Decentralized Asynchronous Coded Caching in Fog-RAN

Wenlong Huang, Yanxiang Jiang, *Member, IEEE*, Mehdi Bennis, *Senior Member, IEEE*, Fu-Chun Zheng, *Senior Member, IEEE*, Haris Gacanin, *Member, IEEE*, and Xiaohu You *Fellow, IEEE*

Abstract— **In this paper, we investigate asynchronous coded caching in fog radio access networks (F-RAN). To minimize the fronthaul load, the encoding set collapsing rule and encoding set partition method are proposed to establish the relationship between the coded-multicasting contents in asynchronous and synchronous coded caching. Furthermore, a decentralized asynchronous coded caching scheme is proposed, which provides asynchronous and synchronous transmission methods for different delay requirements. The simulation results show that our proposed scheme creates considerable coded-multicasting opportunities in asynchronous request scenarios.**

Index Terms— **Fog radio access networks, asynchronous coded caching, coded-multicasting, fronthaul load.**

I. Introduction

With the rapid proliferation of smart devices and mobile application services, wireless networks have been suffering an unprecedented data traffic pressure in recent years, especially at peak-traffic moments. Fog radio access networks (F-RAN), which can effectively reduce the data traffic pressure by placing popular contents closer to users, have been receiving significant attention from both industry and academia. In F-RAN, fog access points (F-APs) are distributed at the edges and connected to the cloud server through fronthaul links. F-APs can use edge computing and caching resources to bring users better quality of experience [1]. Meanwhile, as just a few popular content sources account for most of the traffic load, edge caching has become a trend for content delivery [2], [3]. Moreover, coded caching was firstly proposed in [4] and [5] by encoding the delivered contents to further reduce network congestion.

The main idea of coded caching is that the contents stored in the caches can be used to create coded-multicasting opportunities, such that a single coded-multicasting content transmitted by the cloud server can be useful to a large number of users simultaneously even though they are not requesting the same content. In [4], Maddah-Ali and Niesen proposed a centralized coded caching scheme, in which the centrally coordinated placement phase needs the knowledge of the number of active users in the delivery phase. A decentralized coded caching scheme was further proposed in [5], which achieves order-optimal memory-load tradeoff in the asymptotic regime of infinite file size. The authors of [6] presented a strategy which partitions the file library into subsets of approximately uniform request probability and applies the strategy in [5] to each subset. The authors of [7] studied the case that the file popularity has multiple different levels. A scheme consisting of a random popularity-based caching policy and chromatic-number index coding delivery was proposed in [8], which was proven to be order optimal in terms of average rate. All the schemes in [4]–[8] considered the coded caching problem for the case that user requests are synchronous, i.e., synchronous coded caching. However, user demands for contents is typically asynchronous [9] in reality. The asynchronous request case was first mentioned in [5], and the authors applied the proposed decentralized synchronous coded caching scheme to an asynchronous request scenario in a simple way. In [10], the authors proposed a linear programming formulation for the offline case that the server knows the arrival time before starting transmission. As for the online case that user requests are revealed to the server over time, they considered the situation that users do not have deadlines but wish to minimize the overall completion time.

Motivated by the aforementioned discussions, it is important to study the coded caching problem when user requests are asynchronous, i.e., asynchronous coded caching. We consider the online case with a given maximum request delay to reduce the worst-case load of the fronthaul links in F-RAN. We propose a decentralized asynchronous coded caching scheme, which effectively exploits the coded-multicasting opportunities. Our proposed scheme is applicable for various asynchronous request scenarios by providing asynchronous and synchronous transmission methods, which are chosen according to different delay requirements.

II. System Model

Consider the F-RAN where there are K F-APs and each F-AP serves multiple users. Assume that the users request contents asynchronously during the time interval $(0, T]$. Let $\mathcal{K} = \{1, 2, \cdots, k, \cdots, K\}$ denote the index set of the considered K F-APs. The cloud server has access to a content library of N files, denoted by $W_1, W_2, \cdots, W_N$. Let $\mathcal{N} = \{1, 2, \cdots, n, \cdots, N\}$ denote the index set of N files with $N \geq K$. Assume that the size of each file is F bits and the content library has a uniform popularity distribution. For each F-AP, only one

This work is accepted by IEEE VTC 2018 FALL.

W. Huang and Y. Jiang are with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China, the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China, and the Key Laboratory of Wireless Sensor Network & Communication, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China. (e-mail: yxjiang@seu.edu.cn)

M. Bennis is with the Centre for Wireless Communications, University of Oulu, Oulu 90014, Finland. (e-mail: bennis@ee.oulu.fi)

F. Zheng is with the School of Electronic and Information Engineering, Harbin Institute of Technology, Shenzhen 518055, China, and the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China. (e-mail: fzheng@ieee.org)

H. Gacanin is with the Nokia Bell Labs, Antwerp 2018, Belgium. (e-mail:haris.gacanin@nokia-bell-labs.com)

X. You is with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China. (e-mail:xhyu@seu.edu.cn)

of its served users requests one file during the time interval $(0, T]$, while the F-AP informs the cloud server of the request immediately. For description convenience, we say K F-APs request contents asynchronously during the considered time interval $(0, T]$, where each F-AP only requests one file. Each F-AP has an isolated normalized (by F) cache size M for $0 < M < N$.

In the placement phase, the F-APs are given access to the content library. By using the same setting of this phase in [5], F-AP k is able to store its cache content Z_k using the content library independently of the other F-APs, i.e., in a decentralized manner. Let ϕ_k denote the caching function of F-AP k which maps the content library into the corresponding cache content, i.e., $Z_k = \phi_k(W_1, W_2, \cdots, W_N)$. Note that the size of Z_k is MF bits.

In the delivery phase, the cache contents of all the F-APs are first informed to the cloud server and then noted as cache records by the cloud server. Without loss of generality, assume that the time interval $(0, T]$ is divided into B time slots with $B \geq 2$. Let $\Delta t = T/B$ denote the time duration of each time slot. Then time slot $b \in \{1, 2, \cdots, B\}$ represents the time interval $((b-1)\Delta t, b\Delta t]$. Let $\mathcal{U}_b \subseteq \mathcal{K}$ denote the index set of the F-APs whose requests arrive during time slot b with $\mathcal{U}_b \neq \varnothing$. It is assumed that the cloud server is informed of the requests of the F-APs in $\mathcal{U}_b$ during time slot b and processes them in a unified manner, i.e., the cloud server transmits the coded-multicasting content to all the K F-APs through the fronthaul links at the end of each time slot for the online case. Suppose that the maximum request delay is $\Delta b \in \{1, 2, \cdots, B\}$ time slots, where Δb denotes the maximum number of time slots it takes for an F-AP to recover its requested file. Note that we do not consider the time that it takes for the cloud server to transmit the corresponding contents and the time that it takes for each F-AP to transmit the recovered file to the served user. Therefore, the cloud server can fulfill the requests of the F-APs in $\mathcal{U}_b$ by the end of the time slot $b + \Delta b - 1$.

Let $d_k \in \mathcal{N}$ denote the index of the file requested by F-AP k during $(0, T]$, and $\boldsymbol{d}_b \in \mathcal{N}^{|\mathcal{U}_b|}$ denote the request vector of the corresponding F-APs in $\mathcal{U}_b$. Let ψ_b denote the encoding function of the cloud server at the end of time slot b, which maps the files $W_1, W_2, \cdots, W_N$, the cache contents $Z_1, Z_2, \cdots, Z_K$, and the requests $\boldsymbol{d}_b$ to the coded-multicasting content $X_b \triangleq \psi_b(W_1, W_2, \cdots, W_N, Z_1, Z_2, \cdots, Z_K, \boldsymbol{d}_b)$. Let θ_k denote the decoding function of F-AP k, which maps the received coded-multicasting contents $X_1, X_2, \cdots, X_B$, the cache content Z_k, and the request d_k to the estimate $\hat{W}_{d_k} = \theta_k(X_1, X_2, \cdots, X_B, Z_k, d_k)$ of the requested file W_{d_k} of F-AP k. Each F-AP should be able to recover its requested file successfully from its cached content and the received coded-multicasting content, and then transmit it to the served user. An asynchronous coded caching scheme is said to be feasible if and only if the following condition is satisfied:

$$\lim_{F\to\infty} \max_{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_B} \max_{k\in\mathcal{K}} P\left(\hat{W}_{d_k} \neq W_{d_k}\right) = 0.$$

Note that the worst-case propability of error over all possible requests $\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_B$ with infinite F is maximized in the above condition. The objective of this paper is to find a feasible asynchronous coded caching scheme to minimize the worst-case normalized fronthaul load (over all possible requests $\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_B$) in the delivery phase with a given Δb.

III. The Proposed Decentralized Asynchronous Coded Caching Scheme

In this section, we first propose the encoding set collapsing rule. Then, we present the encoding set partition method. Finally, a novel decentralized asynchronous coded caching scheme is proposed to minimize the fronthaul load.

A. The Proposed Encoding Set Collapsing Rule

Asynchronous coded caching and synchronous coded caching are supposed to be under the same condition when their system parameters M, K, and N are the same. As in the conventional synchronous coded caching schemes under the same condition, such as the Maddah-Ali-Niesen's decentralized scheme [5], $\mathcal{S} \subseteq \mathcal{K}$ for any $s = |\mathcal{S}| \in \{1, 2, \cdots, K\}$ is called an encoding set if a single coded-multicasting content can be useful to the F-APs in $\mathcal{S}$ simultaneously. It can be observed that the subset of $\mathcal{S}$ is also an encoding set. In order to differentiate the same subfile in asynchronous and synchronous coded caching, let $W^{\mathrm{a}}_{k,\mathcal{S}}$ and $W^{\mathrm{s}}_{k,\mathcal{S}}$ denote the bits of the file requested by F-AP k cached exclusively at the F-APs in $\mathcal{S}$ for asynchronous and synchronous coded caching, respectively.

Consider that the requests of the F-APs in $\mathcal{K}\backslash\mathcal{U}_1$ have not arrived yet during time slot 1. Assume that $\mathcal{U}_1 \cap \mathcal{S} \neq \varnothing$ and the requests in $\boldsymbol{d}_1$ should be fulfilled at the end of time slot 1. The cloud server needs to transmit a coded-multicasting content which is useful to the F-APs in $\mathcal{U}_1 \cap \mathcal{S}$ at the end of time slot 1. Thus, we say that the encoding set $\mathcal{S}$ in synchronous coded caching collapses into a subset of $\mathcal{S}$, i.e., $\mathcal{U}_1 \cap \mathcal{S}$, for transmitting the corresponding coded-multicasting content in asynchronous coded caching. Recall that by applying the scheme in [5], the coded-multicasting content that the cloud server transmits for $\mathcal{S}$ in synchronous coded caching is $\oplus_{k\in\mathcal{S}} W^{\mathrm{s}}_{k,\mathcal{S}\backslash\{k\}}$, where $\oplus$ denotes bitwise XOR operation. Accordingly, the cloud server transmits $\oplus_{k\in(\mathcal{S}\cap\mathcal{U}_1)} W^{\mathrm{a}}_{k,(\mathcal{S}\cap\mathcal{U}_1)\backslash\{k\}}$ at the end of time slot 1. According to the above discussions, it is evident that there exists some relationship, called encoding set collapsing rule, between the coded-multicasting contents in asynchronous and synchronous coded caching.

During time slot b, let $\mathcal{U}_y$ and $\mathcal{U}_n$ denote the index sets of the F-APs from which the requests have arrived and not arrived, respectively. For any $\mathcal{S}_1 \subseteq \mathcal{U}_y$ and $\mathcal{S}_2 \subseteq \mathcal{U}_n$, the encoding set $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ in synchronous coded caching collapses into $\mathcal{S}_1$ in asynchronous coded caching. Accordingly, $\oplus_{k\in(\mathcal{S}_1\cup\mathcal{S}_2)} W^{\mathrm{s}}_{k,(\mathcal{S}_1\cup\mathcal{S}_2)\backslash\{k\}}$ collapses into $\oplus_{k\in\mathcal{S}_1} W^{\mathrm{a}}_{k,(\mathcal{S}_1\cup\mathcal{S}_2)\backslash\{k\}}$, which will be practically transmitted by the cloud server at the end of time slot b in asynchronous coded caching.

B. The Proposed Encoding Set Partition Method

Utilizing our proposed encoding set collapsing rule, we now consider what contents are transmitted in asynchronous coded caching. In order to fulfill the requests of the F-APs

Algorithm 1 The Proposed Encoding Set Partition Method

```
1: Initialize $i$, $\beta$, $\gamma$.
2: while $\mathcal{S} \neq \varnothing$ do
3:     $i = i + 1$.
4:     while $\mathcal{U}_{\beta+1} \cap \mathcal{S} = \varnothing$ do
5:         $\beta = \beta + 1$.
6:     end while
7:     if $\gamma - \beta \geq \Delta b$ then
8:         $\mathcal{S}_i = \mathcal{S} \cap \left(\cup_{b=\beta+1}^{\beta+\Delta b} \mathcal{U}_b\right)$,
9:         $\beta = \beta + \Delta b$.
10:    else
11:        $\mathcal{S}_i = \mathcal{S} \cap \left(\cup_{b=\beta+1}^{\gamma} \mathcal{U}_b\right)$.
12:    end if
13:    $\mathcal{S} = \mathcal{S} \backslash \mathcal{S}_i$.
14: end while
```

in asynchronous coded caching, $\mathcal{S}$ may need to be collapsed into a subset of $\mathcal{S}$ many times to transmit the corresponding content at the end of different time slots. For a given Δb, only the requests of the F-APs in $\cup_{i=\max\{1,b-\Delta b+1\}}^{b} \mathcal{U}_i$, referred to as active F-APs, need to be fulfilled during time slot b. Let $\mathcal{U}^{\mathrm{a}} = \cup_{i=\max\{1,b-\Delta b+1\}}^{b} \mathcal{U}_i$ denote the index set of the active F-APs during time slot b. Then, only the files requested by the F-APs in $\mathcal{U}^{\mathrm{a}}$ can be encoded with each other, which means that $\mathcal{S}$ collapses into $\mathcal{S} \cap \mathcal{U}^{\mathrm{a}}$. Moreover, to minimize the fronthaul load, it is equivalent to partition $\mathcal{S}$ into the minimum number of nonoverlapping subsets for transmission in asynchronous coded caching.

Let $(\beta\Delta t, \gamma\Delta t]$ denote the active time interval of $\mathcal{S}$ if $\left(\left(\cup_{b=1}^{\beta} \mathcal{U}_b\right) \cup \left(\cup_{b=\gamma+1}^{B} \mathcal{U}_b\right)\right) \cap \mathcal{S} = \varnothing$ and $\mathcal{U}_b \cap \mathcal{S} \neq \varnothing$ for $b = \beta+1$ and γ, where β and γ are integers with $0 \leq \beta < \gamma \leq B$. Let $\mathcal{S}_i$ denote the ith encoding subset that $\mathcal{S}$ is partitioned into for a given Δb. The detailed encoding set partition method is presented in Algorithm 1.

C. The Proposed Asynchronous Coded Caching Scheme

According to the above discussions, we propose the decentralized asynchronous coded caching scheme which implements the encoding set partition method. In the placement phase, each F-AP randomly selects MF/N bits of each file with uniform probability and fetch them to fill its cache, which is the same as the Maddah-Ali-Niesen's decentralized synchronous coded caching scheme. Note that the placement procedure does not require any coordination and can be operated in a decentralized manner. More specifically, our proposed scheme can operate in the placement phase with an unknown number of F-APs.

In the delivery phase, we propose asynchronous and synchronous transmission methods for the online case, which can be chosen by the cloud server. Note that asynchronous or synchronous here means that the cloud server transmits the coded-multicasting contents asynchronously or synchronously.

1) Asynchronous Transmission Method: When $\Delta b < B$, the asynchronous transmission method is chosen. As the requests in $\boldsymbol{d}_b$ need to be fulfilled by the end of the time slot $b + \Delta b - 1$, we should try to fulfill the requests at the end of time slot $b + \Delta b - 1$ so that sufficient coded-multicasting opportunities can be created. If $\Delta b > 1$, no contents are transmitted at the end of time slot $1, 2, \cdots, \Delta b - 1$. Then, only the requests of the F-APs in $\mathcal{U}^{\mathrm{a}}$ need to be fulfilled by the cloud server at the end of the time slots between $\Delta b - 1$ and B. For description convenience, we say the subfile $W_{k,\mathcal{S}}$ is of type s with $s \in \{0, 1, \cdots, K\}$. Thus, the cloud server transmits a single coded-multicasting content for $\mathcal{S}$ by encoding the subfiles of type $s - 1$ [5]. At the end of this time slot, the cloud server firstly partitions each file W_n into $K+1$ types of nonoverlapping subfiles. Note that there are $\left(\begin{array}{c} K \\ s \end{array}\right)$ subfiles of the form $\left(W_{n,\mathcal{S}}^{\mathrm{a}} : \mathcal{S} \subseteq \mathcal{K}, |\mathcal{S}| = s\right)$ for each type $s \in \{0, 1, \cdots, K\}$, whose sizes are calculated based on the updated cache records. For any s, define $\underline{\chi} = \max\{1, s + |\mathcal{U}_{b-\Delta b+1}| - |\mathcal{K}|\}$ and $\overline{\chi} = \min\{s, |\mathcal{U}_{b-\Delta b+1}|\}$. Consider any $\chi \in \left\{\underline{\chi}, \underline{\chi} + 1, \cdots, \overline{\chi}\right\}$. Focus on an encoding subset $\mathcal{S}^1 \subseteq \mathcal{U}_{b-\Delta b+1}$ with $\left|\mathcal{S}^1\right| = \chi$ and an encoding subset $\mathcal{S}^2 \subseteq \mathcal{K} \backslash \mathcal{U}_{b-\Delta b+1}$ with $\left|\mathcal{S}^2\right| = s - \chi$. Recall that the F-APs in $\left(\mathcal{S}^1 \cup \mathcal{S}^2\right) \backslash \{k\}$ share a subfile which is not available at the cache content Z_k and requested by F-AP $k \in \left(\mathcal{S}^1 \cup \mathcal{S}^2\right)$. For any $\mathcal{S}^1$ and $\mathcal{S}^2$ with any s, in order to avoid some subfiles are transmitted repeatedly, no contents are transmitted if $W_{k,(\mathcal{S}^1 \cup \mathcal{S}^2)\backslash\{k\}}^{\mathrm{a}} = \varnothing$ for $k \in \left(\left(\mathcal{S}^1 \cup \mathcal{S}^2\right) \cap \mathcal{U}^{\mathrm{a}}\right)$. Otherwise, the cloud server transmits the coded-multicasting content as follows:

$$\oplus_{k \in ((\mathcal{S}^1 \cup \mathcal{S}^2) \cap \mathcal{U}^{\mathrm{a}})} W_{k,(\mathcal{S}^1 \cup \mathcal{S}^2)\backslash\{k\}}^{\mathrm{a}}.$$

After the transmission is completed, each F-AP in $\mathcal{U}_{b-\Delta b+1}$ recovers the desirable subfiles of its requested file. Then, each F-AP in $\mathcal{U}_{b-\Delta b+1}$ transmits the recovered subfiles and the corresponding subfiles available in its cache to its served user immediately; thus the user can recover the desirable file. While each F-AP in $\mathcal{U}^{\mathrm{a}} \backslash \mathcal{U}_{b-\Delta b+1}$ also recovers the corresponding desirable subfiles, and then transmit them to its served user at this time. In addition, the cloud server needs to update the cache records of the active F-APs by adding a record of the subfiles recovered by each F-AP in $\mathcal{U}^{\mathrm{a}}$ at the end of this time slot as its cache content. Note that updating the cache records has no influence on the cache contents of the F-APs, which stay unchanged in the delivery phase. The cache records is used to help the cloud server identify whether the subfile to be transmitted is $\varnothing$ in real time before transmission.

At the end of time slot B, all the requests of the F-APs in $\mathcal{U}^{\mathrm{a}}$ can be fulfilled together. Similarly, define $\underline{\chi}' = \max\{1, s + |\mathcal{U}^{\mathrm{a}}| - |\mathcal{K}|\}$ and $\overline{\chi}' = \min\{s, |\mathcal{U}^{\mathrm{a}}|\}$. For any $\mathcal{S}^1 \subseteq \mathcal{U}^{\mathrm{a}}$ of cardinality $\left|\mathcal{S}^1\right| = \chi$ and $\mathcal{S}^2 \subseteq \mathcal{K} \backslash \mathcal{U}^{\mathrm{a}}$ of cardinality $\left|\mathcal{S}^2\right| = s - \chi$ with any s and $\chi \in \left\{\underline{\chi}', \underline{\chi}' + 1, \cdots, \overline{\chi}'\right\}$, the cloud server transmits the coded-multicasting content as follows:

$$\oplus_{k \in \mathcal{S}^1} W_{k,(\mathcal{S}^1 \cup \mathcal{S}^2)\backslash\{k\}}^{\mathrm{a}},$$

where all the subfiles $W_{k,(\mathcal{S}^1 \cup \mathcal{S}^2)\backslash\{k\}}$ are assumed to be zero-padded to the number of bits of the longest subfile in the bit-wise XOR operation. After that, each F-AP in $\mathcal{U}^{\mathrm{a}}$ recovers the subfiles of its requested file, and then transmits the recovered subfiles and the subfiles available in its cache to its served user; thus the user can recover the desirable file.

2) Synchronous Transmission Method: When $\Delta b = B$, the synchronous transmission method is chosen. Firstly, no contents are transmitted at the end of time slot $1, 2, \cdots, B - 1$. At the end of time slot B, for all $\mathcal{S}$ with any s, the cloud server

Algorithm 2 The Proposed Asynchronous Coded Caching Scheme

```
PLACEMENT
for  $k \in \mathcal{K}, n \in \mathcal{N}$ do
    F-AP $k$ independently caches a subset of $MF/N$ bits of file $W_n$, chosen uniformly at random.
end for
------------------------------------------
DELIVERY
Initialize $\mathcal{U}^{\mathrm{a}} = \varnothing$, $b = 1$.
while $b \le B$ do
    if $\Delta b < B$ then
        if $b \le \Delta b - 1$ then
            $\mathcal{U}^{\mathrm{a}} = \mathcal{U}^{\mathrm{a}} \cup \mathcal{U}_b$,
            At the end of time slot $b$, no contents are transmitted.
        else if $\Delta b - 1 < b < B$ then
            $\mathcal{U}^{\mathrm{a}} = \mathcal{U}^{\mathrm{a}} \cup \mathcal{U}_b$.
            for $s = |\mathcal{K}|, |\mathcal{K}| - 1, \cdots, 1$ do
                for $\chi = \max\{1, s + |\mathcal{U}_{b-\Delta b+1}| - |\mathcal{K}|\} : \min\{s, |\mathcal{U}_{b-\Delta b+1}|\}$ do
                    for all $\mathcal{S}^1 \subseteq \mathcal{U}_{b-\Delta b+1}, \mathcal{S}^2 \subseteq \mathcal{K} \backslash \mathcal{U}_{b-\Delta b+1} : \left|\mathcal{S}^1\right| = \chi, \left|\mathcal{S}^2\right| = s - \chi$ do
                        At the end of time slot $b$, no contents are transmitted if $W^{\mathrm{a}}_{k,(\mathcal{S}^1 \cup \mathcal{S}^2)\backslash\{k\}} = \varnothing$ for $k \in \left(\left(\mathcal{S}^1 \cup \mathcal{S}^2\right) \cap \mathcal{U}^{\mathrm{a}}\right)$; Otherwise, the cloud server sends $\oplus_{k \in ((\mathcal{S}^1 \cup \mathcal{S}^2) \cap \mathcal{U}^{\mathrm{a}})} W^{\mathrm{a}}_{k,(\mathcal{S}^1 \cup \mathcal{S}^2)\backslash\{k\}}$.
                    end for
                end for
            end for
            $\mathcal{U}^{\mathrm{a}} = \mathcal{U}^{\mathrm{a}} \backslash \mathcal{U}_{b-\Delta b+1}$;
        else
            $\mathcal{U}^{\mathrm{a}} = \mathcal{U}^{\mathrm{a}} \cup \mathcal{U}_b$.
            for $s = |\mathcal{K}|, |\mathcal{K}| - 1, \cdots, 1$ do
                for $\chi = \max\{1, s + |\mathcal{U}^{\mathrm{a}}| - |\mathcal{K}|\} : \min\{s, |\mathcal{U}^{\mathrm{a}}|\}$ do
                    for all $\mathcal{S}^1 \subseteq \mathcal{U}^{\mathrm{a}}, \mathcal{S}^2 \subseteq \mathcal{K} \backslash \mathcal{U}^{\mathrm{a}} : \left|\mathcal{S}^1\right| = \chi, \left|\mathcal{S}^2\right| = s - \chi$ do
                        At the end of time slot $B$, the cloud server sends $\oplus_{k \in \mathcal{S}^1} W^{\mathrm{a}}_{k,(\mathcal{S}^1 \cup \mathcal{S}^2)\backslash\{k\}}$.
                    end for
                end for
            end for
        end if
    else
        if $b \le B - 1$ then
            At the end of time slot $b$, no contents are transmitted.
        else
            for $s = |\mathcal{K}|, |\mathcal{K}| - 1, \cdots, 1$ do
                for all $\mathcal{S} \subseteq \mathcal{K} : |\mathcal{S}| = s$ do
                    At the end of time slot $B$, the cloud server sends $\oplus_{k \in \mathcal{S}} W^{\mathrm{a}}_{k,\mathcal{S}\backslash\{k\}}$.
                end for
            end for
        end if
    end if
    $b = b + 1$.
end while
```

transmits the coded-multicasting content as follows:

$$\oplus_{k \in \mathcal{S}} W^{\mathrm{a}}_{k,\mathcal{S}\backslash\{k\}}.$$

Then, each F-AP transmits all the subfiles of its requested file to its served user; thus the user can recover the desirable file.

The detailed description of our proposed decentralized asynchronous coded caching scheme is presented in Algorithm 2. Note that χ is used to ensure that $\mathcal{S}^1 \cap \mathcal{S} \neq \varnothing$, so that the coded-multicasting content transmitted by the cloud server for $\mathcal{S}$ is useful to at least one F-AP in $\mathcal{U}_{b-\Delta b+1}$ or $\mathcal{U}^{\mathrm{a}}$. Also note that the problem setting allows for vanishing probability of error as $F \to \infty$.

Example 1: Assume that $N = 4$, $K = 4$, $M = 2$, $B = 4$, $T = 4$ s, $\Delta t = 1$ s, $\Delta b = 2$, $\mathcal{U}_b = \{b\}$ and $d_k = k$ in asynchronous coded caching. It is easy to see that this is the worst-case request. According to Algorithm 2, the coded-multicasting contents transmitted by the cloud server at the end of time slot 2, 3, and 4 are illustrated in Table I, Table II, and Table III, respectively. Note that $\varnothing$ indicates no contents are transmitted in the tables. In addition, subfile $W^{\mathrm{a}}_{3,\{2,4\}}$ is actually $\varnothing$ according to the updated cache records.

TABLE I. The contents transmitted during time slot 2

s	χ	$\mathcal{S}^1$	$\mathcal{S}^2$	$\mathcal{U}^{\mathrm{a}}$	Coded-multicasting Content
4	1	$\{1\}$	$\{2,3,4\}$	$\{1,2\}$	$W^{\mathrm{a}}_{1,\{2,3,4\}} \oplus W^{\mathrm{a}}_{2,\{1,3,4\}}$
3	1	$\{1\}$	$\{2,3\}$	$\{1,2\}$	$W^{\mathrm{a}}_{1,\{2,3\}} \oplus W^{\mathrm{a}}_{2,\{1,3\}}$
3	1	$\{1\}$	$\{2,4\}$	$\{1,2\}$	$W^{\mathrm{a}}_{1,\{2,4\}} \oplus W^{\mathrm{a}}_{2,\{1,4\}}$
3	1	$\{1\}$	$\{3,4\}$	$\{1,2\}$	$W^{\mathrm{a}}_{1,\{3,4\}}$
2	1	$\{1\}$	$\{2\}$	$\{1,2\}$	$W^{\mathrm{a}}_{1,\{2\}} \oplus W^{\mathrm{a}}_{2,\{1\}}$
2	1	$\{1\}$	$\{3\}$	$\{1,2\}$	$W^{\mathrm{a}}_{1,\{3\}}$
2	1	$\{1\}$	$\{4\}$	$\{1,2\}$	$W^{\mathrm{a}}_{1,\{4\}}$
1	1	$\{1\}$	$\varnothing$	$\{1,2\}$	$W^{\mathrm{a}}_{1,\varnothing}$

Consider the same setting as Example 1. Now, we explain how Algorithm 2 implements our proposed encoding set partition method. Focus on $\mathcal{S} = \{1,2,3,4\}$. Firstly, no contents are transmitted at the end of time slot 1. At the end of time slot 2, $W^{\mathrm{a}}_{1,\{2,3,4\}} \oplus W^{\mathrm{a}}_{2,\{1,3,4\}}$ is transmitted with $\mathcal{U}^{\mathrm{a}} = \{1,2\}$. At the end of time slot 3, the cloud server decides not to transmit $W^{\mathrm{a}}_{2,\{1,3,4\}} \oplus W^{\mathrm{a}}_{3,\{1,2,4\}}$ with $\mathcal{U}^{\mathrm{a}} = \{2,3\}$, since $W^{\mathrm{a}}_{2,\{1,3,4\}}$ is $\varnothing$ according to the updated cache records. Thus, no contents are transmitted. Finally, $W^{\mathrm{a}}_{3,\{1,2,4\}} \oplus W^{\mathrm{a}}_{4,\{1,2,3\}}$ is transmitted with $\mathcal{U}^{\mathrm{a}} = \{3,4\}$ at the end of time slot 4. Observe that $W^{\mathrm{s}}_{1,\{2,3,4\}} \oplus W^{\mathrm{s}}_{2,\{1,3,4\}} \oplus W^{\mathrm{s}}_{3,\{1,2,4\}} \oplus W^{\mathrm{s}}_{4,\{1,2,3\}}$ is partitioned into two parts of equal size, i.e., $W^{\mathrm{a}}_{1,\{2,3,4\}} \oplus W^{\mathrm{a}}_{2,\{1,3,4\}}$ and $W^{\mathrm{a}}_{3,\{1,2,4\}} \oplus W^{\mathrm{a}}_{4,\{1,2,3\}}$, for transmission in our proposed asynchronous coded caching scheme.

Remark 1: The main innovation of our proposed scheme is to partition the coded-multicasting contents in synchronous coded caching by using our proposed encoding set partition method, and then transmit the partitioned contents at the end of different time slots. Our proposed scheme can create as many coded-multicasting opportunities as possible while the maximum request delay of each F-AP is no more than Δb time slots.

Remark 2: The asynchronous coded caching problem has been considered in [10]. The authors described their proposed approach based on a system using the centralized synchronous coded caching scheme in [4], while they declared their approach can be applied to general placement schemes. In this paper, we propose a decentralized asynchronous coded caching scheme based on a different system model for the online case, which is more applicable for practical scenarios. Moreover, our proposed scheme can work well for both the online case and offline case.

IV. Simulation Results

In this section, the performance of our proposed decentralized asynchronous coded caching scheme is evaluated via simulations. We adopt the Maddah-Ali-Niesen's decentralized synchronous coded caching scheme and the uncoded caching

TABLE II. The contents transmitted during time slot 3

s	χ	$\mathcal{S}^1$	$\mathcal{S}^2$	$\mathcal{U}^{\mathrm{a}}$	Coded-multicasting Content
4	1	$\{2\}$	$\{1,3,4\}$	$\{2,3\}$	$\varnothing$
3	1	$\{2\}$	$\{1,3\}$	$\{2,3\}$	$\varnothing$
3	1	$\{2\}$	$\{1,4\}$	$\{2,3\}$	$\varnothing$
3	1	$\{2\}$	$\{3,4\}$	$\{2,3\}$	$W^{\mathrm{a}}_{2,\{3,4\}} \oplus W^{\mathrm{a}}_{3,\{2,4\}}$
2	1	$\{2\}$	$\{1\}$	$\{2,3\}$	$\varnothing$
2	1	$\{2\}$	$\{3\}$	$\{2,3\}$	$W^{\mathrm{a}}_{2,\{3\}} \oplus W^{\mathrm{a}}_{3,\{2\}}$
2	1	$\{2\}$	$\{4\}$	$\{2,3\}$	$W^{\mathrm{a}}_{2,\{4\}}$
1	1	$\{2\}$	$\varnothing$	$\{2,3\}$	$W^{\mathrm{a}}_{2,\varnothing}$

TABLE III. The contents transmitted during time slot 4

s	χ	$\mathcal{S}^1$	$\mathcal{S}^2$	$\mathcal{U}^{\mathrm{a}}$	Coded-multicasting Content
4	2	$\{3,4\}$	$\{1,2\}$	$\{3,4\}$	$W^{\mathrm{a}}_{3,\{1,2,4\}} \oplus W^{\mathrm{a}}_{4,\{1,2,3\}}$
3	1	$\{3\}$	$\{1,2\}$	$\{3,4\}$	$W^{\mathrm{a}}_{3,\{1,2\}}$
3	1	$\{4\}$	$\{1,2\}$	$\{3,4\}$	$W^{\mathrm{a}}_{4,\{1,2\}}$
3	2	$\{3,4\}$	$\{1\}$	$\{3,4\}$	$W^{\mathrm{a}}_{3,\{1,4\}} \oplus W^{\mathrm{a}}_{4,\{1,3\}}$
3	2	$\{3,4\}$	$\{2\}$	$\{3,4\}$	$W^{\mathrm{a}}_{3,\{2,4\}}\,(\varnothing) \oplus W^{\mathrm{a}}_{4,\{2,3\}}$
2	1	$\{3\}$	$\{1\}$	$\{3,4\}$	$W^{\mathrm{a}}_{3,\{1\}}$
2	1	$\{3\}$	$\{2\}$	$\{3,4\}$	$\varnothing$
2	1	$\{4\}$	$\{1\}$	$\{3,4\}$	$W^{\mathrm{a}}_{4,\{1\}}$
2	1	$\{4\}$	$\{2\}$	$\{3,4\}$	$W^{\mathrm{a}}_{4,\{2\}}$
2	2	$\{3,4\}$	$\varnothing$	$\{3,4\}$	$W^{\mathrm{a}}_{3,\{4\}} \oplus W^{\mathrm{a}}_{4,\{3\}}$
1	1	$\{3\}$	$\varnothing$	$\{3,4\}$	$W^{\mathrm{a}}_{3,\varnothing}$
1	1	$\{4\}$	$\varnothing$	$\{3,4\}$	$W^{\mathrm{a}}_{4,\varnothing}$

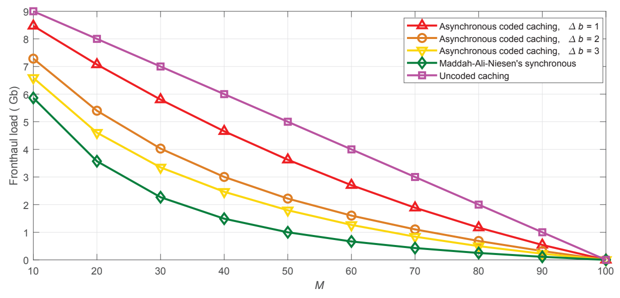

Fig. 1. Fronthaul load versus M.

scheme as the baseline schemes. In our simulations, the number of the F-APs whose requests arrive during time slot b, i.e., $|\mathcal{U}_b|$, is random. Other parameters are set as follows: $F = 1$ Gb, $N = 100$, $K = 10$, $T = 10$ s, $B = 5$.

In Fig. 1, we show the effect of the normalized cache size of each F-AP, i.e., M, on the fronthaul load of each scheme for different Δb. As shown, our proposed scheme can create considerable coded-multicasting opportunities compared with the uncoded caching scheme. Moreover, the fronthaul load decreases and its slope increases when M increases, which is the same as the Maddah-Ali-Niesen's decentralized synchronous coded caching scheme.

In Fig. 2, we show how Δb affects the fronthaul load of each scheme for varying cache sizes. As shown, the fronthaul load of our proposed scheme decreases when Δb increases, which means that our proposed scheme can create more coded-multicasting opportunities with a relaxed delay requirement. Furthermore, the larger Δb is, the more the decrease of the

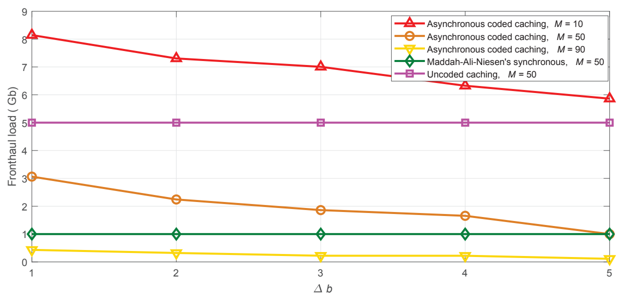

Fig. 2. Fronthaul load versus Δb for varying cache sizes.

fronthaul load of our proposed scheme is when compared with that of the uncoded caching scheme. And the performance gap between the fronthaul load of our proposed scheme and that of the Maddah-Ali-Niesen's decentralized synchronous coded caching scheme is smaller when Δb is larger. The reason for the above results is that a larger Δb leads to a fewer number of the partitioned subsets. Moreover, as Δb determines the upper bound of the request delay, it can be set to a relatively small value in delay-sensitive scenarios and adjusted flexibly to achieve the load-delay tradeoff in other scenarios.

V. Conclusions

In this paper, we have proposed a decentralized asynchronous coded caching scheme for the online case in F-RAN where users asynchronously request contents with the maximum request delay. Our proposed scheme provides asynchronous and synchronous transmission methods to fulfill the delay requirements of different practical scenarios. The simulation results have shown that more coded-multicasting opportunities can be created when the maximum request delay increases in asynchronous request scenarios.

Acknowledgments

This work was supported in part by the Research Fund of the State Key Laboratory of Integrated Services Networks (Xidian University) under grant ISN19-10, the Research Fund of the Key Laboratory of Wireless Sensor Network & Communication (Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences) under grant 2017002, the Hong Kong, Macao and Taiwan Science & Technology Cooperation Program of China under grant 2014DFT10290, the Ericsson and SEU Cooperation Project under grant 8504000335, the National Basic Research Program of China (973 Program) under grant 2012CB316004, and the U.K. Engineering and Physical Sciences Research Council under Grant EP/K040685/2.

References

[1] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, Aug. 2016.

[2] 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.

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

[4] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[5] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, Aug. 2015.

[6] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Transactions on Information Theory*, vol. 63, no. 2, pp. 1146–1158, Feb. 2017.

[7] J. Hachem, N. Karamchandani, and S. Diggavi, "Multi-level coded caching," in *2014 IEEE International Symposium on Information Theory (ISIT)*, June 2014, pp. 56–60.

[8] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3923–3949, June 2017.

[9] M. A. Maddah-Ali and U. Niesen, "Coding for caching: Fundamental limits and practical challenges," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 23–29, Aug. 2016.

[10] H. Ghasemi and A. Ramamoorthy, "Asynchronous coded caching," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 2438–2442.