IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 1 April 2013; revised 25 May 2013; accepted 29 June 2013. Date of publication 15 July 2013;
date of current version 21 January 2014.

Digital Object Identifier 10.1109/TETC.2013.2273358

Crowdsourcing in Cyber-Physical Systems: Stochastic Optimization with Strong Stability

MING LI (Student Member, IEEE) AND PAN LI (Member, IEEE)

Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State, MS 39762 USA

CORRESPONDING AUTHOR: P. LI (li@ece.msstate.edu)

This work was supported by the U.S. National Science Foundation under Grants CNS-1149786 (CAREER Award), ECCS-1128768, and CNS-1147851.

ABSTRACT Cyber-physical systems (CPSs), featuring a tight combination of computational and physical elements as well as communication networks, attracted intensive attention recently because of their wide applications in various areas. In many applications, especially those aggregating or processing a large amount of data over large spatial regions or long spans of time or both, the workload would be too heavy for any CPS element (or node) to finish on its own. How to enable the CPS nodes to efficiently collaborate with each other to accommodate more CPS services is a very challenging problem and deserves systematic research. In this paper, we present a cross-layer optimization framework for hybrid crowdsourcing in the CPSs to facilitate heavy-duty computation. Particularly, by joint computing resource management, routing, and link scheduling, we formulate an offline finite-queue-aware CPS service maximization problem to crowdsource nodes' computing tasks in a CPS. We then find both lower and upper bounds on the optimal result of the problem. In addition, the lower bound result is proved to be a feasible result that guarantees all queues in the network are finite, i.e., network strong stability. Extensive simulations have been conducted to validate the proposed algorithms' performance.

INDEX TERMS Cyber-physical systems, crowdsourcing, stochastic optimization, network strong stability.

I. INTRODUCTION

Advances in embedded computing, communication, and related hardware technologies have recently brought about cyber-physical systems (CPSs) as a new research frontier [1]–[3]. A CPS is typically designed as a network of interacting elements, such as sensors, actuators, controllers, robotics, instead of standalone devices, to fulfill certain objectives. In many applications, such as object recognition and tracking, traffic management, search and rescue, and healthcare monitoring and delivery, a CPS needs to aggregate or process a large amount of data over large spatial regions and/or long spans of time. The workload would be too heavy for any CPS node to finish on its own and hence requires some CPS nodes to collaborate. The fast growth of the CPS nodes' computing capabilities also increases the opportunities for them to help each other. For example, in a system for tracking endangered species, a sensor node needs to compare the captured images with its pre-installed library to check if the moving object in the images is one of the endangered species that the system is looking for. Since image recognition involves high computational complexity, the corresponding workload may not be completed by the node itself. Collaborations from the other sensor nodes are thus indispensable. How to enable the CPS nodes to efficiently collaborate so as to accomplish more computing tasks is a very challenging problem and deserves a systematic study.

There are generally two issues associated with enabling the collaborations in a CPS: whom to collaborate with and how to collaborate, which are in correspondence to two problems: computing resource management and network design. In this paper, we consider a typical CPS where a base station (BS) or an access point (AP) and a number of nodes are distributed in an area for monitoring and complex information processing. The BS/AP is connected to a central cloud environment which can provide sufficient computation capabilities. The nodes may communicate with each other and with the BS/AP using different spectrums. We present a cross-layer optimization framework for hybrid crowdsourcing in CPSs, which utilizes both the central cloud resource and the CPS nodes' resources. The framework aims to find an optimal

crowdsourcing scheme to maximize CPS services by considering both computing resource management and networking constraints.

Two kinds of computing resource management schemes have been proposed in the literature. Following traditional outsourcing approaches [4], [5], CPS nodes could outsource their workloads to the central cloud via the BS or the AP. However, when the number of nodes gets larger and their workloads become heavier, the performance of a CPS may degrade seriously due to the communication bottleneck at the BS or the AP. On the other hand, some other works, e.g., Cloudlet [6] and Serendipity [7], propose to crowdsource the workload of a node to its nearby resource-rich nodes through WiFi connections. They treat the node as a thin-client to access nearby resources, without considering the cooperations among different outsourcing nodes, available nearby resources, and the central cloud resource. Thus, neither of these two kinds of schemes could take full advantage of the computing resources in a system.

Besides, although there have been many related works on throughput optimization in wireless networks [8]–[20], most of them obtain suboptimal results that are either unbounded or still far from the optimal results. Besides, the above works assume static spectrum availabilities, e.g., constant available bandwidths, which does not fully capture the dynamic characteristics of wireless networks. Some research [21]–[25] proposes to use dynamic control methods to address the dynamics in wireless networks. In particular, a link layer scheduling scheme is proposed in [21] utilizing independent sets which are assumed to be given. Ding et al. [23], [24] develop heuristic algorithms for the throughput maximization problem in cognitive ad hoc networks. Gatsis et al. [25] propose a scheduling policy for an offline optimization problem, which requires the future information of the network. These works cannot guarantee finite queue sizes in the network. Thus, data buffers may overflow and packets may get dropped. Moreover, in traditional network optimization problems, the destination for a source node is predefined. In contrast, in crowdsourcing, it is not clear who are the destinations. For each source node, we need to find out the corresponding computing nodes who can help finish the task, the data outsourcing paths, and the schedule of workload delivery, based on the dynamic underlying network condition and the amount of available computing resource at each node.

Therefore, the aforementioned two issues, i.e, computing resource management and network design, are not independent but tightly coupled issues in CPS crowdsourcing, and have not been well studied before. In this paper, by jointly exploring computing resource management, routing, and link scheduling, we formulate an offline finite-queue-aware CPS service maximization problem P1 to crowdsource in hybrid mode nodes' computing tasks in a CPS, under the constraint that all network queues are finite. We consider that the available spectrum bandwidths and computing resources at each node and at the central cloud are time-varying. The formulated problem is a time-coupling stochastic Mixed-Integer Linear Programming (MILP) problem. Previous approaches usually solve such problems based on Dynamic Programming and suffer from the "curse of dimensionality" problem [26]. They also require detailed statistical information on the system random variables, which may be difficult to obtain in practice.

To solve P1, we employ Lyapunov optimization theory [27] to develop online crowdsourcing algorithms, taking into account network dynamics. Enabling the design of online control algorithms for time-varying systems, Lyapunov optimization operates without requiring any knowledge of the system statistics. In the literature, Lyapunov optimization techniques have been adopted to investigate stochastic optimization problems in wireless networks [27]–[34]. Unfortunately, [28]–[30] cannot guarantee all queues are finite. [31]–[33] develop opportunistic scheduling schemes, which maintain finite queue sizes by dropping some packets. [27], [34] propose joint stability and utility optimization algorithms, but assume that the users' input data rate is interior to the network capacity region. More recently, Lyapunov optimization is also applied to resource allocation in cloud ecosystems [35]–[37]. Huang et al. [35] study the energy efficiency at mobile devices, while [36], [37] explore resource management in central cloud servers and formulate Integer Linear Programming (ILP) problems, thus NP-hard in general. Consequently, in spite of the existing studies, none of the developed algorithms can be adopted to solve our problem, nor to keep all queues finite.

Our approach to solving P1 is as follows. We first formulate an online CPS service maximization problem P2 which has relaxed constraints compared to P1. Since P2 is still an MILP problem that is in general NP-hard but needs to be solved in each time slot, we further relax it into a Linear Programming (LP) problem and obtain an upper bound on the optimal result of P1. We also reformulate P1 into an equivalent offline optimization problem P3. Based on P3, by introducing link-layer virtual queues, we formulate an online finite-queue-aware CPS service maximization problem P4, and are able to decompose it into several subproblems: link scheduling, routing, and computing resource allocation. Since all the subproblems are ILP problems, we develop algorithms to solve them respectively and efficiently based on current network states only. We prove that the approximation algorithm can guarantee that all queues in the network are finite. Moreover, we show that the approximation algorithm can lead to a lower bound on the optimal result of P1.

Our main contributions in this paper can be briefly summarized as follows.

- We present a cross-layer optimization framework for hybrid crowdsourcing in CPSs to accommodate CPS services requiring heavy-duty computation.
- We formulate an offline finite-queue-aware CPS service maximization problem, considering dynamic spectrum and computing resource availability, to crowdsource CPS nodes' computing tasks by joint computing resource allocation, routing, and link scheduling.

- We formulate an online finite-queue-aware CPS service maximization problem, and propose an approximation algorithm which can solve the problem efficiently and guarantee all queues are finite, i.e., network strong stability.
- We prove that the proposed approximation algorithm gives a lower bound on the optimal result of the original offline CPS service maximization problem. An upper bound is also obtained.
- Extensive simulation results show that 1) the obtained lower and upper bounds are very tight, and 2) by distributing some computing tasks to other CPS nodes, the crowdsourcing architecture can in fact deliver more tasks to the central cloud and support a lot more CPS services than other outsourcing architectures.

The rest of this paper is organized as follows. In Section II, we briefly introduce our system models. We then formulate in Section III an offline and a relaxed online CPS service maximization problem by joint computing resource allocation and network design, while considering network dynamics. In Section IV, we reformulate an online CPS service maximization problem, and propose a decomposition based approximation algorithm to solve the problem. Section V proves the proposed approximation algorithm guarantees network strong stability, and derives both a lower bound and an upper bound on the optimal result of the offline optimization problem. Simulations are conducted in Section VI for performance evaluation and comparison. We finally conclude this paper in Section VII.

II. SYSTEM MODELS

A. SYSTEM ARCHITECTURE

We consider a CPS where a base station (or an access point) and a number of nodes (e.g., sensors, robotics, and UAVs) are distributed in an area for monitoring and complex information processing. A typical architecture is shown in Fig. 1. The base station is connected to a central cloud environment which can provide sufficient computation capabilities. The CPS nodes may communicate with each other and with the base station using different spectrums. Here we study a hybrid crowdsourcing architecture for CPSs, where both the central cloud and CPS nodes offer their resources to collaboratively accomplish ceratin tasks. Under this architecture, rather than outsource all the computing workloads to the central cloud via the base station[1], which is the most common approach in cloud computing, we take advantage of both the central cloud and the CPS nodes to finish the computing tasks in the system.

Besides, in order to efficiently support crowdsourcing, the base station is in charge of computing resource allocation, routing, and link scheduling. Following the scheduling decisions, each node divides its computing tasks into multiple subtasks, and transmits them to the other nodes and/or the base station accordingly. Note that in this study we do not discuss the transmission scheduling for sending results from computing nodes back to source nodes. The reason is that the computed results are usually simple data or instructions of much smaller sizes, which can be easily delivered.

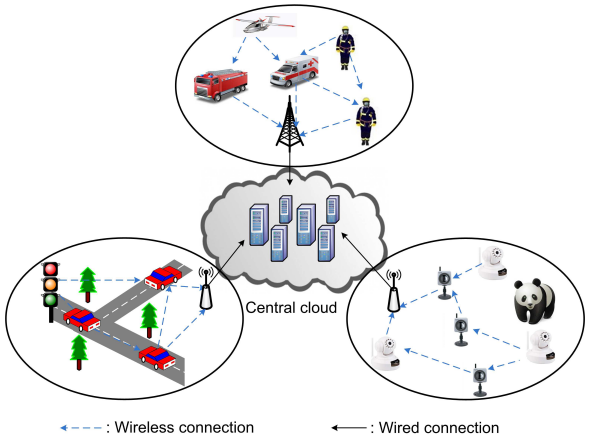


FIGURE 1. Cyber-physical systems for search and rescue, traffic analysis and management, object recognition and monitoring.

B. NETWORK MODEL

Consider a CPS consisting of $\underline{\mathcal{N}} = \{1, 2, \ldots, n, \ldots, N\}$ nodes and a base station B and denote $\mathcal{N} = \underline{\mathcal{N}} \cup \{B\}$. Assume there are a set of delay-insensitive crowdsourcing requests from some CPS nodes, e.g., image processing, statistical data analysis[2]. We denote such requests by $\mathcal{R} = \{1, 2, \ldots, r, \ldots, R\}$, each of which can be denoted as a tuple $\{s_r, v_r(t)\}$ where s_r is the source node of service request r, and $v_r(t)$ stands for the amount of computing job r in time slot t, in terms of the number of instructions.

Since each computing job may require a large amount of computing resources, each job can be divided into multiple subtasks and handled at some nodes and/or the base station. Besides, we assume that a node i is able to process $O_i(t)$ instructions in time slot t according to its current available computing resources. Since $O_i(t)$ depends on the current workload and resource usage of user i, we assume $\{O_i(t)\}_{t=0}^{\infty}$ to be an i.i.d. random process. We also denote by $\epsilon_i^r(t)$ $(i \in \mathcal{N}, r \in \mathcal{R})$ the number of instructions that node i processes for service r in time slot t.

Suppose the CPS nodes are allowed to access a set of spectrum bands $\mathcal{M} = \{1, 2, \ldots, m \ldots, M\}$ with different bandwidths. Assume the bandwidth of band m is an i.i.d. random process denoted by $\{W^m(t)\}_{t=0}^{\infty}$, and can be observed at the beginning of each time slot. Due to their different communication interfaces and geographical locations, the nodes may have different available spectrum bands. Let $\mathcal{M}_i \subseteq \mathcal{M}$ represent the set of available spectrum bands at node $i \in \mathcal{N}$. Then, we may have $\mathcal{M}_i \neq \mathcal{M}_j$ for $i \neq j, i, j \in \mathcal{N}$.

[1]We consider that the base station is connected to the central cloud via a high capacity backbone network. Thus, we consider the central cloud is at the base station in this study.

[2]We will show later that our proposed algorithm can in fact guarantee that the processing delay for each crowdsourcing request is bounded.

C. TRANSMISSION/INTERFERENCE RANGE AND LINK CAPACITY

Suppose the power spectral density of node i on band m is P_i^m. A widely used model [13], [17] for power propagation gain between node i and node j, denoted by g_{ij}, is

$$g_{ij} = C \cdot [d(i,j)]^{-\gamma},$$

where i and j also denote the positions of node i and node j, respectively, $d(i,j)$ refers to the Euclidean distance between i and j, γ is the path loss factor, and C is a constant related to the antenna profiles of the transmitter and the receiver, wavelength, and so on. We assume that the data transmission is successful only if the received power spectral density at the receiver exceeds a threshold P_T^m. Meanwhile, we assume interference becomes non-negligible only if it produces a power spectral density over a threshold of P_I^m at the receiver. Thus, the transmission range for a node i on band m is $R_T^{i,m} = (CP_i^m/P_T^m)^{1/\gamma}$, which comes from $C(R_T^{i,m})^{-\gamma} \cdot P_i^m = P_T^m$. Similarly, based on the interference threshold $P_I^m (P_I^m \leq P_T^m)$, the interference range for a node is $R_I^{i,m} = (CP_i^m/P_I^m)^{1/\gamma}$, which is no smaller than $R_T^{i,m}$. Thus, different nodes may have different transmission ranges/interference ranges on different channels with different transmission power.

In addition, according to the Shannon-Hartley theorem, if node i sends data to node j on link (i,j) using band m, the capacity of link (i,j) on band m during time slot t is

$$c_{ij}^m(t) = W^m(t) \log_2 \left(1 + \frac{g_{ij} P_i^m}{\eta}\right), \tag{1}$$

where $m \in \mathcal{M}_i \cap \mathcal{M}_j$ and η is the thermal noise at the receiver. Note that the denominator inside the log function only contains η. This is because of one of our interference constraints, i.e., when node i is transmitting to node j on band m, all the other neighbors of node j within its interference range are prohibited from using this band. Besides, since the bandwidth of each spectrum $\{W^m(t)\}_{t=0}^{\infty}$ is an i.i.d. random process, $\{c_{ij}^m(t)\}_{t=0}^{\infty}$ is also an i.i.d. random process.

D. DEFINITIONS AND PRELIMINARIES

Next, we introduce some definitions and theorems that we will use in this paper [27].

Definition 1: The time average of a random process $a(t)$, denoted by $\overline{a}$, is $\overline{a} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[a(t)]$.

Definition 2: A discrete time process $a(t)$ is rate stable if $\lim_{t\to\infty} \frac{a(t)}{t} = 0$ with probability 1, and strongly stable if $\lim_{T\to\infty} \sup \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[|a(t)|] < \infty$.

Theorem 1: Let $Q(t)$ denote the queue length of a single-server discrete time queueing system, whose initial state $Q(0)$ is a non-negative real-valued random variable, and future states are driven by stochastic arrival and server processes $a(t)$ and $b(t)$ according to the following dynamic equation:

$$Q(t+1) = \max\{Q(t) - b(t), 0\} + a(t) \text{ for } t \in \{0, 1, 2, \ldots\}.$$

Then $Q(t)$ is rate stable if and only if $\overline{a} \leq \overline{b}$.

Theorem 2: If a queue $Q(t)$ is strongly stable, and there is a finite constant c such that either $a(t) + b^-(t) \leq c$ with probability 1 for all t (where $b^-(t) \triangleq -\min[b(t), 0]$), or $b(t) - a(t) \leq c$ with probability 1 for all t, then $Q(t)$ is rate stable, i.e., $\overline{a} \leq \overline{b}$.

Besides, we say that a network is rate stable or strongly stable if all queues in this network are rate stable or strongly stable as described above.

III. OPTIMIZATION FOR CROWDSOURCING IN CPSs

In this section, we investigate the CPS service optimization problem to crowdsource the nodes' computing tasks in a CPS, by joint computing resource allocation, routing, and link scheduling, with network stability taken into consideration.

A. COMPUTING RESOURCE ALLOCATION CONSTRAINTS

Consider a source node s_r whose computing request contains $v_r(t)$ instructions in time slot t. Let A_r^{max} denote the maximum number of instructions that s_r can possibly generate in one time slot. Then, we have

$$0 \leq v_r(t) \leq A_{max}^r. \tag{2}$$

These $v_r(t)$ instructions may be divided into multiple subtasks and distributed to multiple computing nodes.

In the same time slot, some computing nodes use their available computing resources to process certain number of instructions for the source node s_r, which are those already queued at the nodes. Recall that the number of instructions node i processes for service r is denoted by $\epsilon_i^r(t)$. Denote by $O_i(t)$ the number of instructions node i can process in time slot t. Therefore, we get

$$\sum_{r\in\mathcal{R}} \epsilon_i^r(t) \leq O_i(t). \tag{3}$$

Let O_i^{max} denote the number of instructions that node i can process with all its computing resources. We can subsequently have $\epsilon_i^r(t) \leq \sum_{r\in\mathcal{R}} \epsilon_i^r(t) \leq O_i(t) \leq O_i^{max}$. If $\epsilon_i^r(t) = 0$, it means that node i does not provide computing resource for service r in time slot t. Note that it is possible that $\epsilon_{s_r}^r(t) > 0$, i.e., the source node processes part of the computing task by itself.

B. NETWORK LAYER CONSTRAINTS

Each node may receive computing subtasks of multiple services, and process them by themselves or forward them towards other computing nodes. Thus, each node i maintains a data queue Q_i^r, which is at the network layer, for each computing service request r that goes through it. The queueing law for each Q_i^r is:

$$Q_i^r(t+1) = \max\left\{Q_i^r(t) - \sum_{j\in\mathcal{T}_i} f_{ij}^r(t) - \epsilon_i^r(t), 0\right\} + \sum_{\{j|i\in\mathcal{T}_j\}} f_{ji}^r(t) + v_r(t) \cdot 1_{i=s_r}, \tag{4}$$

where $f_{ij}^r(t)$ denotes the flow rate, in terms of number of instructions, on link (i, j) for service r in time slot t, and $\mathcal{T}_i = \bigcup_{m\in\mathcal{M}_i} \mathcal{T}_i^m$ given that $\mathcal{T}_i^m$ is the set of nodes that can access the band m and are within the transmission range of node i on band m. $1_{\{A\}}$ is a binary indicator function, which is equal to 1 if event A is true and 0 otherwise. Each node updates its queue size in each time slot according to the queueing law (4).

Besides, since there is no incoming data at the source node of service request r, we have the following constraint:

$$\sum_{\{j|i\in\mathcal{T}_j\}} f_{ji}^r(t) = 0, \qquad \forall\, i = s_r, r \in \mathcal{R}. \tag{5}$$

C. LINK SCHEDULING CONSTRAINTS

Next, we illustrate the channel allocation and link scheduling constraints on data transmissions.

Assume band m is available at both node i and node j, i.e., $m \in \mathcal{M}_i \cap \mathcal{M}_j$. We denote

$$s_{ij}^m(t) = \begin{cases} 1, & \text{if node } i \text{ transmits to node } j \text{ using} \\ & \text{channel } m \text{ in time slot } t, \\ 0, & \text{otherwise.} \end{cases}$$

Since a node is not able to transmit to or receive from multiple nodes on the same frequency band simultaneously, we have

$$\sum_{j\in\mathcal{T}_i^m} s_{ij}^m(t) \le 1, \text{ and } \sum_{\{i|j\in\mathcal{T}_i^m\}} s_{ij}^m(t) \le 1. \tag{6}$$

Besides, a node cannot use the same frequency band for transmission and reception, due to "self-interference" at physical layer, i.e.,

$$\sum_{\{i|j\in\mathcal{T}_i^m\}} s_{ij}^m(t) + \sum_{q\in\mathcal{T}_j^m} s_{jq}^m(t) \le 1. \tag{7}$$

Moreover, recall that in this study, we consider each node is only equipped with a single radio, which means each node can only transmit or receive on one frequency band at a time. Thus, we can have

$$\sum_{m\in\mathcal{M}_j} \sum_{\{i|j\in\mathcal{T}_i^m\}} s_{ij}^m(t) + \sum_{m\in\mathcal{M}_j} \sum_{q\in\mathcal{T}_j^m} s_{jq}^m(t) \le 1. \tag{8}$$

Notice that (6) and (7) will hold whenever (8) holds.

In addition to the above constraints at a certain node, there are also constraints due to potential interference among different nodes. In particular, on a frequency band m, if node i uses this band for transmitting data to a neighboring node $j \in \mathcal{T}_i^m$, then any other node that can interfere with node j's reception should not use this band. To model this constraint, we denote by $\mathcal{P}_j^m$ the set of nodes that can interfere with node j's reception on band m, i.e.,

$$\mathcal{P}_j^m = \left\{p | d(p, j) \le R_I^{p,m}, p \ne j, \mathcal{T}_p^m \ne \emptyset\right\}.$$

The physical meaning of $\mathcal{T}_p^m \ne \emptyset$ in the above definition is that node p has at least one neighbor to which it may transmit data and hence cause interference to node j's reception. Therefore, we have

$$\sum_{\{i|j\in\mathcal{T}_i^m\}} s_{ij}^m(t) + \sum_{q\in\mathcal{T}_p^m} s_{pq}^m(t) \le 1 \qquad (\forall p \in \mathcal{P}_j^m). \tag{9}$$

Besides, let $c_{ij}^m(t)$ be the link capacity on link (i, j) over band m in time slot t calculated according to (1). The flow rate over link (i, j) should satisfy the following:

$$\alpha \sum_{r\in\mathcal{R}} f_{ij}^r(t) \le \sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t) s_{ij}^m(t)\Delta t, \tag{10}$$

where α is the length, in terms of number of bits, of an instruction, and Δt is the time duration of one time slot. (10) indicates that the total number of bits transmitted on a link during one time slot cannot exceed the link's capacity multiplied by the duration of a time slot.

D. OFFLINE FINITE-QUEUE-AWARE CPS SERVICE MAXIMIZATION

In the offline finite-queue-aware CPS service maximization problem, we aim to maximize the long-term total amount of computing tasks that can be processed in the network while guaranteeing finite queue backlog size at each node. Specifically, the time-averaged expected total amount of CPS services that can be supported can be calculated as:

$$\sum_{r\in\mathcal{R}} \overline{v_r} = \sum_{r\in\mathcal{R}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[v_r(t)].$$

We utilize the base station to make an optimal decision to allocate computing resources, determine end-to-end outsourcing paths, and schedule the transmissions, so as to support as many computing services as possible from a long term point of view. Thus, the offline finite-queue-aware CPS service maximization problem under the aforementioned constraints can be formulated as follows:

$$\begin{aligned} \textbf{P1:} \qquad & \textbf{Maximize} \quad \psi = \sum_{r\in\mathcal{R}} \overline{v_r} \\ \textbf{s.t.} \quad & \text{Constraints (2)–(3), (5), (8)–(10) for all } t \ge 0, \\ & \mathbf{Q}(t) \text{ is strongly stable} \end{aligned} \tag{11}$$

where $\mathbf{Q}(t) = \{Q_i^r(t), \forall i \in \mathcal{N}, r \in \mathcal{R}\}$. We denote the optimal result of P1 by ψ_{P1}^*. We can see that without the constraint (11), P1 is a time-coupling stochastic Mixed-Integer Linear Programming (MILP) problem, which is already prohibitively expensive to solve. Previous approaches usually solve such problems based on Dynamic Programming and suffer from the "curse of dimensionality" problem [26]. They also require detailed statistical information on the random variables in the problem, i.e., the available spectrums and computing resources at each node, which may be difficult to obtain in practice. The constraint (11) makes P1 an even more complicated problem. In the next subsection, we will formulate a relaxed online optimization problem to break the time coupling, so that the problem can be solved based on the current network state only.

E. ONLINE CPS SERVICE MAXIMIZATION

Here, we employ Lyapunov optimization theory to formulate an online CPS service maximization problem, solving which does not require any knowledge of the network statistics.

Particularly, since the queues maintained in the network are $\mathbf{Q}(t) = \{Q_i^r(t), \forall i \in \mathcal{N}, r \in \mathcal{R}\}$, we can define a Lyapunov function [27] as

$$L(\mathbf{Q}(t)) \triangleq \frac{1}{2}\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}}(Q_i^r(t))^2.$$

This represents a scalar measure of queue congestion in the network. $L(\mathbf{Q}(t))$ being small implies that all queue backlogs are small, while $L(\mathbf{Q}(t))$ being large implies that at least one queue backlog is large. Besides, the one-slot conditional Lyapunov drift can be defined as

$$\Delta(\mathbf{Q}(t)) \triangleq \mathbb{E}[L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t))|\mathbf{Q}(t)]. \quad (12)$$

Since our objective is to support the most computing services, and take control action to limit $\Delta(\mathbf{Q}(t))$, we minimize the following drift-plus-penalty function:

$$\Delta(\mathbf{Q}(t)) - V\mathbb{E}\left[\sum_{r\in\mathcal{R}} v_r(t)|\mathbf{Q}(t)\right],$$

where $V \geq 0$ is a parameter that represents an importance weight on how much we emphasize on the CPS service maximization. Such a scheduling decision can be explained as follows: We want to make $\Delta(\mathbf{Q}(t))$ small to push queue backlog towards a lower congestion state, and we also want to make $\sum_{r\in\mathcal{R}} v_r(t)$ large in each time slot so that we can support more CPS services in the network.

We can have the following lemma.

Lemma 1: Given $\Delta(\mathbf{Q}(t))$ defined in (12), we have

$$\Delta(\mathbf{Q}(t)) - V\mathbb{E}\left[\sum_{r\in\mathcal{R}} v_r(t)|\mathbf{Q}(t)\right] \leq B + \Psi(t), \quad (13)$$

where B is a constant, i.e.,

$$B = \frac{1}{2}\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}}\left[\left(\max_{j\in\mathcal{T}_i}\left\{\frac{1}{\alpha}c_{ij}^{max}\Delta t\right\} + O_i^{max}\right)^2 + \left(\max_{\{j|i\in\mathcal{T}_j\}}\left\{\frac{1}{\alpha}c_{ji}^{max}\Delta t\right\} + A_r^{max}\cdot 1_{i=s_r}\right)^2\right],$$

and $\Psi(t)$ is related to the variables $f_{ij}^r(t)$'s, $v_r(t)$'s, and $\epsilon_i^r(t)$'s, i.e.,

$$\Psi(t) = \mathbb{E}\Bigg[\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}} Q_i^r(t)\Bigg(\sum_{\{j|i\in\mathcal{T}_j\}} f_{ji}^r(t) + v_r(t)\cdot 1_{i=s_r} - \sum_{j\in\mathcal{T}_i} f_{ij}^r(t) - \epsilon_i^r(t)\Bigg)|\mathbf{Q}(t)\Bigg] - V\mathbb{E}\Big[\sum_{r\in\mathcal{R}} v_r(t)|\mathbf{Q}(t)\Big].$$

Note that c_{ij}^{max} denotes the maximum possible link capacity of link (i, j). Since according to (1) $c_{ij}^m(t)$ depends on $d(i, j)$, P_i^m and $W^m(t)$ ($m \in \mathcal{M}_i \cap \mathcal{M}_j$), among which $d(i, j)$ and P_i^m are constants, then c_{ij}^{max} is determined by W^{max}, i.e., the maximum bandwidth that the channels available on link (i, j) can have.

Proof: Please refer to Appendix A, available in the online supplemental material, for the detailed proof. ■

Based on the drift-plus-penalty framework, our objective is to observe the current channel state, available computing resources, as well as queue backlogs $\mathbf{Q}(t)$, and to make control decisions to minimize the right-hand-side of (13), i.e., a weighted sum of $\mathbf{Q}(t)$ plus a penalty $-\mathbb{E}[\sum_{r\in\mathcal{R}} v_r(t)|\mathbf{Q}(t)]$. Since B is a constant, we only need to minimize $\Psi(t)$. We now use the concept of opportunistically minimizing an expectation [27], which is to minimize:

$$\Psi'(t) = \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}} Q_i^r(t)\Bigg(\sum_{\{j|i\in\mathcal{T}_j\}} f_{ji}^r(t) + v_r(t)\cdot 1_{i=s_r} - \sum_{j\in\mathcal{T}_i} f_{ij}^r(t) - \epsilon_i^r(t)\Bigg) - V\cdot\left(\sum_{r\in\mathcal{R}} v_r(t)\right).$$

Therefore, the problem of online CPS service maximization can be formulated as follows:

$$\textbf{P2:} \quad \textbf{Minimize} \quad \Psi'(t)$$
$$\textbf{s.t.} \quad \text{Constraints (2)–(3), (5), (8)–(10)}$$

We denote the optimal solution of $v_r(t)$ to P2 by $\hat{v}_r^*(t)$, and the corresponding value of the objective function of P1, i.e., $\sum_{r\in\mathcal{R}} \overline{v_r}$, by ψ_{P2}^*. We can see that P2 is a Mixed-Integer Linear Programming (MILP) problem, which is in general NP hard to solve [38]. Note that we do not include constraint (11) in P2 because otherwise P2 would still be a very complicated problem to solve. Since P2 has relaxed constraints compared to P1, we use it to obtain an upper bound on ψ_{P1}^* which will become clear later.

IV. AN APPROXIMATION ALGORITHM FOR FINITE-QUEUE-AWARE CPS SERVICE OPTIMIZATION

We notice that P2 cannot guarantee finite queue sizes in the network. In this section, we first reformulate P1 into an equivalent offline optimization problem P3. Based on P3, by introducing link-layer virtual queues, we formulate an online finite-queue-aware CPS service maximization problem P4, and propose an approximation algorithm that can efficiently solve it in each time slot.

In particular, by summing the inequality (10) over all $t \in \{0, 1, \ldots, T-1\}$, and taking expectation and limitation of both sides, we can get

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\alpha\sum_{r\in\mathcal{R}} f_{ij}^r(t)\right] \leq \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t\right]. \quad (14)$$

Applying (14) as an extra constraint, the optimization problem P1 is equivalent to the following optimization problem

P3:

$$\textbf{P3:} \quad \textbf{Maximize} \quad \psi = \sum_{r\in\mathcal{R}} \overline{v_r}$$

s.t. Constraints (2)–(3), (5), (8)–(11), (14) for all t.

We denote the optimal result of P3 by ψ^*_{P3}. In what follows, we will formulate a drift-plus-penalty problem based on P3, which we call P4.

A. MODELING VIRTUAL QUEUES

Consider a virtual queue $X_{ij}(t)$ at node i for each of its one-hop neighbor j with the following queueing law:

$$X_{ij}(t+1) = \max\left\{X_{ij}(t) - \frac{1}{\alpha}\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t, 0\right\} + \sum_{r\in\mathcal{R}} f_{ij}^r(t). \tag{15}$$

This virtual queue can be understood as the link-layer buffer on link (i, j). The queue backlog $X_{ij}(t)$ represents the total number of instructions stored at node i to be transmitted to node j at the beginning of time slot t[3].

For queue $X_{ij}(t)$, we have

$$\begin{aligned}&\frac{1}{\alpha}\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t - \sum_{r\in\mathcal{R}} f_{ij}^r(t)\\ &\le \frac{1}{\alpha}\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t\\ &\le \frac{1}{\alpha}c_{ij}^{max}\Delta t\end{aligned} \tag{16}$$

where $\frac{1}{\alpha}c_{ij}^{max}\Delta t$ is a constant, due to $\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} s_{ij}^m(t) \le \sum_{m\in\mathcal{M}_i}\sum_{j\in\mathcal{T}_i^m} s_{ij}^m(t) \le 1$ according to (8), i.e., one node can transmit to at most one neighbor on one band at a time. Therefore, if we can guarantee the strong stability of this queue, we can ensure its rate stability, i.e., constraint (14), according to Theorem 2. Besides, the virtual queue backlog is always nonnegative according to the queueing law (15).

Rather than utilizing the virtual queue $X_{ij}(t)$ directly, we build another virtual queue $Y_{ij}(t)$ based on $X_{ij}(t)$ as $Y_{ij}(t) = \sigma X_{ij}(t)$, where $\sigma = \max_{i\in\mathcal{N},j\in\mathcal{T}_i}\{\frac{1}{\alpha}c_{ij}^{max}\Delta t\}$. Consequently, its queuing law is

$$Y_{ij}(t+1) = \max\left\{Y_{ij}(t) - \frac{\sigma}{\alpha}\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t, 0\right\} + \sigma\sum_{r\in\mathcal{R}} f_{ij}^r(t). \tag{17}$$

Note that if we can guarantee the strong stability of $Y_{ij}(t)$, the strong stability of $X_{ij}(t)$ directly follows, and constraint (14) can be satisfied.

[3]In order to guarantee that the queue size of $X_{ij}(t)$ is an integer in each time slot, the service rate of the queue should in fact be $\lfloor\frac{1}{\alpha}\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t\rfloor$. Here, we assume $\frac{1}{\alpha}\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t$ to be integers for simplicity.

B. ONLINE FINITE-QUEUE-AWARE CPS SERVICE MAXIMIZATION

Next, we formulate an online finite-queue-aware CPS service maximization problem corresponding to P3. Notice that in problem P3, two types of queues $\mathbf{\Theta}(t) = \{\mathbf{Q}(t), \mathbf{Y}(t)\}$ are maintained, i.e., the real queue (network-layer queue) $\mathbf{Q}(t) = \{Q_i^r(t), \forall r \in \mathcal{R}, i \in \mathcal{N}\}$ and the virtual queue (link-layer queue) $\mathbf{Y}(t) = \{Y_{ij}(t), \forall i \in \mathcal{N}, j \in \mathcal{T}_i\}$. We assume $\mathbf{Q}(0) = \mathbf{0}$ and $\mathbf{Y}(0) = \mathbf{0}$. Thus, we can similarly define a Lyapunov function of $\mathbf{\Theta}(t)$ as

$$L(\mathbf{\Theta}(t)) \triangleq \frac{1}{2}[\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}}(Q_i^r(t))^2 + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{T}_i}(Y_{ij}(t))^2].$$

Consequently, its one-slot conditional Lyapunov drift can be defined as

$$\Delta(\mathbf{\Theta}(t)) \triangleq \mathbb{E}\Big[L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t))|\mathbf{\Theta}(t)\Big]. \tag{18}$$

Similar to that in P2, i.e., to support more CPS services and to limit the queue backlogs, we intend to minimize an upper bound on the following drift-plus-penalty expression:

$$\Delta(\mathbf{\Theta}(t)) - V\mathbb{E}\Big[\sum_{r\in\mathcal{R}} v_r(t)|\mathbf{\Theta}(t)\Big]. \tag{19}$$

We can have the following lemma.

Lemma 2: Given $\Delta(\mathbf{\Theta}(t))$ defined in (18), we have

$$\begin{aligned}&\Delta(\mathbf{\Theta}(t)) - V\mathbb{E}\Big[\sum_{r\in\mathcal{R}}\epsilon_i^r(t)|\mathbf{\Theta}(t)\Big]\\ &\le B' + \Psi_1(t) + \Psi_2(t) + \Psi_3(t)\end{aligned} \tag{20}$$

where B' is a constant, i.e.,

$$\begin{aligned}B' = \frac{1}{2}\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}}\Big[&\big(\max_{j\in\mathcal{T}_i}\{\frac{1}{\alpha}c_{ij}^{max}\Delta t\} + O_i^{max}\big)^2\\ &+\big(\max_{\{j|i\in\mathcal{T}_j\}}\{\frac{1}{\alpha}c_{ji}^{max}\Delta t\} + A_r^{max}\cdot 1_{i=s_r}\big)^2\Big]\\ &+\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{T}_i}\big(\frac{\sigma}{\alpha}c_{ij}^{max}\Delta t\big)^2,\end{aligned}$$

$\Psi_1(t)$ is only related to the link scheduling variables $s_{ij}^m(t)$'s, i.e.,

$$\Psi_1(t) = -\frac{\sigma}{\alpha}\mathbb{E}\Big[\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{T}_i}\Big(Y_{ij}(t)\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t\Big)|\mathbf{Y}(t)\Big],$$

$\Psi_2(t)$ is only related to the routing variables $f_{ij}^r(t)$'s, i.e.,

$$\begin{aligned}\Psi_2(t) = &\mathbb{E}\Big[\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}} Q_i^r(t)\Big(\sum_{\{j|i\in\mathcal{T}_j\}} f_{ji}^r(t) - \sum_{j\in\mathcal{T}_i} f_{ij}^r(t)\Big)|\mathbf{Q}(t)\Big]\\ &+\mathbb{E}\Big[\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{T}_i}\Big(\sigma Y_{ij}(t)\sum_{r\in\mathcal{R}} f_{ij}^r(t)\Big)|\mathbf{Y}(t)\Big],\end{aligned}$$

and $\Psi_3(t)$ is only related to the computing resource allocation variables $v_r(t)$'s and $\epsilon_i^r(t)$'s, i.e.,

$$\Psi_3(t) = \mathbb{E}\Big[\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}}\Big(Q_i^r(t)(v_r(t)\cdot 1_{i=s_r} - \epsilon_i^r(t))\Big)|\mathbf{Q}(t)\Big] - V\mathbb{E}\Big[\sum_{r\in\mathcal{R}} v_r(t)|\mathbf{\Theta}(t)\Big].$$

Proof: Please refer to Appendix B, available in the online supplemental material, for the detailed proof. ■

From Lemma 2 , to minimize the right-hand-side of (20) is to minimize $\Psi_1(t)+\Psi_2(t)+\Psi_3(t)$. Note that $\Psi_1(t)$, $\Psi_2(t)$ and $\Psi_3(t)$ are all conditional expectations. The same as that in P2, we use the concept of opportunistically minimizing an expectation, i.e., to minimize $\Psi_1'(t)+\Psi_2'(t)+\Psi_3'(t)$ where

$$\Psi_1'(t) = -\frac{\sigma}{\alpha}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{T}_i}\Big(Y_{ij}(t)\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t\Big),$$

$$\Psi_2'(t) = \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}}\Big(Q_i^r(t)(\sum_{\{j|i\in\mathcal{T}_j\}} f_{ji}^r(t) - \sum_{j\in\mathcal{T}_i} f_{ij}^r(t)\Big) + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{T}_i}\Big(\sigma Y_{ij}(t)\sum_{r\in\mathcal{R}} f_{ij}^r(t)\Big),$$

$$\Psi_3'(t) = \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}}(Q_i^r(t)(v_r(t)\cdot 1_{i=s_r} - \epsilon_i^r(t))) - V\sum_{r\in\mathcal{R}} v_r(t).$$

Thus, the online finite-queue-aware CPS service maximization problem can be formulated as

$$\textbf{P4:}\quad \textbf{Minimize}\quad \Psi_1'(t)+\Psi_2'(t)+\Psi_3'(t)$$
$$\textbf{s.t.}\quad \text{Constraints (2)–(3), (5), (8)–(10)}$$
$$\mathbf{Q}(t)\text{ and }\mathbf{Y}(t)\text{ are strongly stable.}\qquad (21)$$

Note that the constraint (14) has been left out in P4 (compared to P3) since it can be guaranteed if $\mathbf{Y}(t)$ is strongly stable as shown before.

C. A DECOMPOSITION BASED APPROXIMATION ALGORITHM

In P4, we notice that $\Psi_1'(t)$, $\Psi_2'(t)$ and $\Psi_3'(t)$ are related to variables $s_{ij}^m(t)$'s, $f_{ij}^r(t)$'s, and $v_r(t)$'s and $\epsilon_i^r(t)$'s, respectively. Thus, in the following we decompose P4 into four subproblems (from S1 to S4) and solve them respectively to obtain a suboptimal and feasible solution to P4.

1) LINK SCHEDULING

First, we minimize $\Psi_1'(t)$ by finding the optimal link scheduling policy, i.e., determining the variables $s_{ij}^m(t)$'s ($\forall i\in\mathcal{N}$, $j\in\mathcal{T}_i, m\in\mathcal{M}_i\cap\mathcal{M}_j$), as follows:

$$\textbf{S1:}\quad \textbf{Minimize}\quad \Psi_1'(t)$$
$$\textbf{s.t.}\quad \text{Constraints (8)–(9).}$$

Since the only variables $s_{ij}^m(t)$'s can only take value of 0 or 1, the above formulated problem is a Binary Integer Programming (BIP) problem, which can be solved by applying the traditional *branch-and-bound* or *branch-and-cut* [38] approach. In the following, we develop a more efficient greedy algorithm to find a suboptimal solution to this problem, which is called the sequential-fix (SF) algorithm based on a similar idea to that in [17] and [12].

The main idea of SF is to fix the values of $s_{ij}^m(t)$'s sequentially through a series of relaxed linear programming problems. Specifically, we first set $s_{ij}^m(t)$'s to 0 if $Y_{ij}(t)=0$, remove all the terms associated with such $s_{ij}^m(t)$'s from the objective function, and eliminate the related constraints in (8) and (9). Then, in each iteration, we first relax all the 0-1 integer constraints on $s_{ij}^m(t)$'s to $0\le s_{ij}^m(t)\le 1$ to transform the problem to a linear programming (LP) problem. Then, we solve this LP to obtain an optimal solution with each $s_{ij}^m(t)$ being between 0 and 1. Among all the values, we set the largest $s_{ij}^m(t)$ to 1. After that, by (8), we can fix $s_{pj}^n(t)=0$ and $s_{jq}^n(t)=0$ for any $n\in\mathcal{M}_j$, $\{p|j\in\mathcal{T}_p^n, p\ne i\}$, $q\in\mathcal{T}_j^n$, and by (9), we can fix $s_{pj}^m(t)=0$ and $s_{tq}^m(t)=0$ for any $\{p|j\in\mathcal{T}_p^m, p\ne i\}$, $t\in\mathcal{P}_j^m$, $q\in\mathcal{T}_t^m$. Besides, if the result includes some $s_{ij}^m(t)$'s with the value of 1, we can set those $s_{ij}^m(t)$'s to 1 and perform an additional fixing for the largest fractional variable in the current iteration as illustrated above. Having fixed some $s_{ij}^m(t)$'s, we remove all the terms associated with those already fixed $s_{ij}^m(t)$'s from the objective function, eliminate the related constraints in (8) and (9), and update the problem to a new one for the next iteration. The iteration continues until we fix all $s_{ij}^m(t)$'s to be either 0 or 1.

2) ROUTING

Second, we minimize $\Psi_2'(t)$ by finding the optimal routing policy, i.e., determining the variables $f_{ij}^r(t)$'s ($\forall r\in\mathcal{R}$, $i\in\mathcal{N}, j\in\mathcal{T}_i$), as follows:

$$\textbf{Minimize}\quad \Psi_2'(t)$$
$$\textbf{s.t.}\quad \text{Constraints (5), (10).}$$

By reorganizing the objective function, we can reformulate S2 as follows:

$$\textbf{S2:}\quad \textbf{Minimize}$$
$$\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{T}_i}\big(-Q_i^r(t)+Q_j^r(t)+\sigma Y_{ij}(t)\big)\cdot f_{ij}^r(t)$$
$$\textbf{s.t.}\quad \text{Constraints (5), (10).}$$

We can see that S2 is an Integer Linear Programming (ILP) problem with the only variables being $f_{ij}^r(t)$'s. We notice that the total flow rate $\sum_{r\in\mathcal{R}} f_{ij}^r(t)$ over link (i,j) does not affect the flow rates over other links $\{(p,q)|p\ne i\cap q\ne j\}$, and only depends on its link capacity according to the constraint (10). Besides, the objective function of S2 can be viewed as a weighted sum of the variables $f_{ij}^r(t)$'s. Therefore, we can determine the flow rate over any link (i,j) at node i locally, based on its current queue backlogs $Q_i^r(t)$ and $Y_{ij}(t)$, and the queue backlogs of node j, i.e., $Q_j^r(t)$. In the following, we will propose an algorithm to obtain the optimal solution of $f_{ij}^r(t)$'s.

In particular, a node i ($\forall i \in \mathcal{N}$) first sets the variables $f_{ij}^r(t)$'s ($\forall j = s_r, r \in \mathcal{R}$) to 0 according to constraint (5). In order to minimize the objective function, node i also sets the variables $f_{ij}^r(t)$'s ($\forall j \in \mathcal{T}_i, r \in \mathcal{R}$) with non-negative coefficients to 0. For all the variables $f_{ij}^r(t)$'s (with negative coefficients) over link (i,j), node i finds out the one with the smallest coefficient, and sets it to $\frac{1}{\alpha}\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t$ while the rest to 0. It means that the link (i,j) is fully utilized to deliver that specific computing service. Besides, if there are multiple variables $f_{ij}^r(t)$'s with the same smallest coefficients, node i randomly picks one of them and sets it to $\frac{1}{\alpha}\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t$ while the rest to be 0. Note that $s_{ij}^m(t)$'s are known from the link scheduling optimization problem S1. It is possible that $\frac{1}{\alpha}\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c_{ij}^m(t)s_{ij}^m(t)\Delta t$ is equal to 0 if $\sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} s_{ij}^m(t) = 0$. Then, the corresponding variable $f_{ij}^r(t)$ is also equal to 0.

3) COMPUTING RESOURCE ALLOCATION

Third, we minimize $\Psi_3'(t)$ by finding the optimal computing resource allocation policy, i.e., determining the variables $\epsilon_i^r(t)$'s ($\forall r \in \mathcal{R}, i \in \mathcal{N}$) and $v_r(t)$'s ($\forall r \in \mathcal{R}$), as follows:

$$\textbf{Minimize} \quad \Psi_3'(t)$$
$$\textbf{s.t.} \quad \text{Constraints (2)–(3).}$$

Notice that $v_r(t)$'s and $\epsilon_i^r(t)$'s are uncoupled variables. Therefore, the above problem can be further divided into two individual problems, S3 and S4, which are related to $v_r(t)$'s and $\epsilon_i^r(t)$'s, respectively:

$$\textbf{S3:} \quad \textbf{Minimize} \quad \sum_{r\in\mathcal{R}} (Q_{s_r}^r(t) - V)v_r(t)$$
$$\textbf{s.t.} \quad \text{Constraint (2)}$$

where $Q_{s_r}^r(t)$ is the queue size for service r at its source node s_r, and

$$\textbf{S4:} \quad \textbf{Minimize} \quad \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{N}} -Q_i^r(t)\epsilon_i^r(t)$$
$$\textbf{s.t.} \quad \text{Constraint (3)}$$

We can find that both S3 and S4 are ILP problems. Since the computing resource allocation at one computing node does not affect the allocation policy at any other nodes according to constraints (2) and (3), we develop the following algorithms to enable each computing node to locally find its optimal computing resource allocation policy.

The optimal solution to S3 can be directly obtained as follows. Particularly, for any $r \in \mathcal{R}$, we have

$$v_r(t) = \begin{cases} A_{max}^r, & \text{if } Q_{s_r}^r(t) - V < 0 \\ 0, & \text{otherwise.} \end{cases}$$

In order to minimize the objective function of S4, among all the service tasks that go through it (including the one generated by itself and those generated by the other nodes), node i sets the $\epsilon_i^r(t)$ with the smallest coefficient to $O_i(t)$, i.e., the amount of all its available computing resources, and the other variables to 0. Besides, if there are multiple variables with the same smallest negative coefficient, node i randomly picks one of them and sets it to $O_i(t)$ and the others to 0.

In each time slot, the online finite-queue-aware CPS service maximization problem P4 can be solved after S1–S4 are solved. The queues $\mathbf{Q}(t)$ and $\mathbf{Y}(t)$ are also updated in each time slot according to the queueing laws (4) and (17), respectively. Although the constraint (21) is not considered in S1–S4, we will show in the next section that both $\mathbf{Q}(t)$ and $\mathbf{Y}(t)$ are strongly stable. We denote the corresponding time-averaged expected total CPS services, i.e., the value of the objective function of P1 and P3, by ψ_{P4}.

V. PERFORMANCE ANALYSIS

In this section, we first prove that the proposed approximation algorithm can guarantee network strong stability. Then, we derive both lower and upper bounds on the optimal result of P1.

A. NETWORK STRONG STABILITY

Although [27], [34] have proved that all network queues are finite when applying Lyapunov optimization theory to solve stochastic network optimization problems, they assume that the users' input data is interior to the network capacity region and the formulated drift-plus-penalty function is optimally solved. Since we do not have any such assumption and our proposed approximation algorithm finds a feasible solution to P4 by joint computing resource management, routing, and scheduling, it is a more challenging problem to prove all the network queues are strongly stable in our case.

Theorem 3: Our proposed decomposition based approximation algorithm for P4 guarantees that the queues $\mathbf{Q}(t)$ and $\mathbf{Y}(t)$ are both strongly stable.

Proof: We first demonstrate the strong stability of $\mathbf{Q}(t)$ by considering an arbitrary queue $Q_i^r(t)$. Specifically, we prove by induction that $Q_i^r(t) \le V + A_{max}^r$ when $i = s_r$ and $i \ne s_r$, separately.

1) $i = s_r$: We first investigate the stability of $Q_{s_r}^r(t)$, whose queueing law is as follows:

$$Q_{s_r}^r(t+1) = \max\{Q_{s_r}^r(t) - \sum_{j\in\mathcal{T}_{s_r}} f_{s_rj}^r(t) - \epsilon_{s_r}^r(t), 0\} + v_r(t). \tag{22}$$

When $t = 0$, we have $Q_{s_r}^r(0) = 0 < V + A_{max}^r$.

Assume that we have $Q_{s_r}^r(t) \le V + A_{max}^r$ when $t = t'$ ($t' \ge 0$).

- If $Q_{s_r}^r(t') \ge V$, according to the optimal solution to S3, we know that $v_r(t') = 0$. Thus, we have
$$Q_{s_r}^r(t'+1) \le Q_{s_r}^r(t') \le V + A_{max}^r.$$
- If $Q_{s_r}^r(t') < V$, according to the optimal solution to S3, we get that $v_r(t') = A_{max}^r$. Following (22), we have
$$Q_{s_r}^r(t'+1) \le Q_{s_r}^r(t') + A_{max}^r \le V + A_{max}^r.$$

Therefore, we have $Q^r_{s_r}(t) \le V + A^r_{max}$, and hence $Q^r_{s_r}(t)$ is strongly stable.

2) $i \neq s_r$: We then explore the stability of $Q^r_i(t)$ when $i \neq s_r$, whose queueing law can be expressed as:

$$Q^r_i(t+1) = \max\{Q^r_i(t) - \sum_{j\in\mathcal{T}_i} f^r_{ij}(t) - \epsilon^r_i(t), 0\} + \sum_{\{j|i\in\mathcal{T}_j\}} f^r_{ji}(t). \quad (23)$$

When $t = 0$, we have $Q^r_i(0) = 0 < V + A^r_{max}$.

Assume that we have $Q^r_i(t) \le V + A^r_{max}$ when $t = t'$ ($t' \ge 0$). Since only one neighboring node can transmit to node i in time slot t', we denote it by node j. Consider the coefficient in front of $f^r_{ji}(t')$ in the objective function of $S2$.

- If $Q^r_i(t') < Q^r_j(t') - \sigma Y_{ji}(t')$, according to (23), we have

$$\begin{aligned} Q^r_i(t'+1) &\le Q^r_i(t') + f^r_{ji}(t') \\ &< Q^r_j(t') - \sigma Y_{ji}(t') + f^r_{ji}(t') \\ &\le Q^r_j(t') \\ &\le V + A^r_{max}, \end{aligned}$$

  The third inequality above can be proved in the following two cases.
  - If $Y_{ji}(t') = 0$, according to the solution to S1, we can know that $s^m_{ji}(t') = 0\ \forall m \in \mathcal{M}_j \cap \mathcal{M}_i$, and hence $f^r_{ji}(t') = 0$. Thus, the inequality holds.
  - If $Y_{ji}(t') \ge 1$, we have $\sigma Y_{ji}(t') \ge f^r_{ji}(t')$, as $f^r_{ji}(t') \le \max_{p\in\mathcal{N}, q\in\mathcal{T}_p}\{\frac{1}{\alpha} c^{max}_{pq}\Delta t\} = \sigma$ as defined before.
- If $Q^r_i(t') \ge Q^r_j(t') - \sigma Y_{ji}(t')$, according to our proposed solution to S2, we know that $f^r_{ji}(t') = 0$. Following (23), we have

$$Q^r_i(t'+1) \le Q^r_i(t') \le V + A^r_{max}.$$

  Therefore, we also have $Q^r_i(t) \le V + A^r_{max}$.

Based on the above results, we can see that an arbitrary queue $Q^r_i(t)$ is finite in any time slot. Thus, $\mathbf{Q}(t)$ is strongly stable by Definition 2.

Next, we prove the strong stability of $\mathbf{Y}(t)$, and specifically,

$$Y_{ij}(t) \le \sigma \cdot \max_{0\le k\le t} \sum_{r\in\mathcal{R}} f^r_{ij}(k) \quad (24)$$

for any $i \in \mathcal{N}, j \in \mathcal{T}_i$, by induction.

Consider an arbitrary queue $Y_{ij}(t)$.

When $t = 0$, we have $Y_{ij}(0) = 0$, and hence (24) holds.

Assume (24) holds when $t = t'$, i.e., $Y_{ij}(t') \le \sigma \cdot \max_{0\le k\le t'} \sum_{r\in\mathcal{R}} f^r_{ij}(k)$. Then, when $t = t' + 1$, we have

$$Y_{ij}(t'+1) = \max\Big\{Y_{ij}(t') - \frac{\sigma}{\alpha} \sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c^m_{ij}(t') s^m_{ij}(t')\Delta t, 0\Big\} + \sigma \sum_{r\in\mathcal{R}} f^r_{ij}(t').$$

If $Y_{ij}(t') > \frac{\sigma}{\alpha} \sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c^m_{ij}(t') s^m_{ij}(t')\Delta t$, with inequality (10), we have

$$\begin{aligned} Y_{ij}(t'+1) \le Y_{ij}(t') &\le \sigma \max_{0\le k\le t'} \sum_{r\in\mathcal{R}} f^r_{ij}(k) \\ &\le \sigma \max_{0\le k\le t'+1} \sum_{r\in\mathcal{R}} f^r_{ij}(k). \end{aligned}$$

If $Y_{ij}(t') \le \frac{\sigma}{\alpha} \sum_{m\in\mathcal{M}_i\cap\mathcal{M}_j} c^m_{ij}(t') s^m_{ij}(t')\Delta t$, then

$$Y_{ij}(t'+1) = \sigma \sum_{r\in\mathcal{R}} f^r_{ij}(t') \le \sigma \max_{0\le k\le t'+1} \sum_{r\in\mathcal{R}} f^r_{ij}(k).$$

Therefore, (24) holds when $t = t' + 1$ as well.

Since $\sum_{r\in\mathcal{R}} f^r_{ij}(t) \le \frac{1}{\alpha} c^{max}_{ij}\Delta t$, we have that $Y_{ij}(t) \le \frac{\sigma}{\alpha} c^{max}_{ij}\Delta t$ and hence always finite and strongly stable. ■

B. LOWER AND UPPER BOUNDS ON ψ^*_{P1}

In what follows, we obtain both lower and upper bounds on the optimal results of P1, i.e., ψ^*_{P1}.

Theorem 4: The scheduling policy obtained by our proposed approximation algorithm serves as a suboptimal yet feasible solution to P1, and the corresponding time-averaged expected amount of CPS services serves as a lower bound on the optimal result of P1, i.e., $\psi^*_{P1} \ge \psi_{P4}$.

Proof: Recall that the proposed approximation algorithm finds a feasible solution to P4 in each time slot, which satisfies all the constraints in P4, i.e., (2)–(3), (5), (8)–(10), and (21) as we have proved in the previous subsection. Since $\mathbf{Y}(t)$ is strongly stable, according to (16) and Theorem 2, we can know that $\mathbf{Y}(t)$ is also rate stable, i.e., the constraint (14) holds as well. Therefore, the scheduling policies we found in all time slots serve as a feasible solution to P3. Since P3 is equivalent to P1, the obtained scheduling policies serve as a feasible solution to P1 as well. Thus, the corresponding time-averaged expected amount of CPS services, i.e., ψ_{P4}, is no larger than the optimal result of P1, i.e., $\psi_{P4} \le \psi^*_{P1}$. ■

Next, we find an upper bound on ψ^*_{P1}. We first present a lemma as follows.

Lemma 3: The time-averaged expected amount of CPS services achieved by optimally solving P2, denoted by ψ^*_{P2}, is within a constant gap $\frac{B}{V}$ from the maximum time-averaged expected CPS services achieved by P1, i.e., ψ^*_{P1}. Particularly, we have

$$\psi^*_{P2} \ge \psi^*_{P1} - \frac{B}{V},$$

where B and V are defined in Section III-E.

Proof: Please refer to Appendix C, available in the online supplemental material, for the detailed proof. ■

Notice that P1 and P2 are both Mixed-Integer Linear Programming (MILP) problems. We relax P1 to a Linear Programming (LP) problem denoted by $\overline{P1}$, and formulate a corresponding online CPS service maximization problem denoted by $\overline{P2}$. We can see that $\overline{P2}$ is a relaxed LP problem based on P2, and can be easily solved. Denote the time averaged expected amount of CPS services obtained by

optimally solving $\overline{P1}$ and $\overline{P2}$ by $\psi^*_{\overline{P1}}$ and $\psi^*_{\overline{P2}}$, respectively. By Lemma 3, we can know that

$$\psi^*_{\overline{P2}} \geq \psi^*_{\overline{P1}} - \frac{B}{V}.$$

Since obviously we also have $\psi^*_{\overline{P1}} \geq \psi^*_{P1}$, we can arrive at the following result.

Theorem 5: The optimal result of P1 is upper bounded by $\psi^*_{P1} \leq \psi^*_{\overline{P2}} + \frac{B}{V}$, where $\psi^*_{\overline{P2}}$ can be obtained by optimally solving $\overline{P2}$, and B and V are defined in Section III-E.

Theorem 4 and Theorem 5 give a lower bound and an upper bound on the optimal result of P1, i.e, ψ^*_{P1}, respectively. Thus, we can not only better estimate the optimal result, but also know an upper bound on the gap between our lower bound and the optimal result. As will be shown later, the obtained lower bound and upper bound can be very close with properly chosen parameters. It indicates that our approximate solution is very effective.

In addition, note that we consider delay-insensitive CPS services in this study, and service delay is not our primary concern. On the other hand, we have proved in Theorem 3 that the queue backlog size at each node is finite in our proposed algorithm. Thus, the queueing delay at each node is finite and the processing delay of each CPS request is finite as well.

VI. SIMULATION RESULTS

In this section, we carry out extensive simulations to demonstrate the performance of our proposed algorithms in finding the lower and upper bounds on the optimal result ψ^*_{P1}, and the performance improvement due to employing the hybrid crowdsourcing architecture. Simulations are conducted under CPLEX 12.4 on a computer with a 2.27 GHz CPU and 24 GB RAM. In particular, we consider a square network of area 1000 $m \times 1000$ m, where a base station (BS) is located at the center, and 20 nodes are uniformly and randomly distributed. Five nodes have CPS service requests, i.e., certain amounts of computing instructions to be outsourced. Each instruction is 8 bytes long. We assume the number of instructions each node can process in each time slot, i.e., $O_i(t)$, is independently and uniformly distributed within $[5 \times 10^5, 10^6]$, while the number of instructions the BS can process is 10^7, a constant. We set A^r_{max}'s ($1 \leq r \leq R$) to 5×10^6. Besides, there are a basic spectrum band whose bandwidth is 1 MHz, and four other opportunistic spectrum bands, the bandwidth of each of which is independently and uniformly distributed within [0.5 MHz, 1 MHz] in each time slot. At each node, only a random subset of the spectrum bands are available, while the BS can utilize all of those bands. Some other important simulation parameters are listed as follows. The path loss exponent is 4 and $C = 62.5$. The noise power spectral density is $\eta = 3.34 \times 10^{-18}$ W/Hz at all nodes. The transmission power spectral density of nodes is $8.1 \times 10^9 \eta$, and the reception threshold and interference threshold are both 8.1 η. Thus, the transmission range and the interference range on all frequency bands are all equal to 500 m. All the results presented below are collected after the experiments run for a period of $T = 100$ time slots unless otherwise specified. We set the duration of each time slot to 1 s. In this paper, our proposed Lyapunov-based approximation algorithm guarantees the finite queue backlog size in the network, while most previous approaches do not. Thus, it is not very fair to compare with them. Moreover, the hybrid crowdsourcing architecture makes the problem studied in this paper a very unique problem. In fact, due to similar reasons, most previous works, e.g., [28]–[34], focus on the performance evaluation of their proposed algorithms and do not compare with other schemes, either. Nevertheless, we will compare the performance of the proposed hybrid crowdsourcing architecture with that of other architectures.

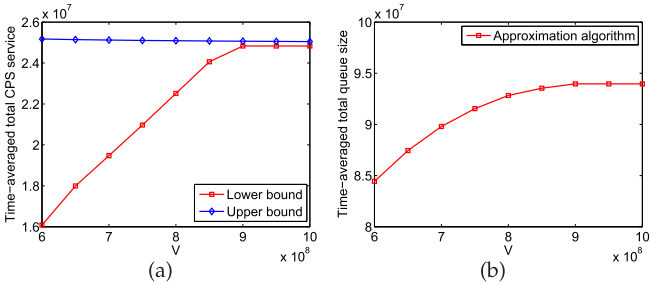


FIGURE 2. CPS service maximization (time-averaged results). (a) Time-averaged expected total CPS services. (b) Time-averaged expected total queue backlog of all nodes.

A. CPS SERVICE MAXIMIZATION

We first illustrate in Fig. 2(a) both the lower and upper bounds on the optimal result ψ^*_{P1} of the offline CPS service maximization problem. Recall that the lower bound is achieved by our proposed approximation algorithm, i.e., ψ_{P4}, and the upper bound is obtained by solving the relaxed problem $\overline{P2}$, i.e., $\psi^*_{\overline{P2}} + B/V$. We find that the lower and upper bounds get closer to each other as V increases, and that the lower bound first increases fast when V grows from 6×10^8 to 9×10^8 and then becomes stable. In particular, when $V = 9 \times 10^8$, the lower bound is 2.46×10^7 instructions, and the upper bound is 2.50×10^7 instructions. We can see that the lower bound is about 98.4% of the upper bound, i.e., the obtained bounds are very tight and hence very close to the optimal result. We then shown in Fig. 2(b) the time-averaged expected total queue backlog of all nodes incurred by our proposed approximation algorithm, i.e., the queue length corresponding to the lower bound result in Fig. 2(a). We find that the queue backlog in the approximation algorithm increases as V grows because a larger V means that we emphasize more on the CPS service maximization than on the queue size. From the results in Fig. 2(a) and Fig. 2(b), we can see that there is a tradeoff between CPS service and queue length in the design of our approximation algorithm. Specifically, in order to support more CPS services, we need to have a larger V which leads to a larger queue length.

The results in Fig. 2 are obtained by taking the average over 100 time slots as mentioned before. In Fig. 3, we show the results in each time slot. Specifically, we present in Fig. 3(a)

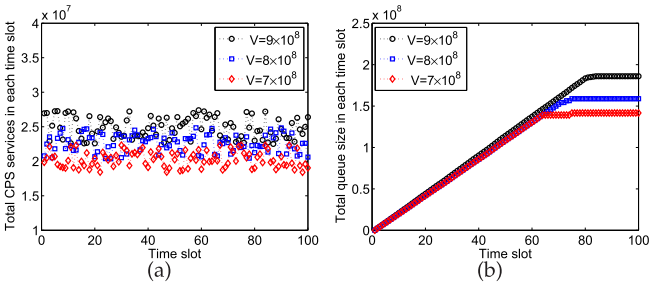


FIGURE 3. Online finite-queue-aware CPS service maximization (results in each time slot). (a) Expected total CPS services in each time slot. (b) Expected total queue size in each time slot.

the expected CPS services supported by the proposed approximation algorithm when $V = 7 \times 10^8$, 8×10^8, and 9×10^8. We can see that in each time slot the amount of CPS services that can be supported is larger under a larger V. The corresponding expected total queue backlogs of all nodes are shown in Fig. 3(b). We find that the expected total queue lengths first increase linearly with time, and then become steady after $t = 62$, 75, and 83, when $V = 7 \times 10^8$, 8×10^8, and 9×10^8, respectively. Since the expected total queue length is finite, each single queue in the network is finite as well in each time slot and hence strongly stable.

B. PERFORMANCE COMPARISON

One main advantage of the proposed hybrid crowdsourcing architecture, i.e., outsourcing the computing tasks to both the CPS nodes and the BS (or the central cloud), is that it can fully utilize the computing capabilities of CPS nodes and avoid communication congestion at the BS, and hence support more computing services. In the following, we compare the performance of the hybrid crowdsourcing architecture with that of another two outsourcing architectures, i.e., outsourcing to the BS and outsourcing to one-hop neighbors.

As shown in Fig. 4, we present the achievable time-averaged expected total CPS services under hybrid crowdsourcing, outsourcing to the BS, and outsourcing to 1-hop neighbors architectures with different network settings. Recall that we denote by N and R the number of users and the number of CPS service requests, respectively.

Fig. 4(a) demonstrates the achievable time-averaged expected total CPS services. We can see that with the same R and the same N, the performance of hybrid crowdsourcing is better than that of outsourcing to the BS and outsourcing to 1-hop neighbors due to the utilization of computing capabilities of both CPS nodes and the BS, and the avoidance of the communication congestion at the BS. Moreover, when N gets larger, the amount of services that can be supported increases under hybrid crowdsourcing and outsourcing to 1-hop neighbors architectures, but does not increase much under outsourcing to the BS architecture because it has limited network throughput capacity and does not utilize the computing capabilities at the other nodes.

Fig. 4(b) shows the time-averaged expected CPS services processed at the BS when $R = 3$ and 4, and N ranges from 4 to 20. We find that with the same R and the same N, the amount of CPS services processed at the BS under outsourcing to the BS architecture is higher than that under outsourcing to 1-hop neighbors architecture. This is because more nodes try to utilize the computing resource at the base station under outsourcing to the BS architecture. More interestingly, we also find that the amount of CPS services processed at the BS under hybrid crowdsourcing architecture is higher than that under outsourcing to the BS architecture. The reason is that rather than outsourcing all the computing tasks to the BS, the hybrid crowdsourcing architecture distributes computing tasks to both normal nodes and the BS, and thus can alleviate the contentions in the network and enhance the throughput to the BS. In other words, *the hybrid crowdsourcing architecture can make more use of the computation capability of the BS (or the central cloud) by distributing some computing tasks to other nodes.*

The time-averaged expected CPS services processed at the source nodes are shown in Fig. 4(c), which do not change when N increases. This is because all the source nodes can use all their available resources to provide some CPS services with no communication cost. Fig. 4(d) shows the time-averaged expected CPS services processed at the other nodes. Since under outsourcing to the BS architecture the computing resources at the other nodes are not utilized, the amount of CPS services handled at the other nodes is 0. We can also see that the other nodes can process more CPS services under hybrid crowdsourcing architecture than under outsourcing to 1-hop neighbors architecture, since in the latter case the services are only outsourced to the nodes that are within source nodes' 1-hop neighboring areas.

Fig. 5 demonstrates the time-averaged expected queue backlogs under hybrid crowdsourcing, outsourcing to the BS, and outsourcing to 1-hop neighbors architectures. From Fig. 5(b), we can see that with the same N and the same R, the queue length at the BS under hybrid crowdsourcing architecture is smaller than that under outsourcing to the BS architecture, since not all the computing tasks are sent to the BS. Besides, the queue length at the BS under hybrid crowdsourcing architecture is larger than that under outsourcing to 1-hop neighbors architecture because the services supported at the BS is larger in the former case as shown in Fig. 4(b).

From Fig. 5(c), we find that with the same R and the same N, the queue length at the source nodes under hybrid crowdsourcing architecture is larger than that under the other two architectures. This is because according to (4) the queue length at a source node is at least the amount of CPS services it requested, i.e., $v_r(t)$, and the hybrid crowdsourcing architecture can support the most CPS services as shown by Fig. 4(a).

Fig. 5(d) shows the queue length at all the other nodes. We can see that the queue length under hybrid crowdsourcing architecture is smaller than that under outsourcing to the BS architecture, because under hybrid crowdsourcing architecture other nodes can also process some computing

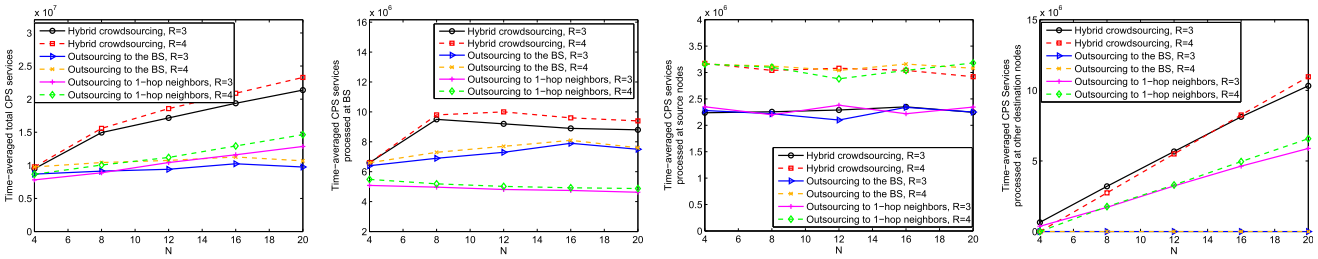


FIGURE 4. Achievable CPS services under hybrid crowdsourcing, outsourcing to the BS, and outsourcing to 1-hop neighbors architectures. (a) Time-averaged expected total CPS services. (b) Time-averaged expected CPS services processed at the BS. (c) Time-averaged expected CPS services processed at the source nodes. (d) Time-averaged expected CPS services processed at all the other nodes.

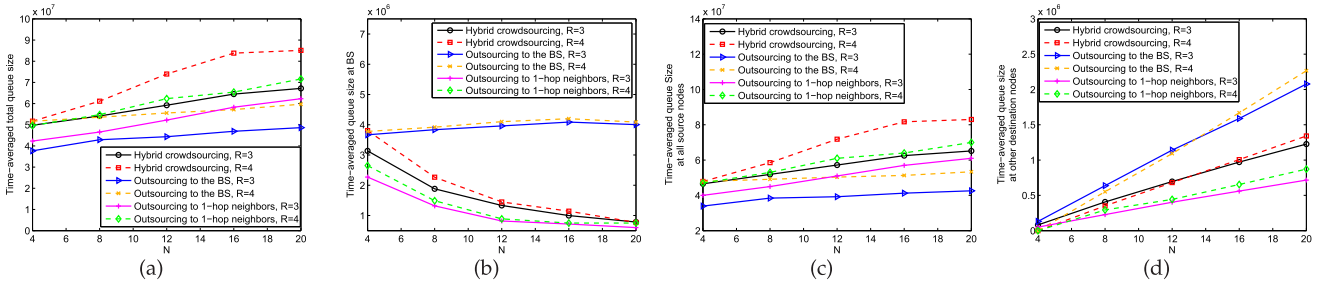


FIGURE 5. Queue backlogs under hybrid crowdsourcing, outsourcing to the BS, and outsourcing to 1-hop neighbors architectures. (a) Time-averaged expected total queue size. (b) Time-averaged expected queue size at the BS. (c) Time-averaged expected queue size at source nodes. (d) Time-averaged expected queue size at all the other nodes.

tasks. Besides, the queue length under hybrid crowdsourcing architecture is still larger than that under outsourcing to 1-hop neighbors architecture because the services processed at the other nodes is fewer in the latter case as shown in Fig. 4(d). Since the queue at the source nodes is the longest among these three queues, the time-averaged expected total queue length under hybrid crowdsourcing architecture is the largest among that under all the three architectures as shown in Fig. 5(a).

VII. CONCLUSION

In this paper, we employ a hybrid crowdsourcing architecture, which utilizes both the central cloud and some other CPS nodes to provide computing resources for resource-limited CPS nodes to accomplish their workloads. We have studied how to support CPS services by jointly considering computing resource allocation, routing, and link scheduling. Specifically, we investigate the offline CPS service maximization problem while considering network dynamics, e.g., dynamic spectrum and computing resource availability, and the finite network queues sizes. A feasible lower bound has been found by developing an approximation algorithm to solve an online CPS service maximization problem, which has been proved to guarantee network strong stability. An upper bound has also been obtained. We have demonstrated that the lower and upper bounds are very tight and hence close to the optimal result. Besides, we find that our proposed hybrid crowdsourcing architecture can support more CPS services than other outsourcing architectures. The reason is not only that hybrid crowdsourcing architecture can utilize both the central cloud resources and CPS nodes' computing resources, but also that by distributing some tasks to other nodes, hybrid crowdsourcing architecture can deliver more computing tasks to the base station and hence make more use of the central cloud's computation capability.

REFERENCES

[1] W. Wolf, "The good news and the bad news," *IEEE Comput.*, vol. 40, no. 11, pp. 104–105, Sep. 2007.

[2] X. Cao, P. Cheng, J. Chen, and Y. Sun, "An online optimization approach for control and communication co-design in networked cyber-physical systems," *IEEE Trans. Ind. Inf.*, vol. 9, no. 1, pp. 439–450, Jan. 2013.

[3] X. Cao, J. Chen, Y. Zhang, and Y. Sun, "Development of an integrated wireless sensor network micro-environment monitoring system," *ISA Trans.*, vol. 47, no. 3, pp. 247–255, Jul. 2008.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, pp. 50–58, Apr. 2010.

[5] (2011 Jan.). *NIST Definition of Cloud Computing* [Online]. Available: http://www.nist.gov/itl/cloud/

[6] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervas. Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.

[7] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. ACM MobiHoc*, Jun. 2012, pp. 1–10.

[8] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *Proc. ACM MobiCom*, Sep. 2003, pp. 1–14.

[9] H. Zhai and Y. Fang, "Impact of routing metrics on path capacity in multirate and multihop wireless ad hoc networks," in *Proc. IEEE ICNP*, Nov. 2006, pp. 1–10.

[10] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proc. IEEE Conf. Decision Control*, Dec. 2004, pp. 1484–1489.

[11] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 302–315, Apr. 2006.

[12] Y. T. Hou, Y. Shi, and H. D. Sherali, "Optimal spectrum sharing for multi-hop software defined radio networks," in *Proc. 26th IEEE INFOCOM*, May 2007, pp. 1–9.

[13] Y. T. Hou, Y. Shi, and H. D. Sherali, "Spectrum sharing for multi-hop networking with cognitive radios," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 1, pp. 146–155, Jan. 2008.

[14] J. Tang, S. Misra, and G. Xue, "Joint spectrum allocation and scheduling for fair spectrum sharing in cognitive radio wireless networks," *J. Comput. Netw. (Elsevier)*, vol. 52, no. 11, pp. 2148–2158, Aug. 2008.

[15] Z. Feng and Y. Yang, "Joint transport, routing and spectrum sharing optimization for wireless networks with frequency-agile radios," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 1–9.

[16] L. Ding, T. Melodia, S. N. Batalama, and J. D. Matyjas, "Distributed routing, relay selection, and spectrum allocation in cognitive and cooperative Ad Hoc networks," in *Proc. IEEE SECON*, Jun. 2010, pp. 1–9.

[17] M. Pan, C. Zhang, P. Li, and Y. Fang, "Joint routing and link scheduling for cognitive radio networks under uncertain spectrum supply," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 2237–2245.

[18] M. Pan, P. Li, and Y. Fang, "Cooperative communication aware link scheduling for cognitive vehicular Ad-Hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 4, pp. 760–768, May 2012.

[19] L. Gao, X. Wang, and Y. Xu, "Multi-radio channel allocation in multi-hop wireless networks," *IEEE Trans. Mobile Comput.*, vol. 8, no. 11, pp. 1454–1468, Nov. 2009.

[20] L. Zhou, X. Wang, Y. Li, B. Zheng, and B. Geller, "Optimal scheduling for multiple description video streams in wireless multihop networks," *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 534–536, Jul. 2009.

[21] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," in *Proc. 25th IEEE INFOCOM*, Apr. 2006, pp. 1–13.

[22] Y.-H. Lin and R. L. Cruz, "Opportunistic link scheduling, power control, and routing for multi-hop wireless networks over time varying channels," in *Proc. Allerton Conf. Commun. Control Comput.*, Sep. 2005, pp. 1–216.

[23] L. Ding, T. Melodia, S. N. Batalama, J. D. Matyjas, and M. J. Medley, "Cross-layer routing and dynamic spectrum allocation in cognitive radio ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 4, pp. 1969–1979, May 2010.

[24] L. Ding, T. Melodia, S. N. Batalama, and J. D. Matyjas, "Distributed routing, relay selection, and spectrum allocation in cognitive and cooperative Ad Hoc networks," in *Proc. 7th IEEE SECON*, Jun. 2010, pp. 1–9.

[25] N. Gatsis, A. Ribeiro, and G. B. Giannakis, "A class of convergent algorithms for resource allocation in wireless fading networks," *IEEE Trans. Wireless Commun.*, vol. 9, no. 5, pp. 1808–1823, May 2010.

[26] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1. 2nd ed. Belmont, MA, USA: Athena Scientific, 2007.

[27] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. San Mateo, CA, USA: Morgan, 2010.

[28] V. J. Venkataramanan, X. Lin, L. Ying, and S. Shakkottai, "On scheduling for minimizing end-to-end buffer usage over multihop wireless networks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[29] H. Li, W. Huang, Z. L. C. Wu, and F. C. M. Lau, "Utility-maximizing data dissemination in socially selfish cognitive radio networks," in *Proc. 8th IEEE MASS*, Oct. 2011, pp. 212–221.

[30] D. Xue and E. Ekici, "Guaranteed opportunistic scheduling in multihop cognitive radio networks," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 2984–2992.

[31] M. J. Neely, "Intelligent packet dropping for optimal energy-delay tradeoffs in wireless downlinks," *IEEE Trans. Autom. Control*, vol. 54, no. 3, pp. 565–579, Mar. 2009.

[32] M. J. Neely, "Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1728–1736.

[33] J. J. Jaramillo and R. Srikant, "Optimal scheduling for fair resource allocation in Ad Hoc networks with elastic and inelastic traffic," *IEEE/ACM Trans. Netw.*, vol. 19, no. 4, pp. 1125–1136, Aug. 2011.

[34] R. Urgaonkar and M. J. Neely, "Opportunistic scheduling with reliability guarantees in cognitive radio networks," *IEEE Trans. Mobile Comput.*, vol. 8, no. 6, pp. 766–777, Jun. 2009.

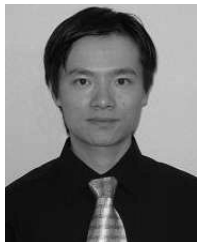
[35] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[36] S. Ren, Y. He, and F. Xu, "Provably-efficient job scheduling for energy and fairness in geographically distributed data centers," in *Proc. 32nd IEEE ICDCS*, Jun. 2012, pp. 22–31.

[37] X. Qiu, H. Li, C. Wu, Z. Liy, and F. C. Lau, "Cost-minimizing dynamic migration of content distribution services into hybrid clouds," in *Proc. IEEE INFOCOM*, Apr. 2012, pp. 2571–2575.

[38] Y. Pochet and L. Wolsey, *Production Planning by Mixed Integer Programming*. New York, NY, USA: Springer-Verlag, Apr. 2006.

MING LI received the B.E. degree in electrical engineering from Sun Yat-Sen University, Kaohsiung, China, in 2007, and the M.E. degree in electrical engineering from the Beijing University of Posts and Communications, Beijing, China, in 2010. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Mississippi State University. Her current research interests include cross-layer optimization, and security and privacy in cognitive radio networks, cloud computing, and smart grids.



PAN LI received the B.E. degree in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2005, and the Ph.D. degree in electrical and computer engineering from University of Florida, Gainesville, FL, USA, in 2009. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Mississippi State University. His current research interests include capacity and connectivity investigation, cross-layer optimization and design, and security and privacy in wireless networks, complex networks, cyber-physical systems, mobile computing, and cloud computing. He has been serving as an Editor for the *IEEE Journal on Selected Areas in Communications – Cognitive Radio Series* and the *IEEE Communications Surveys and Tutorials*, a Feature Editor for the *IEEE Wireless Communications*, a Guest Editor for the *IEEE Wireless Communications SI on User Cooperation in Wireless Networks*, and a TPC Co-Chair of *Wireless Networking Symposium*, the IEEE ICC in 2013. He received the National Science Foundation CAREER Award in 2012 and is a member of ACM.