient resource allocation for wireless virtualization using time-space division," in *Proc. IEEE 8th Int. Conf. Wireless Commun. Mobile Comput. (IWCMC)*, Limassol, Aug. 2012, pp. 59–64.

[9] F. Fu and U. C. Kozat, "Stochastic game for wireless network virtualization," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 84–97, Feb. 2013.

[10] Y. Zhou, Y. Li, G. Sun, D. Jin, L. Su, and L. Zeng, "Game theory based bandwidth allocation scheme for network virtualization," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Miami, FL, USA, Dec. 2010, pp. 1–5.

[11] C. Mobile, "C-RAN: The road towards green RAN," China Mobile, Beijing, China, Tech. Rep. V2.5, Oct. 2011. [Online]. Available: http://labs.chinamobile.com/report/view_59826

[12] A. Kearney. (2012). *The Rise of the Tower Business*. [Online]. Available: http://www.atkearney.com

[13] 3GPP, "Technical specification group services and system aspects; network sharing; architecture and functional description," 3rd Generat. Partnership Project (3GPP), Sophia-Antipolis, France, Tech. Rep. TS 23.251 V11.5.0, Mar. 2013. [Online]. Available: http://www.3gpp.org/ftp/Specs/htmlinfo/23251.htm

[14] 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 Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[15] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Orlando, FL, USA, Mar. 2012, pp. 1107–1115.

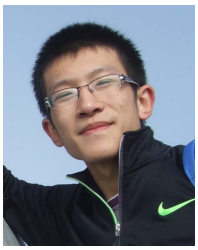
[16] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Trans. Commun.*, vol. 62, no. 10, pp. 3665–3677, Oct. 2014.

[17] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Multicast-aware caching for small cell networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Istanbul, Turkey, Apr. 2014, pp. 2300–2305.

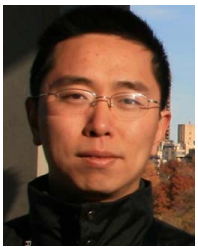
[18] M. Dehghan *et al.*, "On the complexity of optimal routing and content caching in heterogeneous networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Hong Kong, Apr./May 2015, pp. 936–944.

[19] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Wireless video content delivery through coded distributed caching," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Ottawa, ON, Canada, Jun. 2012, pp. 2467–2472.
[20] K. Poularakis, V. Sourlas, P. Flegkas, and L. Tassiulas, "On exploiting network coding in cache-capable small-cell networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Funchal, Portugal Jun. 2014, pp. 1–5.
[21] K. Hamidouche, W. Saad, and M. Debbah, "Many-to-many matching games for proactive social-caching in wireless small cell networks," in *Proc. 12th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, Hammamet, Tunisia, May 2014, pp. 569–574.
[22] K. Poularakis and L. Tassiulas, "Exploiting user mobility for wireless content delivery," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Istanbul, Turkey, Jul. 2013, pp. 1017–1021.
[23] T. Nakamura *et al.*, "Trends in small cell enhancements in LTE advanced," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 98–105, Feb. 2013.
[24] V. Krishna, *Auction Theory*. San Diego, CA, USA: Academic, 2009.
[25] G. Demange, D. Gale, and M. Sotomayor, "Multi-item auctions," *J. Political Econ.*, vol. 94, no. 4, pp. 863–872, 1986.
[26] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Sydney, NSW, Australia, Jun. 2014, pp. 1897–1903.
[27] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.
[28] H. Xu, T. Zhang, Z. Zeng, and D. Liu, "Distributed user association for delay-load tradeoffs in energy harvesting enabled HetNets," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, New Orleans, LA, USA, Mar. 2015, pp. 386–390.
[29] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Netw.*, vol. 11, no. 3, pp. 333–340, May 2005.
[30] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: Wiley, 1990.
[31] H. B. Leonard, "Elicitation of honest preferences for the assignment of individuals to positions," *J. Political Economy*, vol. 91, no. 3, pp. 461–479, Jun. 1983.
[32] D. B. West, *Introduction to Graph Theory*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.
[33] L. Lovász and M. D. Plummer, Eds., *Matching Theory*. Providence, RI, USA: AMS, 2009.
[34] J. Lorincz, T. Garma, and G. Petrovic, "Measurements and modelling of base station power consumption under real traffic loads," *Sensors*, vol. 12, no. 4, pp. 4281–4310, Mar. 2012.
[35] Z. Avramova, S. Wittevrongel, H. Bruneel, and D. De Vleeschauwer, "Analysis and modeling of video popularity evolution in various online video content systems: Power-law versus exponential decay," in *Proc. 1st Int. Conf. Evolving Internet*, Aug. 2009, pp. 95–100.
[36] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM*, New York, NY, USA, Mar. 1999, pp. 126–134.

Zhiwen Hu (S'15) received the B.S. degree in electronic engineering from Peking University, China, in 2015, where he is currently pursuing the Ph.D. degree with the School of Electronics Engineering and Computer Science. His current research interests include caching in wireless networks, game theory in 5G networks, and wireless M2M communications.



Zijie Zheng (S'14) received the B.S. degree in electronic engineering from Peking University, China, in 2014, where he is currently pursuing the Ph.D. degree with the School of Electronics Engineering and Computer Science. His current research interests include game theory in 5G networks, wireless powered networks, and mobile social networks.



Tao Wang (SM'11) received the B.S. and Ph.D. degrees from Peking University, in 1999 and 2006, respectively. He was a Post-Doctoral Researcher with Intel from 2006 to 2008, where he was a Staff Research Scientist. He joined Peking University as a Faculty Member in 2010. He is currently an Associate Professor with the School of Electronics Engineering and Computer Science, Peking University. In the past five years, he received many research fundings and authored over 30 research papers in top journals/conferences, such as the *ISCA*, the MICRO, the HPCA, the MobiCom, the IEEE TRANSACTIONS ON MOBILE COMPUTING, and the IEEE TRANSACTIONS ON COMPUTERS. His current research interests are computer architecture, reconfigurable wireless network architecture, and wireless e-health. He is also a Chinese Computer Society Senior Member. He received the 2008 Intel China Employee of the Year award, which was the highest individual award at Intel, China.

Lingyang Song (S'03–M'06–SM'12) received the Ph.D. degree from the University of York, U.K., in 2007, where he received the K.M.Stott Prize for excellent research. He was a Research Fellow with the University of Oslo, Norway, until rejoining Philips Research U.K., in 2008. In 2009, he joined the School of Electronics Engineering and Computer Science, Peking University, China, as a Full Professor. He was an Editor of two text books entitled *Wireless Device-to-Device Communications and Networks* and *Full-Duplex Communications and Networks* (Cambridge University Press). His main research interests include MIMO, cognitive and cooperative communications, security, and big data. He is currently on the Editorial Board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. He has been an IEEE Distinguished Lecturer since 2015. He is the recipient of the IEEE Leonard G. Abraham Prize in 2016 and the IEEE Asia Pacific Young Researcher Award in 2012.

Xiaoming Li (S'87–M'87–SM'03) received the Ph.D. degree in computer science from the Stevens Institute of Technology, USA. He is currently a Professor with Peking University, China. His research interests include search engine and web mining. He is a fellow of Computer Federation of China and a member of Eta Kappa Nu. He serves on the Editorial Board of *Concurrency and Computation* (John Wiley). He received the CCF Wang Xuan Award and Outstanding Educator Award, in 2013 and 2014.