Dynamic Resource Scheduling in Cloud Radio Access Network with Mobile Cloud Computing

Xinhou Wang[1], Kezhi Wang[2], Song Wu*[1], Sheng Di[3], Kun Yang[2], Hai Jin*[1]
[1]Services Computing Technology and System Lab, Cluster and Grid Computing Lab
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China
[2]University of Essex, UK [3]Argonne National Laboratory, USA
E-mails: [1]{xwang, wusong, hjin}@hust.edu.cn [2]{kezhi.wang, kunyang}@essex.ac.uk [3]sdi1@anl.gov

***Abstract*—Nowadays, by integrating the *cloud radio access network* (C-RAN) with the *mobile cloud computing* (MCC) technology, *mobile service provider* (MSP) can efficiently handle the increasing mobile traffic and enhance the capabilities of mobile users' devices to provide better *quality of service* (QoS). But the power consumption has become skyrocketing for MSP as it gravely affects the profit of MSP. Previous work often studied the power consumption in C-RAN and MCC separately while less work had considered the integration of C-RAN with MCC. In this paper, we present a unifying framework for optimizing the power-performance tradeoff of MSP by jointly scheduling network resources in C-RAN and computation resources in MCC to minimize the power consumption of MSP while still guaranteeing the QoS for mobile users. Our objective is to maximize the profit of MSP. To achieve this objective, we first formulate the resource scheduling issue as a stochastic problem and then propose a *R*esource onl*I*ne s*CH*eduling (*RICH*) algorithm using Lyapunov optimization technique to approach a time average profit that is close to the optimum with a diminishing gap (1/V) for MSP while still maintaining strong system stability and low congestion to guarantee the QoS for mobile users. With extensive simulations, we demonstrate that the profit of *RICH* algorithm is 3.3× (18.4×) higher than that of *active* (*random*) algorithm.**

I. Introduction

Nowadays, the existing cellular network is facing the pressure to increase the capacity so as to meet the increasing number of smart devices and the corresponding mobile traffic demand [3]. With the increasing requests from *user equipments* (UEs), *mobile service providers* (MSPs) need to build more *base station* (BS) sites. However, in an intensifying competitive marketplace and rapid technological changes, MSPs are challenged with deployment of traditional BS as the cost (*capital expenditures* (CAPEX) and *operational expenditures* (OPEX)) is high, while the return (revenue gained by the increasing requests) is not high enough [6].

To address this challenge, *cloud radio access network* (C-RAN) has been proposed and soon received significant attention in both academia and industry [6], [12]. Unlike typical RANs where the *baseband units* (BBUs) and the radio units are situated together, C-RAN is a cloud computing based, centralized, clean and collaborative RAN [6]. C-RAN divides the traditional BS into three parts, namely, serval *remote radio heads* (RRHs), BBU pool, and the fronthaul link connecting RRH to the BBU pool which is a high-bandwidth, high-speed, low latency fiber transport link. However, more and more resource-hungry applications such as multimedia applications and gaming appear in our daily life, which gives resource-constrained and battery-limited UEs much pressure [4]. *Mobile cloud computing* (MCC) is envisioned as a promising approach to address such a challenge [8].

By integrating the C-RAN with the MCC technology, MSP can not only handle the increasing mobile traffic by using C-RAN technology, but also enhance the capabilities of mobile devices with the powerful mobile cloud platforms. Although some excellent works have been done to study both C-RAN [1], [12] and MCC [2], [10], these two important areas have traditionally been addressed separately in the literature. The research of integration of C-RAN with MCC is rarely less. Fortunately, some works [14], [15] have shown that the combination of MCC and C-RAN can provide better *quality of service* (QoS) for mobile users. Therefore, it is necessary to consider these two technologies together.

Considering a typical mobile system which consists of C-RAN and mobile clouds in Fig. 1. In C-RAN, each RRH serves and receives requests from a couple of UEs that are close to this RRH. Mobile UEs are charged for each received requests. The RRHs are connected to the BBU pool via a fronthaul network which consumes power to transmit requests. All the BBUs are aggregated together to form a BBU pool, in which a *Despatcher* is used to despatch requests across several mobile clouds. Each mobile cloud has multiple servers which consume power to process requests from different UEs.

It has been widely acknowledged that electricity cost of power consumption becomes skyrocketing for MSP [6]. For example, China Mobile has to spend more than one billion dollars for the electricity cost every year [6]. Hence, a facing problem of MSP is to minimize the power consumption of the whole system. Moreover, due to the mobility of mobile UEs, the arrival of requests from mobile UEs are always unpredictable [9], which may lead to fluctuating revenues for

*The Corresponding Authors are Song Wu and Hai Jin. The research was supported by National Science Foundation of China under grant 61232008, National 863 Hi-Tech Research and Development Program under grant 2015AA01A203 and 2014AA01A302, Chinese Universities Scientific Fund under grant 2013TS094 and also supported partially by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research Program, under Contract DE-AC02-06CH11357.

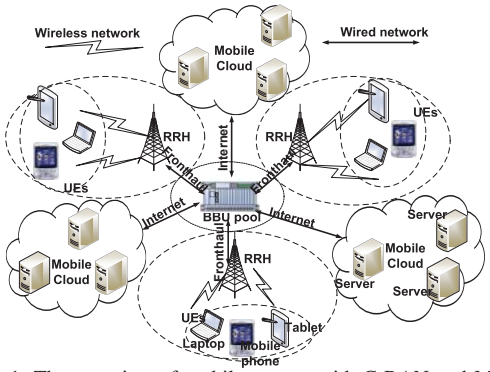

Fig. 1: The overview of mobile system with C-RAN and MCC

MSP over time.

Therefore, with the presence of unpredictable requests from mobile users and the skyrocketing electricity cost of power consumption, the objective of the mobile system is to maximize the profit of MSP. We need to optimize such a trade-off between performance and power to minimize the power consumption of MSP while still guaranteeing the QoS for mobile users, the mobile system needs to tackle the following scheduling challenges: (1) how to schedule each fronthaul link by turning to an ON state for transmitting requests into the BBU pool and an OFF state to decline users' requests for fronthaul power conservation; (2) how to despatch the received requests from different users to its corresponding servers in different mobile clouds; (3) how to schedule each server by switching to a running state for processing requests or an idle state for server power conservation.

To address the above-mentioned challenges, in this paper, we apply the Lyapunov optimization technique [11] to design a *R*esource onl*I*ne s*CH*eduling algorithm (*RICH*) which schedules the fronthaul links, BBU *Despatcher* and servers independently and concurrently, solely based on the current system state. Therefore, it is attractive for large-scale mobile system and is feasible for online implementation. Our main contributions can be summarized as follows:

- We present a unifying optimization framework for maximizing the profit of MSP who manages both network system (C-RAN) and computing system (MCC).
- We design the *RICH* algorithm for joint optimization of fronthaul links scheduling, requests despatching and servers scheduling, which can efficiently handle the unpredictable and time-varying mobile user requests. These requests can be unpredictable and time-varying due to the mobility of mobile users.
- With extensive simulations, we demonstrate that our algorithm can approach a time average profit that is close to the optimum with a diminishing gap (1/V) for MSP while still maintaining strong system stability and low congestion to guarantee the QoS for mobile users.

II. Power-Performance Model

A. System Architecture

The mobile system architecture includes two parts, i.e., C-RAN and mobile clouds. In our system, there are M RRHs distributed in different geographic small cells everywhere. Each RRH i serves and receives requests from a set of UEs that are close this RRH. Such a set of UEs is denoted as a *representative* user [13]. Accordingly, the mobile system has M users $\mathcal{U} \triangleq \{1, 2, \cdots, M\}$. In this paper, we consider a discrete time-slotted system where the time slot length varies from hundreds of milliseconds to several minutes [17]. In every time slot t $(= 0, 1, 2, \cdots)$, all UEs in U_i can send their requests to the RRH i. The system aggregates all requests received from all UEs by RRH i as $A_i(t)$ and the time average rate of such an arrival process can be denoted as $\lambda_i = \mathbb{E}\{A_i(t)\}$.

Similar to previous work in mobile networking [14], we consider a quasi-static scenario where the UEs remain unchanged during a time slot. Hence, we can assume that UEs served by one RRH will not influence UEs served by another RRH and each variable $A_i(t)$ is independent and identically distributed (i.i.d.) over time slots. Without loss of generality, we also assume $A_i(t) \leq A_i^{max}$, where A_i^{max} is the maximum request of the wireless network between UEs and the RRH i due to the bandwidth capacity. Since UEs have its own mobility [9] and the requests are dynamic and unpredictable, we do not assume any a *priori* knowledge of the statistics of $A_i(t), \forall i \in \mathcal{U}, \forall t$.

The RRHs are connected to the BBU pool via a fronthaul network which consumes power to transmit requests. Inspired by [5], we simply assume the i-th fronthaul constraint as the maximum requests in one time slot, i.e., $C_i \leq C_i^{max}$.

In mobile cloud, N clouds $\mathcal{C} \triangleq \{1, 2, \cdots, N\}$ are geographically located all over the country. Each mobile cloud $j, \forall j \in \mathcal{C}$ has large number of servers which process the requests transmitted from the *Despatcher*. We assume that each cloud launches a server i to process requests from U_i.

We summarize the key notations in Table I.

TABLE I: Key Notations

Notation	Description
$\mathcal{U}$	all representative users U_i
$\mathcal{C}$	all mobile clouds C_i
M	number of users
N	number of clouds
$A_i(t)$	arrival requests for RRH i at time slot t
λ_i	time average rate of $A_i(t)$
$a_i(t)$	fronthaul scheduling policy
$R_i(t)$	requests transmitted by fronthaul link i at time slot t
$D_{ij}(t)$	requests despatched from user i to mobile cloud j
$X_i(t)$	queue backlog of buffer queue for users i
$b_{ij}(t)$	server scheduling policy in mobile clouds
$Q_{ij}(t)$	queue backlog of each server i in each mobile cloud j
p_i^f	time average power consumption of fronthaul link i
p_j^s	time average power consumption of cloud j
V	control parameter in Lyapunov technique
α_i	non-negative normalized parameter for U_i
β	non-negative normalized parameter for fronthaul link
γ	non-negative normalized parameter for mobile cloud
ω	normalized power consumption of a idle server

B. Dynamic Scheduling

Under the architecture of mobile system above, the system aims to maximize the profit of MSP by scheduling fronthaul links *Despatcher* in the BBU pool and servers in the mobile clouds.

Fronthaul Scheduling: In every time slot t, the mobile system needs to determine a subset of requests of each user $R_i(t)$, that can be transmitted into the BBU pool through the fronthaul links: $0 \leq R_i(t) \leq A_i(t)$. The fronthual scheduling policy is to schedule each fronthaul link by tuning to an ON (OFF) state for transmitting (declining) requests from the RRH i to the BBU pool. Such fronthaul scheduling policies $a_i(t)$ are denoted as the l_0-*norm* of $R_i(t)$ (i.e., $a_i(t) = ||R_i(t)||_0$), which can be indicated by the following function:

$$a_i(t) = ||R_i(t)||_0 = \begin{cases} 1 & \text{fronthaul link } i \text{ is ON, } R_i(t) > 0 \\ 0 & \text{fronthaul link } i \text{ is OFF, } R_i(t) = 0 \end{cases}$$

The transferred requests $R_i(t)$ need to satisfy this constraint: $R_i(t) \leq a_i(t) C_i^{max}$, where C_i^{max} refers to as the capacity limitation of fronthaul link i.

BBU-based Requests Despatching: The *Despatcher* in the BBU pool will despatch received requests to the corresponding server hosted in clouds. We assume that the amount of admitted requests $R_i(t)$ are queued in a buffer queue for each user i in the BBU pool before despatching to the corresponding queue for each server in the mobile clouds. Let $X_i(t)$ denotes the backlog of this buffer queue i at time slot t with $X_i(0) = 0, \forall i \in \mathcal{U}$. Also let $D_{ij}(t)$ denotes the requests despatched from user i to cloud j. Then the *queueing dynamics* [11] can be characterized by:

$$X_i(t+1) = \max\{X_i(t) - \sum_{j=1}^{N} D_{ij}(t), 0\} + R_i(t) \qquad (1)$$

Intuitively, the despatching decisions $D_{ij}(t)$ must satisfy $\sum_{j=1}^{N} D_{ij}(t) \leq X_i(t)$.

Cloud Server Scheduling: The last scheduling policy is to schedule each server by switching the server to an *idle* state to keep the requests waiting in this server's queue, or resuming the server to a *running* state to process requests that are waiting in this server' queue [1]. The following indicator function shows such a server scheduling policy:

$$b_{ij}(t) = \begin{cases} 1 & \text{if server } i \text{ on cloud } j \text{ is running.} \\ 0 & \text{if server } i \text{ on cloud } j \text{ is idle.} \end{cases}$$

We first assume that each server from cloud j can process B_j requests in one time slot and we can assume there exist a maximum level B^{max} for all servers in all clouds, i.e., $B_j \leq B^{max}$. Then, let $Q_{ij}(t)$ denotes as the queue backlog of server i on cloud j which means the total number of requests that are waiting in the queue at the beginning of time slot t with $Q_{ij}(0) = 0$. Also, we can quantify $b_{ij}(t)B_j$ as the service rate of the queue and $D_{ij}(t)$ as the arrival rate. By doing so, we have the following *queuing dynamics* [11]:

$$Q_{ij}(t+1) = \max\{Q_{ij}(t) - b_{ij}(t)B_j, 0\} + D_{ij}(t) \qquad (2)$$

C. Power-Performance Tradeoff

After designing the scheduling policies above, we then derive the system throughput and the power consumption incurred by fronthaul links and servers. Intuitively, the throughput and power consumption are contradictory to each other. We present a unifying optimization framework for profit maximization of MSP.

1) Time Average Throughput: In the mobile system, the overall system throughput in terms of the total number of processing requests is one of the most significant performance metrics. Specially, we define the time average throughput r_i for each UE set U_i as $r_i = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{R_i(\tau)\}$.

Together with r_i, we define the time average transmission capacity a_i for each fronthaul link i as $a_i = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{a_i(\tau)\}$, which is the frequency to turn fronthual link i to ON state. Similarly, we define the time average consumed capacity b_{ij} for each server i in cloud j as $b_{ij} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{b_{ij}(\tau)\}$, which is the frequency to run server i in mobile cloud j.

2) Time Average Power Consumption: We analyze two power consumption models in this part including the power consumption of the fronthaul and the power consumption of servers in mobile clouds.

For the power of fronthaul, it consumes a constant power once it opens (i.e., ON state) [5]. Without loss of generality, we consider a normalized power consumption $P^f(\mu) = \mu \in \{0, 1\}$, where $\mu = 0$ $(= 1)$ represents the OFF (ON) state of a fronthaul link. Based on this fronthaul power model, for a fronthaul link $i \in \mathcal{U}$, its normalized power consumption in time slot t is given as $P_i^f(t) = P_i^f(a_i(t)) = a_i(t)$.

Accordingly, the time average of normalized power consumption p_i^f of each fronthaul link $\forall i \in \mathcal{U}$ in C-RAN can be defined as $p_i^f = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{P_i^f(\tau)\}$. Then, $\sum_{i=1}^{M} p_i^f$ is the overall time average power consumption of all fronthaul links.

For the power of server, it has been widely studied [17] that the amount of power consumed by a server is primarily associated with its current CPU running speed μ. In this paper, we employ a very basic power consumption model [17] of servers as $P^s(\mu) = \omega\mu^v + (1-\omega)$. Without loss of generality, a normalized speed $0 \leq \mu \leq 1$ and its corresponding normalized power consumption $P^s(\mu)$ are considered in this paper. Intuitively, the server stays idle when $\mu = 0$ and running with maximum CPU speed $\mu = 1$. The parameter v is empirically determined as $v \geq 1$ in practical [17]. With another parameter $0 \leq \omega \leq 1$, the term $1 - \omega$ represents the normalized power consumption of an idle server.

Based on the above power consumption model, the normalized load and power consumption for cloud j are given as follows,

$$\mu_j(t) = \sum_{i=1}^{M} b_{ij}(t)/M,$$

$$P_j^s(t) = \omega(\mu_j(t))^v + (1-\omega).$$

Accordingly, the time average of normalized power consumption of each cloud j in the mobile system can be defined as $p_j^s = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{P_j^s(\tau)\}$. Then, $\sum_{j=1}^{N} p_j^s$ is the overall time average power consumption of all clouds.

3) Time Average Profit Maximization: Now, we define our scheduling objective as the MSP's time average profit which

[1] Similar to [17], we only switch the server between running and idle state, rather than frequently turning ON/OFF servers per time slot, which would incur considerable overhead [7] (e.g., startup time). This overhead can be longer than the time slot used in this paper.

includes the time average throughput revenue $\bar{e_t} = \sum_{i=1}^{M} \alpha_i r_i$, the time average fronthaul electricity cost $\bar{e_f} = \sum_{i=1}^{M} \beta p_i^f$ and the time average server electricity cost $\bar{e_s} = \sum_{j=1}^{N} \gamma p_j^s$. The parameters α_i, β and γ are presented in Table I.

Given the above time average revenue brought by the throughput and cost for both fronthaul and server power consumption, we formulate the maximization of time average profit as the following *stochastic optimization* problem:

$$\begin{aligned} \mathcal{P}: \quad & \max \quad \bar{e_t} - \bar{e_f} - \bar{e_s}, \\ & s.t. \quad 0 \le r_i \le \lambda_i, r_i \le a_i C_i^{max}, \\ & \quad r_i \le \textstyle\sum_{j=1}^{N} b_{ij} B_j. \end{aligned}$$

III. Online Algorithm

Since UEs always have their own mobility [9] and the arrival requests are unpredictable, we cannot get the future information about UEs' requests. Our considerations on power consumption and queue stability lead us to design a resource online scheduling algorithm (i.e., *RICH*) based on the Lyapunov optimization framework [11], which has been widely used in power consumption optimization problem [17].

A. Problem Transformation Using Lyapunov Optimization

1) Characterizing the Stability-Profit Tradeoff: Let $\mathbf{Q}(t) = (Q_{ij}(t))$ and $\mathbf{X}(t) = (X_i(t))$ denote the matrixes of the actual queues maintained by servers and the buffer queues for UEs sets in the BBU pool. After that, we can use $\mathbf{\Theta}(t) = [\mathbf{Q}(t); \mathbf{X}(t)]$ to denote the combined matrix of all queues and define a Lyapunov function $L(\mathbf{\Theta}(\mathbf{t}))$ as follows:

$$L(\mathbf{\Theta}(\mathbf{t})) = \frac{1}{2}\{\sum_{i \in \mathcal{U}} X_i^2(t) + \sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{C}} Q_{ij}^2(t)\}. \tag{3}$$

This function represents a scalar metric of congestion [11] for the mobile cloud. Intuitively, a small value of $L(\mathbf{\Theta})$ suggests that all the queue backlogs are small, i.e., the corresponding mobile system has *strong stability*.

Based on Eq. 3, we need to push the Lyapunov function towards a lower congestion state by defining the conditional *1-slot Lyapunov drift* [11] as follows:

$$\Delta(\mathbf{\Theta}(t)) = \mathbb{E}\{L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t))|\mathbf{\Theta}(t)\} \tag{4}$$

Under the Lyapunov optimization, the scheduling policies $a_i(t)$, $D_{ij}(t)$ and $b_{ij}(t)$ should be chosen to minimize the infimum bound on the following drift-minus-profit [11]:

$$\begin{aligned} &\Delta(\mathbf{\Theta}(t)) - V\mathbb{E}\{\sum_{i \in \mathcal{U}} \alpha_i R_i(t) \\ &-\beta \sum_{i \in \mathcal{U}} P_i^f(t) - \gamma \sum_{j \in \mathcal{C}} P_j^s(t)|\mathbf{\Theta}(t)\} \end{aligned} \tag{5}$$

The parameter $V \ge 0$ represents a *design knob* that is used to balance *the tradeoff between the profit maximization and the drift*. For example, a large value of V implies that the mobile system prefers to achieve more profit rather than keep the system queue backlogs at a low level.

2) Bounding the Drift-Minus-Profit: To derive the infimum bound of the *drift-minus-profit* given in Eq. 5, the following Lemma is needed.

Lemma 1. *For any time slot t, given any scheduling policies, the* drift-minus-profit *(Eq. 5) can be deterministically bounded as follows:*

$$\begin{aligned} &\Delta(\mathbf{\Theta}(t)) - V\mathbb{E}\{\sum_{i=1}^{M} \alpha_i R_i(t) \\ &-\beta \sum_{i=1}^{M} P_i^f(t) - \gamma \sum_{j=1}^{N} P_j^s(t)|\mathbf{\Theta}(t)\} \le B \end{aligned}$$

$$-\sum_{i=1}^{M} \mathbb{E}\{R_i(t)(V\alpha_i - X_i(t)) - V\beta a_i(t)|\mathbf{\Theta}(t)\} \tag{6}$$

$$-\sum_{i=1}^{M} \sum_{j=1}^{N} \mathbb{E}\{D_{ij}(t) B_j (X_i(t) - Q_{ij}(t))|\mathbf{\Theta}(t)\} \tag{7}$$

$$-\sum_{j=1}^{N} \mathbb{E}\{B_j \sum_{i=1}^{M} Q_{ij}(t) b_{ij}(t) - V\gamma P_j^s(t)|\mathbf{\Theta}(t)\} \tag{8}$$

where $B = \frac{1}{2}[MN(B^{max})^2 + 3\sum_{i=1}^{M}(max\{A_i^{max}, C_i^{max}\})^2]$. *Proof (see Lemma 4.6 in [11]).*

B. Optimal Resource Online Scheduling Algorithm (RICH)

In this subsection, we design an optimal resource online scheduling algorithm, *RICH*, to minimize the infimum bound in Lemma 1 by equivalently maximizing the terms (6)(7)(8) on the right-hand-side in each time slot t.

1) Fronthaul Scheduling: The fronthaul scheduling policies $a_i(t)$ can be decided by maximizing the term (6) in Lemma 1. Recall that different UEs set served by different RRHs cannot influence each other in our system (see Sec. II). Therefore, the fronthaul scheduling policies $a_i(t)$ for different U_i are independent which means that the maximization of (6) can be decomposed to compute the following sub-problem ($\mathcal{SP}1$) concurrently.

$$\begin{aligned} \mathcal{SP}1: \quad & \max_{R_i(t), a_i(t)} \quad R_i(t)(V\alpha_i - X_i(t)) - V\beta a_i(t) \\ & s.t. \quad 0 \le R_i(t) \le A_i(t), a_i(t) = ||R_i(t)||_0, \\ & \quad R_i(t) \le a_i(t) C_i^{max}. \end{aligned}$$

The *non-convex l_0-norm* problem $V\beta a_i(t) = V\beta||R_i(t)||_0$ can be solved by applying the *l_1-norm relaxation* [16] technique. We have the following relaxed Problem ($\mathcal{SP}2$)

$$\begin{aligned} \mathcal{SP}2: \quad & \max_{R_i(t)} \quad R_i(t)(V\alpha_i - V\beta - X_i(t)) \\ & s.t. \quad 0 \le R_i(t) \le A_i(t), R_i(t) \le C_i^{max}. \end{aligned}$$

The Problem ($\mathcal{SP}2$) is a simple linear programming problem and we can derive the optimal value of $R_i(t)$ as:

$$R_i(t) = \begin{cases} min\{A_i(t), C_i^{max}\}, & X_i(t) < V\alpha_i - V\beta, \\ 0, & \text{else} \end{cases} \tag{9}$$

then we can have the fronthaul scheduling policies as:

$$a_i(t) = ||R_i(t)||_0 = \begin{cases} 1, & X_i(t) < V\alpha_i - V\beta, \\ 0, & \text{else} \end{cases} \tag{10}$$

2) BBU-based Requests Despatching: The BBU-based despatching policies $D_{ij}(t)$ can be decided by maximizing the term (7) in Lemma 1. Similar to the fronthaul scheduling policies, the requests despatching policies $D_{ij}(t)$ of different U_i are also independent which means that the maximization of

(7) can be decomposed to compute the following sub-problem ($\mathcal{SP}3$) concurrently.

$$\mathcal{SP}3: \quad \max_{D_{ij}(t)} \quad \sum_{j=1}^{N} D_{ij}(t)(X_i(t) - Q_{ij}(t))$$
$$s.t. \quad 0 \leq \sum_{j=1}^{N} D_{ij}(t) \leq X_i(t).$$

The above Problem ($\mathcal{SP}3$) is a weighted linear programming problem. The optimal despatching strategy for each U_i tends to despatch as many buffered requests as possible to the server with the least backlog:

$$D_{ij}(t) = \begin{cases} X_i(t) & j = j^i \text{ and } X_i(t) > Q_{ij^i}(t) \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

where $j^i = arg\ min_{\forall j \in C} Q_{ij}(t)$ means the shortest queue among all the N queues on N clouds for U_i.

3) Cloud Server Scheduling: The running or idle state of each server on cloud j can be scheduled by maximizing the term (8) in Lemma 1. Recall that the power consumption model is based on the individual server in Sec. II-C2, therefore the indicator function $b_{ij}(t)$ is independent among different clouds. The maximization of term (8) can be decomposed into the following sub-problem ($\mathcal{SP}4$):

$$\mathcal{SP}4: \quad \max_{b_{ij}(t)} \quad B_j \sum_{i=1}^{M} Q_{ij}(t) b_{ij}(t) - V\gamma P_j^s(t)$$
$$s.t. \quad b_{ij}(t) \in \{0, 1\}.$$

In Problem ($\mathcal{SP}4$), we find that the scheduling policy $b_{ij}(t)$ in cloud j is weighted by $B_j Q_{ij}(t)$ while the growth of power consumption caused by running each server is the same (recall the model of $P_j^s(t) = \omega(\sum_{i=1}^{M} b_{ij}(t)/M)^v + (1-\omega)$ in Sec. II-C2). Hence, we can re-rank all servers hosted on cloud j according to their queue backlog in a decreasing order and search from the server with the most backlog to the server with the least backlog. If the growth of $B_j \sum_{i=1}^{M} Q_{ij}(t) b_{ij}(t)$ exceeds the growth of power consumption (i.e., $V\gamma P_j^s(t)$) caused by running a server i, then we need to schedule server i to the running state. Once the growth of $B_j \sum_{i=1}^{M} Q_{ij}(t) b_{ij}(t)$ is smaller than the growth of $V\gamma P_j^s(t)$ for server i, we need to schedule server i and the other servers to the idle state.

Specifically, our algorithm is described in Alg. 1. For a given time slot t, the fronthaul scheduling policies $a_i(t)$ with Eq. 10 cost $O(M)$, and the despatching policies $D_{ij}(t)$ with Eq. 11 cost $O(MN)$. While for the cloud server scheduling policies $b_{ij}(t)$, it costs $O(MlogM)$ to sort M queue backlogs for each cloud. Thus the time complexity of Alg. 1 is $O(MNlogM)$.

Algorithm 1 *RICH Algorithm*

1: At the beginning of each time slot t, the MSP gets the queue backlogs $X_i(t)$ and $Q_{ij}(t)$.
2: Determine the fronthaul scheduling policies $a_i(t)$ with Eq. 10, BBU-based requests despatching policies $D_{ij}(t)$ with Eq. 11 and the cloud server scheduling policies $b_{ij}(t)$ with the solution in Sec. III-B3.
3: Update the queues $X_i(t)$ and $Q_{ij}(t)$ according to Eq. 1 and Eq. 2, respectively.

C. Optimality Analysis

Theorem 1. *Assume that all the queues are initialized to 0. Assume that all arrivals in a time slot $A_i(t)$ i.i.d. and are upper bounded by finite constants so that $A_i(t) \leq A_i^{max}$, $\forall i \in \mathcal{U}, \forall t$. Then, implementing the* RICH *every time slot with any $V \geq 0$ yields the following performance bounds:*

(1) The worst case queue backlog for U_i buffered in the BBU pool $X_i(t)$ is upper bounded by a finite constant over time slot as follows,

$$X_i(t) \leq V\alpha_i + min\{A_i^{max}, C_i^{max}\} \quad (12)$$

Similarly, the worst case queue backlogs for U_i on any cloud j is upper bounded over time slot as follows,

$$Q_{ij}(t) \leq V\alpha_i + 2min\{A_i^{max}, C_i^{max}\} \quad (13)$$

(2) The time average profit achieved by the Alg. 1 is within B/V of the optimal value:

$$\lim_{t\to\infty} \inf\{\sum_{i=1}^{M} \alpha_i r_i - \beta \sum_{i=1}^{M} p_i^f - \gamma \sum_{j=1}^{N} p_j^s\} \geq \eta^* - \frac{B}{V}, \quad (14)$$

where $\eta^* = \sum_{i=1}^{M} \alpha_i r_i^* - \beta \sum_{i=1}^{M} p_i^{f*} - \gamma \sum_{j=1}^{N} p_j^{s*}$, and r_i^*, p_i^{f*} and p_j^{s*} are the optimal values of Problem ($\mathcal{P}$). Proof (see Theorem 4.8 in [11]).

IV. Evaluation

In this section, we evaluate *RICH* in a mobile system consists with C-RAN and MCC by conducting simulations. We have $M = 10$ sets of UEs which include a subset of UEs (e.g., phones, tablets), each U_i has a server hosted in all $N = 400$ mobile clouds, different servers in different clouds can process different number of requests $B_j = 10, j = 1, 2, \cdots, 10$. We assume that every request consumes the same resources (e.g., CPU cycles) [15]. Specifically, the requests from each U_i arrive according to a *random process* of mean rate λ_i. For each mobile user, we set $A_i^{max} = 2\lambda_i$ as the maximum requests limited by the bandwidth capacity of the wireless network between mobile user i and the RRH i. We plot the mean arrival rate for each mobile user and the capacity limitation of each fronthaul link in Table II.

TABLE II: Arrival Rates of Different Mobile Users and Fronthaul Capacities

User i	1	2	3	4	5	6	7	8	9	10
$\lambda_i(\times 10^2)$	1	1	1.5	1.5	2	2	2.5	2.5	3	3
$C_i^{max}(\times 10^2)$	1	1	1	1	1	1.5	1.5	1.5	1.5	1.5

In our setting, each mobile user can offload requests to at most 400 servers across clouds, with a maximum processing capacity of $400 \times B_j = 4000$. We choose a typical setting of parameter $v = 2$ and $\omega = 0.5$ [17] for the normalized server power consumption. We set the parameter α as 1 for all users. Finally, we choose an empirical value of parameter $\beta = 0.6$ and $\gamma = 2$. The following simulations are carried out for 10^5 time slots.

For comparison, we consider two other benchmark algorithms, i.e., *random* and *active*. For *random* algorithm, at every time slot, it randomly schedules the fronthaul links, despatches the transmitted requests and schedules the cloud servers. On the contrary, for *active* algorithm, it switches all fronthaul links to the ON state and all servers in mobile clouds to the running state at every time slot. While for the despatching policy, we use the same policy used in our *RICH*.

A. Algorithm Optimality and System Stability

Fig. 2 plots the time average profit for different values of the control parameter V compared to *random* and *active* algorithms. We observe that: (1) As the value of V increases, the time average profit improves and converges to the

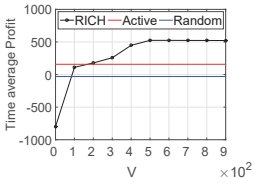

Fig. 2: Time average profit vs. different V under *RICH*, *random* and *active* algorithms

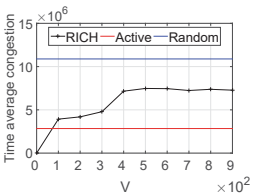

Fig. 3: Time average system congestion vs. different V under *RICH*, *random* and *active* algorithms

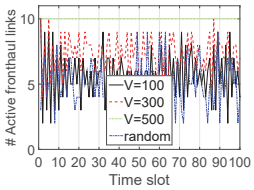

Fig. 4: Number of active fronthaul links vs. different time slots under *RICH* and *random* algorithms with $V = 100, 300$ and 500

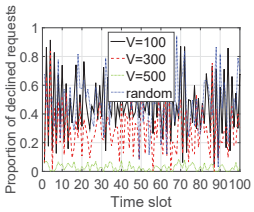

Fig. 5: Proportion of declined requests vs. different time slots under *RICH* and *random* algorithms with $V = 100, 300$ and 500

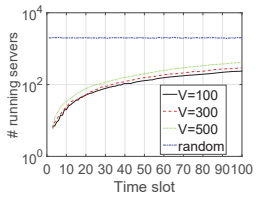

Fig. 6: Number of active servers vs. different time slots under *RICH* and *random* algorithms with $V = 100, 300$ and 500

maximum level for larger values of V. This quantitatively corroborates Theorem 1 in that *RICH* can approach to the optimal profit with a diminishing gap $(1/V)$ (captured by Eq. 14). However, such an improvement starts to diminish with excessive increases of V, which can aggravate the congestion of queues in the system (captured by Eq. 3). (2) While for other algorithms, they can only achieve very less profit due to the scheduling policies they use. For example, for *active* algorithm, it opens all fronthaul links and servers in the mobile clouds. By doing this, they can process many requests but incur a large amount of power consumption so that the profit is also very less. Specially, the profit of *RICH* algorithm is $3.3\times$ $(18.4\times)$ higher than that of *active* (*random*) algorithm.

We then verify the mobile system stability here. We plot the time average queue congestion [11] captured by Eq. 3 under different control values of V in Fig. 3. As shown in Fig. 3, the time average queue congestion increases with the growth of V. Along with Fig. 2, this reflects the tradeoff between profit maximization and system stability shown in Sec. II-C under *RICH*.

B. The Effectiveness of Scheduling Policies

We then evaluate the effectiveness of these three policies here compared with *random* algorithm. We will evaluate the active fronthaul links, the proportion of declined requests and the active servers for *RICH* when $V = 100$, 300 and 500 over time slots. From Fig. 4, we can see that the fronthaul links are dynamically scheduled in our *RICH* and more fronthaul links have been switched to ON state when V increases. When V is excessive high, *RICH* schedules all fronthaul links to ON state according to the fronthaul scheduling policies in Eq. 10. While in Fig. 5, the mobile system declines less requests when the V increase. But it still declines requests with a high paremeter V due the capacity limitation of fronthaul links described in Sec. II-A. From Fig. 6, we can see that more servers are scheduled to running state when the time slots increase. That is because the backlog of queue maintained by each server in mobile clouds increase and more servers have be scheduled to running state based on the solution described in Sec. III-B3.

V. Conclusion

In response to dynamic and unpredictable requests of mobile users due to its mobility served by a mobile system with C-RAN and MCC, we propose a unifying optimization framework for maximizing the profit of MSP. Then we design a *RICH* algorithm which jointly schedules resources both in C-RAN and MCC. Our algorithm can approach a time average profit that is close to the optimum with a gap (1/V) for MSP, while still maintaining strong stability and low congestion to guarantee the QoS for mobile users.

References

[1] A. Checko, H. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. Berger, and L. Dittmann. Cloud RAN for mobile networks: A technology overview. *IEEE Communications Surveys Tutorials*, 17(1):405–426, Firstquarter 2015.

[2] X. Chen, J. Wu, Y. Cai, H. Zhang, and T. Chen. Energy-efficiency oriented traffic offloading in wireless networks: A brief survey and a learning approach for heterogeneous cellular networks. *IEEE Journal on Selected Areas in Communications*, 33(4):627–640, April 2015.

[3] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012-2017, February 2013.

[4] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. MAUI: Making smartphones last longer with code offload. In *Proc. of MobiSys*, pages 49–62, 2010.

[5] B. Dai and W. Yu. Sparse beamforming and user-centric clustering for downlink cloud radio access network. *IEEE Access*, 2:1326–1339, 2014.

[6] C. M. R. Institute. C-RAN white paper: The road towards green RAN. [online]. Available: http://labs.chinamobile. com/cran, June 2014.

[7] H. Jin, X. Wang, S. Wu, S. Di, and X. Shi. Towards optimized fine-grained pricing of IaaS cloud platform. *IEEE Transactions on Cloud Computing*, 3(4):436–448, October 2015.

[8] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *Proc. of INFOCOM*, pages 945–953, March 2012.

[9] W. Li, Y. Zhao, S. Lu, and D. Chen. Mechanisms and challenges on mobility-augmented service provisioning for mobile cloud computing. *IEEE Communications Magazine*, 53(3):89–97, March 2015.

[10] G. Nan, Z. Mao, M. Li, Y. Zhang, S. Gjessing, H. Wang, and M. Guizani. Distributed resource allocation in cloud-based wireless multimedia social networks. *IEEE Network*, 28(4):74–80, July 2014.

[11] M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing System*, Morgan & Claypool, 2010.

[12] X. Rao and V. Lau. Distributed fronthaul compression and joint signal recovery in Cloud-RAN. *IEEE Transactions on Signal Processing*, 63(4):1056–1065, February 2015.

[13] S. Ren and M. van der Schaar. Dynamic scheduling and pricing in wireless cloud computing. *IEEE Transactions on Mobile Computing*, 13(10):2283–2292, October 2014.

[14] K. Wang, K. Yang, and C. Magurawalage. Joint energy minimization and resource allocation in C-RAN with mobile cloud. *IEEE Transactions on Cloud Computing*, PP(99):1–1, 2016.

[15] K. Wang, K. Yang, X. Wang, and C. Magurawalage. Cost-effective resource allocation in C-RAN with mobile cloud. In *Proc. of ICC*, pages 1–6, May 2016.

[16] J. Zhao, T. Quek, and Z. Lei. Coordinated multipoint transmission with limited backhaul data transfer. *IEEE Transactions on Wireless Communications*, 12(6):2762–2775, June 2013.

[17] Z. Zhou, F. Liu, H. Jin, B. Li, B. Li, and H. Jiang. On arbitrating the power-performance tradeoff in SaaS clouds. In *Proc. of INFOCOM*, pages 872–880, April 2013.