Stochastic Optimal Control for Participatory Sensing Systems with Heterogenous Requests

Tong Liu†, Yanmin Zhu†, Qian Zhang‡, Athanasios Vasilakos†
†Shanghai Jiao Tong University;
‡Hong Kong University of Science and Technology;
†National Technical University of Athens
†{liutong25691640, yzhu}@sjtu.edu.cn; ‡qianzh@cse.ust.hk; †vasilako@ath.forthnet.gr

Abstract—We consider the crucial problem of maximizing the system-wide performance which takes into account request processing throughput, smartphone user experience and system stability in a participatory sensing system with cooperative smartphones. Three important controls need to be made, i.e., 1) request admission control, 2) task allocation, and 3) task scheduling on smartphones. It is highly challenging to achieve the optimal system-wide performance, given arbitrary unknown arrivals of sensing requests, intrinsic tradeoff between request processing throughput and smartphone user experience degradation, and heterogeneous requests. Little existing work has studied this crucial problem of maximizing the system-wide performance of a participatory sensing system as a whole. In response to the challenges, we propose an optimal online control approach to maximize the system-wide performance of a participatory sensing system. Exploiting the stochastic Lyapunov optimization techniques, it derives the optimal online control strategies for request admission control, task allocation and task scheduling on smartphones. The most salient feature of our approach is that the achieved system-wide performance is arbitrarily close to the optimum, despite unpredictable and arbitrary request arrivals. Rigorous theoretical analysis and comprehensive simulation evaluation demonstrate the efficacy of our online control approach.

I. INTRODUCTION

In the past few years, participatory sensing [1][2] has become increasingly important. It has created unprecedented opportunities for smartphone users to contribute their sensed data and collaboratively provide large-scale sensing services. By collecting and sharing data through participatory sensing, one is able to have a good view over a large-scale phenomenon in real time yet with a low cost. A number of such participatory sensing applications [3][4] have been developed, such as noise mapping [3] and environment monitoring [4].

A typical participatory sensing system consists of a platform residing in the cloud and many mobile smartphones equipped with various sensors, as shown in Fig.1. Sensing requests arrive at the platform dynamically. The platform allocates these sensing requests to smartphones for processing. It is important to emphasize that it incurs considerable costs when a resource-constrained smartphone performs sensing tasks, e.g., energy consumption (by the smartphone processor and corresponding sensors [5]), human intervention, etc. As a result, the smartphone user experience is degraded by processing sensing tasks. In general, if more sensing tasks are simultaneously processed on a smartphone, the user experience is degraded more significantly. In this work, we consider heterogeneous sensing requests. A sensing request may differ from each other in required resources and hence cause different costs.

In this paper, we consider a practical type of participatory sensing system with cooperative smartphones [4]. The cooperative smartphones are willing to provide sensing services to the system, e.g., by reaching an agreement with the platform. Such a participatory system as a whole aims at maximizing a system-wide performance, taking into account request processing throughput, system stability, smartphone user experience and fairness among smartphones. A good system-wide performance requires the system to process more requests, retain acceptable experience for smartphone users and maintain good system stability.

To maximize the system-wide performance in a participatory sensing system, three important layers of controls need to be made. (1) Request admission control. A serious problem for a participatory sensing system is the imbalanced demand and supply: the system capacity cannot sustain the ever increasing sensing requests. If the system was overloaded, it would be in an instable state and sensing requests would suffer poor performance like long processing delays. Therefore, it is indispensable for the system to make admission control, deciding whether to accept an incoming sensing request into the system for processing. (2) Task allocation. As sensing requests are accepted into the system, the platform needs to
allocate them to smartphones. Task allocation should balance the tasks processed by each smartphone and make sure each smartphone is not congested due to too many tasks; otherwise, these sensing tasks may suffer long processing delays. (3) Task scheduling on smartphones. Each smartphone is allocated a pool of sensing tasks. As mentioned before, if a smartphone processes too many tasks simultaneously, it contributes to a high processing throughput but the user experience of the smartphone is degraded significantly. Thus, task scheduling is important to balance request processing throughput and smartphone user experience.

However, it is particularly challenging to achieve the optimal system-wide performance because of the unique characteristics of participatory sensing systems. First, sensing requests may arrive at the system at any time, which is unknown beforehand. In the meanwhile, however, the controls must be made on the fly. Second, the three layers of controls are coupled and there is interplay among the controls. Third, there is an intrinsic tradeoff between system processing throughput and system stability as well as smartphone user experience. Finally, sensing requests are heterogenous, and should be handled differently when control decisions are made.

Little existing work has studied the crucial problem of maximizing the system-wide performance of a participatory sensing system as a whole. Most of existing studies simply assume that all sensing requests can be served by the system and the aggregate demand is lower than the system capacity, which is not true in reality. A number of research efforts [6][7][8][9] have proposed incentive mechanisms with static sensing requests. They typically consider strategic smartphones which take actions solely for maximizing their own payoffs. These mechanisms fail to maximize system-wide performance in the participatory sensing system with online arrivals of requests.

In response to the challenges, we propose an optimal online control approach for a participatory sensing system with cooperative smartphones. Exploiting the stochastic Lyapunov optimization techniques [10], it first transforms the original system to a queuing system with virtual queues for making admission control. Then, it constructs a new objective taking both system-wide performance maximization and system stability into consideration. By solving the new objective, we derive the online control strategies for request admission control, task allocation and task scheduling on smartphones. Rigorous theoretical analysis and comprehensive simulation evaluation demonstrate the efficacy of our online control approach.

The major contributions are summarized as follows.

- It is the first attempt, to the best of our knowledge, to investigate the optimal online control problem of maximizing the system-wide performance in a participatory sensing system with heterogeneous sensing requests.
- We propose an optimal control approach based on the stochastic Lyapunov optimization technique. Requiring no knowledge about arrivals of sensing requests, it achieves the system-wide performance that is arbitrarily close to the optimum. In addition, we provide a distributed implementation, which enables each smartphone to independently schedule its own tasks, also reserving private smartphone information.
- We have performed both rigorous theoretical analysis and comprehensive simulation study, and the results confirm the superiority of our algorithm compared with other online control algorithms.

The remainder of the paper proceeds as follows. The system model, control framework and problem formulation are presented in Section II. Section III delves into the details of our stochastic optimal control algorithm. The performance of the proposed online control algorithm is evaluated in Section IV. We discuss related work in Section V. We finally conclude the paper in Section VI.

II. SYSTEM MODEL AND ONLINE CONTROL FRAMEWORK

A. Participatory Sensing System Model

We consider a participatory sensing system consisting of a platform residing in the cloud, and a number of distributed cooperative smartphones. Smartphones are willing to provide sensing services. We consider the system has a stable set of smartphones, denoted by \( N = \{1,2,\cdots,n\} \). For ease of discussion, we divide the time into slots of equal size, and each time slot \( t \) is in \( \{0,1,\cdots,\tau,\cdots\} \).

We consider heterogeneous sensing requests which may differ in required sensors, required smartphone resources (such as processor and memory) or human interventions. There are \( m \) types of sensing requests, and the set of types is denoted by \( \mathcal{M} = \{1,2,\cdots,m\} \). Each request is associated with a single type. Each request of type \( i \) incurs the same amount \( \nu_i \) of costs on all smartphones.

In each time slot \( t \), a number of different types of sensing requests may arrive at the system. Let \( R_i(t) \) denote the number of all arrived sensing requests of type \( i \in \mathcal{M} \) in time slot \( t \). Considering arbitrary arrivals of sensing requests, we assume that the numbers of sensing requests arriving at different slots are independent and identically distributed (i.i.d.). We denote the expected rate of sensing requests of type \( i \) in each time slot as \( \bar{R}_i = \mathbb{E}\{R_i(t)\} \). In reality, the number of arriving requests \( R_i(t) \) in each time slot is highly dynamic and can spike abruptly. We assume that each type of requests has a certain peak which is denoted by \( R_i^{\max}, \forall i \in \mathcal{M} \). Therefore, \( R_i(t) \leq R_i^{\max}, \forall i \in \mathcal{M}, \forall t \in \{0,1,\cdots\} \).

B. Online Control Framework

We next introduce a three-layer online control framework that addresses three important control decisions in a participatory sensing system, as shown in Fig. 2.

- **Request admission control.** The framework places the first-layer control module for request admission. In each time slot, many heterogeneous sensing requests arrive at the system. For each type of sensing requests, the request admission control module decides the number of sensing requests that are accepted to be processed in the system. We denote the number of admitted sensing requests of type \( i \) in each time slot as \( A_i(t), \forall i, \forall t \). As the number of admitted sensing requests...
Fig. 2. The three-layer online control framework, which consists of admission control, task allocation and task scheduling on smartphones. On each smartphone, a queue is maintained for each type of tasks, which holds all pending tasks of a certain type.

must be equal or less than the number of arriving requests, we have $0 \leq A_i(t) \leq R_i(t), \forall i, \forall t$.

**Task allocation.** The framework places the second-layer control module for task allocation. As soon as the system has admitted a sensing request of type $i$, the task allocation module decides which smartphone the task is assigned to. Let the number of sensing tasks of type $i$ allocated to smartphone $j$ in time slot $t$ be denoted as $A_{ij}(t), \forall i \in \mathcal{M}, \forall j \in \mathcal{N}$. It is clear that $\sum_{j \in \mathcal{N}} A_{ij}(t) = A_i(t), \forall i, \forall t$.

As it is hardly possible for a smartphone to immediately complete all allocated sensing tasks, each smartphone maintains a separate queue for each type of sensing tasks. The queue holds all pending tasks of the specific type. The sensing tasks waiting in a queue are served on a first-in-first-out (FIFO) basis. We define the queue backlog for the queue of type $i$ on smartphone $j$, denoted by $Q_{ij}(t)$, as the number of pending tasks waiting in the queue.

**Task scheduling:** The framework places the third-layer control module for task scheduling. In each time slot, the task scheduling module decides which queues pop tasks for processing. It is worth noting that since a smartphone typically has limited resources, it is impractical for a smartphone to simultaneously process the pending tasks of each type. Thus, it is crucial to choose a proper set of sensing tasks for processing in each time slot. Processing the set of sensing tasks should not degrade too much of the user experience. Note that we assume for each smartphone, it takes a single time slot to process a sensing task of any type. This is practical since we can always divide a larger sensing job into multiple sensing tasks of equal size that can be processed in one time slot. We introduce an indication variable $s_{ij}(t)$ to denote the selection result for each type $i$.

$$s_{ij}(t) = \begin{cases} 1, & \text{a task of type } i \text{ on } j \text{ is processed}, \\ 0, & \text{no tasks of type } i \text{ on } j \text{ are processed}. \end{cases}$$

For the queue maintaining tasks of type $i$ on smartphone $j$, its queue backlog $Q_{ij}(t)$ can be analyzed as follows. Recall that in each time slot the number of sensing tasks pushed into the queue is $A_{ij}(t)$. Then, we can derive that,

$$Q_{ij}(t+1) = \max\{Q_{ij}(t) - s_{ij}(t), 0\} + A_{ij}(t). \quad (2)$$

**C. System-Wide Performance and Problem Formulation**

With the online control framework, we finally characterize the system-wide performance of a participatory system, jointly considering utility of request processing throughput, and smartphone user experience degradation.

The request processing throughput of a participatory sensing system in time slot $t$ is the sum of admitted requests $A_i(t), \forall i \in \mathcal{M}$, denoted by $\sum_{i \in \mathcal{M}} A_i(t)$. Let $\pi_i, \forall i \in \mathcal{M}$ denote the averaged throughput of type $i$ over time slots, such that $\pi_i = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E(A_i(\tau)), \forall i \in \mathcal{M}$.

**Throughput utility.** We consider that the participatory sensing system can gain some utility by successfully processing an amount of sensing requests. As the achieved throughput is dynamic over time, we define throughput utility based on the time-averaged throughput $\sum_{i \in \mathcal{M}} \pi_i$. Intuitively, the larger throughput is achieved, the more utility is gained. The throughput utility is formulated as a linear function $\mu \sum_{i \in \mathcal{M}} \pi_i$, where system parameter $\mu$ represents the ratio of utility to throughput. Maximizing the throughput utility is one of the most important objectives for improving the system-wide performance.

**Smartphone user experience degradation.** It is intuitive that more smartphone user experience is degraded as the total cost of sensing tasks being processed is higher. In addition, the decreasing rate of a smartphone user experience becomes larger as the cost increases. Accordingly, we define user experience degradation as follows,

**Definition 1.** The user experience degradation $D_j(t)$ of smartphone $j$ in time slot $t$ is defined as

$$D_j(t) = \beta_j \left( \sum_{i \in \mathcal{M}} \nu_i s_{ij}(t) \right)^\theta,$$  \quad (3)$$

where $\theta \geq 2$, $\beta_j$ is a smartphone-specific positive coefficient for characterizing the user experience sensitivity to the total cost. With a higher $\beta_j$, the user experience degrades more quickly.

**Insight:** $\theta$ characterizes the sensitivity of a smartphone user experience to the total cost. For ease of presentation, we adopt $\theta = 2$ in the following. However, our design can be easily extended to other settings of $\theta$.

Accordingly, the time average of user experience degradation $D_j(t)$ is denoted by $\bar{d}_j = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E(D_j(\tau)), \forall j \in \mathcal{N}$.

**System-wide performance.** With the availability of throughput utility and smartphone experience degradation, we define the system-wide performance, denoted by $\varphi$.

$$\varphi = \mu \sum_{i \in \mathcal{M}} \pi_i - \sum_{j \in \mathcal{N}} \bar{d}_j \quad (4)$$
Then, we can present the problem of maximizing the system-wide performance in the participating sensing system, as defined in the following.

**Definition 2.** The problem of maximizing the time-averaged system-wide performance for a participatory sensing system is defined as

\[
\max \quad \mu \sum_{i \in \mathcal{M}} \bar{\pi}_i - \sum_{j \in \mathcal{N}} \underline{d}_j \\
\text{s.t.} \quad 0 \leq \bar{\pi}_i \leq \bar{\pi}_i, \forall i \in \mathcal{M} \\
\text{constraint} \quad \underline{\pi}_i \leq \underline{\pi}_i, \forall i \in \mathcal{M}.
\]

**Insight:** The first constraints exist as \( \mathbb{E}(A_i(t)) \leq \mathbb{E}(R_i(t)) \) holds in each time slot. The second constraints exist as the number of requests completed in one time slot cannot be larger than the number of smartphones.

It is challenging to make optimal control decisions for maximizing the system-wide performance. There are two major challenges. First, sensing requests arrive at the system dynamically and unpredictably. This suggests that the key parameters affecting the system-wide performance would be time varying. This rules out the applicability of offline algorithms. Second, the number of smartphones can be huge and the associated computation complexity of a central algorithm can be exceptionally high. The heterogeneous nature of sensing requests further increases the computation complexity.

### III. Stochastic Optimal Control

#### A. Overview

In response to the aforementioned challenges, we design a stochastic optimal control approach based on the Lyapunov optimization [11][10]. As a result of this approach, we propose an online and distributed algorithm, which holistically determines all control decisions in the three-layer control framework.

Exploiting the Lyapunov optimization for solving the problem in (5), our online stochastic control approach consists of four key components. First, it converts the time-average constraint (6) into a queue stability problem by using the technique introduced in [12]. Thus, a participatory sensing system is modeled as a standard queuing system. Second, Lyapunov function and Lyapunov drift are derived for the queuing system according to Lyapunov optimization [10]. These two terms are used to characterize the stability of a queuing system. Third, taking both system-wide performance maximization and system stability into account, we reformulate the objective and propose an optimal online control algorithm. The algorithm computes all control decisions for the three layers of controls. Fourth, we present a distributed implementation of the online algorithm in which each smartphone can independently schedule its own tasks.

#### B. Virtual Queue Modeling

According to the Lyapunov optimization [12], we first transform our problem in (5) to a queue stability problem by introducing auxiliary variables \( \alpha_i(t) \) for each \( A_i(t), \forall i \in \mathcal{M} \). The values of \( \alpha_i(t) \) are decided in each time slot under the constraint \( 0 \leq \alpha_i(t) \leq R_{i, \text{max}} \). The time average of \( \alpha_i(t) \) is defined as \( \bar{\alpha}_i = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}(\alpha_i(\tau)), \forall i \in \mathcal{M} \).

The transformed problem is as follows,

\[
\max \quad \sum_{i \in \mathcal{M}} \mu \bar{\pi}_i - \sum_{j \in \mathcal{N}} \underline{d}_j \\
\text{s.t.} \quad \bar{\pi}_i \leq \underline{\pi}_i, \forall i \in \mathcal{M} \\
\text{constraint} \quad \underline{\pi}_i \leq \underline{\pi}_i, \forall i \in \mathcal{M}.
\]

where the definition of queue stability is involved in the following definition.

**Definition 3 (Queue and System Stability).** A queue \( Q \) is strongly stable if and only if

\[
\lim_{t \to \infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}(Q(\tau)) < \infty,
\]

where \( Q(\tau) \) is the backlog of the queue in time slot \( \tau \). A system is stable if and only if all queues in the system are strongly stable.

**Insight:** The transformation can be intuitively explained as follows. The constraints (6) and (7) are automatically satisfied under arbitrary control strategies when (10) and (11) hold. The results of the original problem can be obtained from the transformed problem by choosing \( \bar{\pi}_i = \overline{\pi}_i, \forall i \).

To ensure the time-averaged constraints in (9) are satisfied, we use virtual queue [12] \( H_i \) for each type \( i \). Similar to \( Q_i(t) \), the queue backlog can be expressed as follows,

\[
H_i(t+1) = \max[H_i(t) - A_i(t), 0] + \alpha_i(t).
\]

Here, \( \alpha_i(t) \) and \( A_i(t) \) can be respectively viewed as the arriving rate and the outgoing rate of virtual queue \( H_i \) in time slot \( t \). We prove that the transformation is valid in the following.

**Lemma 1.** The constraints in (9) are satisfied under the condition that the virtual queues \( H_i(t) \) are strongly stable, which are updated by (12) in each time slot.

**Proof:** As virtual queues are updated by (12), the inequality \( H_i(t+1) \geq H_i(t) - A_i(t) + \alpha_i(t) \) holds through taking out \( \max \) function. After being summed over time slots \( t \in \{0, 1, \ldots, t-1\} \) and divided by \( t \), the inequality leads to

\[
\frac{H_i(t) - H_i(0)}{t} + \frac{1}{t} \sum_{\tau=0}^{t-1} A_i(\tau) \geq \frac{1}{t} \sum_{\tau=0}^{t-1} \alpha_i(\tau),
\]

with \( H_i(0) = 0 \). By first taking expectations and then taking limits as \( t \) tends to infinity, we obtain

\[
\lim_{t \to \infty} \frac{\mathbb{E}(H_i(t))}{t} + \underline{\pi}_i \geq \overline{\pi}_i.
\]

Moreover, if virtual queues are strongly stable, we have \( \lim_{t \to \infty} \mathbb{E}(H_i(t))/t = 0 \) from Definition 3. Combining with (13), the constraints \( \underline{\pi}_i \leq \overline{\pi}_i, \forall i \in \mathcal{M} \) are satisfied. Therefore, the lemma is proved.

Based on Lemma 1, the participatory sensing system has
been converted into a queuing system with two layers of queues, and task allocation is performed between them, as shown in Fig. 2. The first layer consists of virtual queues, denoted by $H(t) = (H_i(t))$, while the second layer consists of actual queues, denoted by $Q(t) = (Q_{ij}(t))$. We use $\Theta(t) = (Q(t); H(t))$ to denote the matrix of all queues.

C. Problem Reformulation Based on Lyapunov Optimization

In addition to maximizing the system-wide performance, a queuing system also takes system stability into account. We first characterize the tradeoff between the queue stability and system-wide performance based on Lyapunov optimization [10], and then reformulate the objective by deriving the upper bound of the tradeoff.

For a queue matrix $\Theta(t)$, the Lyapunov function $L(\Theta(t))$ [10] is defined as follows,

$$L(\Theta(t)) = \frac{1}{2}[\sum_{i \in M} H_i^2(t) + \sum_{i \in M} \sum_{j \in N} Q_{ij}^2(t)], \quad (14)$$

which is a scalar measurement of the matrix, respecting the overall situation of queue congestion in a queuing system. Specially, a small value of $L(\Theta(t))$ implies that all queues included in $\Theta(t)$ have small queue backlogs. Otherwise, a large value of $L(\Theta(t))$ implies that at least one queue in $\Theta(t)$ has a large backlog. A system maintaining a small $L(\Theta(t))$ over time slots $t = 0, 1, \ldots, \tau, \ldots$ implies its strong stability.

Next, we define the one-step conditional Lyapunov drift

$$\Delta(\Theta(t)) = \mathbb{E}[L(\Theta(t + 1)) - L(\Theta(t))|\Theta(t)]. \quad (15)$$

$\Delta(\Theta(t))$ represents the expectation of the difference between the values of Lyapunov function in consecutive time slots. The value of $\Delta(\Theta(t))$ being less than zero means the Lyapunov function is pushed towards a smaller value, compared with that of the previous time slot. In other words, the smaller value of $\Delta(\Theta(t))$ we obtain in time slot $t$, the stronger stability the system is pushed towards.

Aiming at minimizing the Lyapunov drift and maximizing the system-wide performance, we define the drift-minus-utility function combining the two folds as follows.

**Definition 4.** The drift-minus-utility function in each time slot $t$ is defined as

$$\Delta(\Theta(t)) - V \mathbb{E}\{\sum_{i \in M} \alpha_i(t) - \sum_{j \in N} D_j(t)|\Theta(t)\} \leq B - \sum_{i \in M} \mathbb{E}[H_i(t)A_i(t) - \sum_{j \in N} Q_{ij}(t)A_{ij}(t)|\Theta(t)] \quad (18)$$

Combining the above inequalities, the upper bound of $\Delta(\Theta(t))$ is obtained as (17), by defining

$$B = \frac{|M||N|+3\sum_{i \in M} (R_{i_{max}}^2)^2}{2}.$$

With Lemma 2, the upper bound of the drift-minus-utility is obtained immediately, which is expressed in the following inequality.

**Lemma 2.** Under arbitrary control decisions, the following inequality holds with $B = \frac{|M||N|+3\sum_{i \in M} (R_{i_{max}}^2)^2}{2}$.

$$\Delta(\Theta(t)) \leq B - \sum_{i \in M} H_i(t)\mathbb{E}\{A_i(t) - \alpha_i(t)|\Theta(t)\} \quad (17)$$

**Proof:** According to the definition of $\Delta(\Theta(t))$, there exists $\Delta(\Theta(t)) = \frac{1}{2}\mathbb{E}[\sum_{i \in M} H_i^2(t) - H_i^2(t)] + \sum_{i \in M} \sum_{j \in N} [Q_{ij}^2(t + 1) - Q_{ij}^2(t)]|\Theta(t)]$. Based on the fact that $(\max[a, b, 0] + c)^2 \leq a^2 + b^2 + c^2 - 2a(b - c), \forall a, b, c \geq 0$ and the definitions in (12)(2)(1), we have

$$H_i(t + 1) - H_i(t) \leq A_i^2(t) + \alpha_i^2(t) - 2H_i(t)A_i(t) - \alpha_i(t) \leq 2(R_{i_{max}}^2)^2 - 2H_i(t)A_i(t) - \alpha_i(t),$$

$$Q_{ij}^2(t + 1) - Q_{ij}^2(t) \leq s_{ij}^2(t) + A_{ij}^2(t) - 2Q_{ij}(t)s_{ij}(t) - A_{ij}(t),$$

$$\sum_{j \in N} [s_{ij}^2(t) + A_{ij}^2(t)] \leq |N| + (R_{i_{max}}^2)^2.$$

D. Design of Optimal Online Control Algorithm

Based on the analysis presented in previous subsections, we propose an optimal online control algorithm, via minimizing the upper bound of the drift-minus-utility in each time slot.

The basic idea of our algorithm is to maximize terms (18) (19) and (20) in the right side of the inequality, respectively, since there are no coupled variables among the three terms. The values of $\alpha_i(t), A_i(t), A_{ij}(t)$ and $s_{ij}(t)$ are calculated based on the knowledge of $H(t)$ and $Q(t)$. Moreover, $H(t)$ and $Q(t)$ are updated according to the calculation results in each time slot. In summary, our optimal online control algorithm proceeds in four steps:

- **Step 1:** Computing auxiliary variables $\alpha_i(t)$ by maximizing (18);
- **Step 2:** Making decisions on request admission and task allocation by computing $A_i(t)$ and $A_{ij}(t)$ in (19);
- **Step 3:** Scheduling the tasks according to the values of $s_{ij}(t), \forall i \in M, \forall j \in N$ decided by maximizing (20);
Step 4: Updating the queue backlogs $H(t)$ and $Q(t)$.

The four steps are operated consecutively in each time slot, and then executed repeatedly over time slots. The details of each step are explained in the following.

1) Auxiliary Variable Computation: As the terms associated with variable $\alpha_i(t)$ in (18) are independent, the maximization of (18) can be decoupled into several subproblems as follows, and computed concurrently.

$$\max \quad V \mu \alpha_i(t) - H_i(t) \alpha_i(t) \quad \text{(21)}$$

s.t. \quad 0 \leq \alpha_i(t) \leq R^\text{max}_i, \forall i \in \mathcal{M},

Given the values of $V$, $\mu$, $H_i(t)$ and $R^\text{max}_i$, the problem (21) is a linear programming. Considering the constraints, the optimal results of (21) are

$$\alpha_i(t) = \begin{cases} 0, & H_i(t) > \mu V, \quad \forall i \in \mathcal{M}. \\ \hat{R}^\text{max}_i, & \text{else.} \end{cases} \quad \text{(22)}$$

**Insight:** The optimal value of $\alpha_i(t)$ is directly decided by the comparison between $H_i(t)$ and $\mu V$. If $H_i(t)$ is small, it implies the value of $\alpha_i(t)$ is close to $A_i(t)$, which shows the system is stable currently. Thus, a large value of $\alpha_i(t)$ should be chosen to improve the request processing throughput. Otherwise, a large value of $H_i(t)$ means the corresponding virtual queue suffers congestion, and a small value of $\alpha_i(t)$ should be chosen.

2) Request Admission and Task Allocation: Similar to (18), the maximization of (19) can be decoupled into several independent subproblems associated with $A_i(t)$ and $(A_{i1}(t), A_{i2}(t), \ldots, A_{in}(t))$ for each $i \in \mathcal{M}$. One of the subproblems is formulated as follows,

$$\max_{A_i(t), A_{ij}(t)} H_i(t)A_i(t) - \sum_{j \in \mathcal{N}} A_{ij}(t)Q_{ij}(t) \quad \text{(23)}$$

s.t. \quad 0 \leq A_i(t) \leq R_i(t), \forall i \in \mathcal{M},

$$A_i(t) = \sum_{j \in \mathcal{N}} A_{ij}(t).$$

This problem is complex as there are several coupled variables. For this reason, we first simplify the problem by assuming $A_i(t)$ is a fixed and known parameter. Then, the problem can be rewritten as the following expression,

$$\min_{A_{ij}(t)} \sum_{j \in \mathcal{N}} A_{ij}(t)Q_{ij}(t) \quad \text{(24)}$$

s.t. \quad $\sum_{j \in \mathcal{N}} A_{ij}(t) = A_i(t), \forall i \in \mathcal{M}.$

The problem in (24) characterizes how to allocate tasks to smartphones given the number of admitted requests. The solution to problem (24) is formulated in the following. We can find the strategy of task allocation is to allocate all admitted requests to the smartphone with the shortest queue. Intuitively, this strategy promotes task balancing among smartphones and shorter request processing delays.

$$A_{ij}(t) = \begin{cases} A_i(t), & \text{if } j = \arg \min_{j \in \mathcal{N}}(Q_{ij}(t)), \forall i \in \mathcal{M}. \\ 0, & \text{otherwise.} \end{cases} \quad \text{(25)}$$

Based on the above allocation strategy, we next decide the optimal value of $A_i(t)$ in (23) which can be rewritten as

$$\max_{A_i(t)} H_i(t)A_i(t) - A_i(t)Q_{ij}(t) \quad \text{(26)}$$

s.t. \quad $0 \leq A_i(t) \leq R_i(t), \forall i \in \mathcal{M},$

where $j^*_i = \arg \min_{j \in \mathcal{N}}(Q_{ij}(t)), \forall i \in \mathcal{M}$. The problem in (26) is a linear programming problem with respect to $A_i(t)$. The solution is

$$A_i(t) = \begin{cases} R_i(t), & H_i(t) \geq Q_{ij^*_i}(t), \forall i \in \mathcal{M}. \\ 0, & \text{else.} \end{cases} \quad \text{(27)}$$

**Insight:** From (27), we can see the strategy for request admission control is a threshold-based strategy. When the shortest queue backlog of type $i$ is smaller than the backlog of the corresponding virtual queue, all arrived requests in the current time slot are admitted. Otherwise, all arrived requests are denied. The idea of this strategy is to improve the throughput $A_i(t)$ when system congestion is not serious, and to ensure the stability at the same time.

3) Task Scheduling: Scheduling the tasks to be processed on each smartphone in time slot $t$ can be performed by solving the problem of maximizing (20). As $s_{ij}(t) \in \{0, 1\}, \forall i \in \mathcal{M}, \forall j \in \mathcal{N}$, the problem is a non-linear 0-1 integer programming, and hence NP-hard. The optimal solution to the problem can be obtained by the branch-and-bound algorithm [13].

4) Queue Updating: Finally, the queue backlogs of virtual and actual queues are updated. $Q_{ij}(t)$ can be updated based on (2) with the values of $A_{ij}(t)$ and $s_{ij}(t)$, and $H_i(t)$ can be updated based on (12) with the values of $\alpha_i(t)$ and $A_i(t)$.

E. Distributed Implementation

Previously we have proposed the optimal online control algorithm. This is a centralized algorithm, since it should be executed by the central platform and gather all necessary information about smartphones. However, the number of smartphones can be huge. This suggests that the computation load for the central platform is high. The central point must also maintain all state variables about each smartphone, e.g., task queues on each smartphone. In addition, the experience degradation function is private information. Some smartphones may not be willing to release such private information. Thus, in this subsection we propose a distributed implementation of the previously proposed optimal online control algorithm, which requires no private information about the experience degradation function.

The main idea of the distributed implementation is to divide all computations of the online algorithm into two parts. The first part is to be performed by the platform, which includes 1) Auxiliary Variable Computation, and 2) Request Admission and Task Allocation. The second part is to be performed jointly by the smartphones, which includes 3) Task Scheduling.

The key question of this distributed implementation is: why can the third step of task scheduling be jointly performed by each smartphone. The main reason is that the problem of maximizing (20) can be decomposed into $|\mathcal{N}|$ subproblems.
with respect to \( j \), and each subproblem can be optimally solved by the corresponding smartphone. One subproblem is as follows,

\[
\max_{s_{ij}(t)} \sum_{i \in \mathcal{M}} Q_{ij}(t)s_{ij}(t) - V \left( \sum_{i \in \mathcal{M}} \nu_i s_{ij}(t) \right)^2
\]

\[s_{ij}(t) \in \{0, 1\}, \forall i \in \mathcal{M}.
\]

We can see that each subproblem contains a set of decision variables that are independent of other subproblems. In addition, the set of input variables to each subproblem, including \( Q_{ij}(t) \), \( \nu_i \), \( \beta_j \) and \( V \) are known to smartphone \( j \). Consequently, each smartphone solves its own subproblem, obtaining the scheduling result for the tasks on this smartphone itself. As a result, we can easily conclude that the scheduling results of all smartphones together are equivalent to the results computed by the centralized control algorithm.

To allow the platform to proceed correctly, each smartphone \( j \) should send its scheduling result \( s_{ij}(t) \), \( i \in \mathcal{M} \) back to the platform. After receiving all scheduling results from the smartphones, the platform updates the queue backlogs \( Q_{ij}(t) \) on its own side. As for the fourth step of Updating Queue in the centralized control algorithm, virtual queues \( H_{ij}(t) \), \( i \in \mathcal{M} \) are updated by the platform and actual queues \( Q_{ij}(t) \), \( i \in \mathcal{M} \) on each smartphone \( j \) are updated by the smartphone itself.

**Insights:** From the distributed implementation of the algorithm, we can see that neither the platform nor other smartphones require the experience degradation function of a smartphone, which is only accessed by the smartphone itself.

**E. Optimality Analysis**

We next analyze the optimality in terms of time-averaged performance and stability of the system under our proposed online control algorithm, as shown in Theorem 1.

**Theorem 1.** For arbitrary request arrivals in each time slot \((R_1(t), R_2(t), \cdots, R_m(t))\) and any tradeoff parameter \( V \), the time-averaged performance achieved by our algorithm is approximately optimal. The gap compared with the optimum is within \( \frac{B}{V} \), as shown in the following inequality,

\[
\lim_{t \to \infty} \left\{ \mu \sum_{i \in \mathcal{M}} \bar{\alpha} - \sum_{j \in \mathcal{N}} \bar{d}_j \right\} \geq \varphi^* - \frac{B}{V},
\]

where \( \varphi^* = \mu \sum_{i \in \mathcal{M}} \bar{\alpha}^* - \sum_{j \in \mathcal{N}} \bar{d}_j^* \) are the optimal solutions of problem (5), derived with complete information over all time slots. \( B \) is a constant given in Lemma 2. Moreover, the system stability is guaranteed as the following inequalities hold,

\[
H_{ij}(t) \leq \mu V + R_{ij}^{\max}, \forall i \in \mathcal{M},
\]

\[
Q_{ij}(t) \leq \mu V + 2R_{ij}^{\max}, \forall i \in \mathcal{M}, \forall j \in \mathcal{N}.
\]

**Insight:** Theorem 1 shows that the time-averaged system-wide performance deviates by at most \( O(1/V) \) from the optimum, with virtual and actual queues are bounded within \( O(V) \) over time. This result also illustrates \( V \) is a tradeoff between time-averaged performance and system stability. A large value of \( V \) implies that the system emphasize the performance maximization according to (16). The time-averaged utility is arbitrarily close to the optimum if \( V \) tends to infinity according to (29). However, the large queue backlogs imply the delays for requests waiting in queues are tremendously long according to (30) and (31).

**Proof:** First, we prove the strong stability of the virtual queues \( H_{ij}(t) \) and actual queues \( Q_{ij}(t) \) under our optimal online algorithm.

For \( H_{ij}(t) \), suppose \( H_{ij}(t) \leq \mu V + R_{ij}^{\max} \) holds for \( \forall i \in \mathcal{M} \). At time \( t = 0 \), all queues are initialized to empty. For time \( t + 1 \), we consider the following three cases:

1) if \( H_{ij}(t) > \mu V \): based on (22), we have \( \alpha_i(t) = 0 \).

According to (12), we have \( H_{ij}(t+1) \leq H_{ij}(t) + \alpha_i(t) \leq \mu V + R_{ij}^{\max} \), which implies that (30) holds in this case.

2) if \( H_{ij}(t) \leq \mu V \): \( \alpha_i(t) = R_{ij}^{\max} \). We have \( H_{ij}(t) \leq H_{ij}(t) + \alpha_i(t) \leq \mu V + R_{ij}^{\max} \), which also satisfies (30).

Above all, \( H_{ij}(t) \leq \mu V + R_{ij}^{\max} \) holds for all time slots. Similarly, the strong stability of actual queues \( Q_{ij}(t) \) can be proved, based on the control (25) and (27) in our algorithm. We leave the proof to readers for brevity.

Next, we prove the optimality of the time averaged performance in Theorem 1, which requires the following lemma:

**Lemma 3.** For any arrival rates \((\tau_1, \tau_2, \cdots, \tau_M)\), there exists a randomized stationary control policy \( \phi \) that chooses feasible controls \( \alpha_i^\delta(t), A_i^\delta(t), s_i^\delta(t) \) and \( s_j^\delta(t) \), \( \forall i \in \mathcal{M}, \forall j \in \mathcal{N}, \forall t \), which have the following steady values,

\[
\alpha_i^\delta(t) = \pi_i^* \quad (32)
\]

\[
E\{s_i^\delta(t)\} \geq E\{R_i^\delta(t)\}
\]

\[
E\{A_i^\delta(t)\} = \pi_i^* \quad E\{D_j^\delta\} = \bar{d}_j^*.
\]

As Lemma 3 can be proved with similar techniques in [10], we omit the details for brevity. The proof of (29) under our algorithm is given as follows.

Let \( \varphi(t) = \sum_{i \in \mathcal{M}} \mu \alpha_i(t) - \sum_{j \in \mathcal{N}} D_j(t) \) denote the utility of the participatory sensing system in time slot \( t \). Our algorithm minimizes the left side of (18) in each time slot, among all feasible decisions. We have

\[
\Delta(\Theta(t)) = V\varphi(t)|\Theta(t)|
\]

\[
\leq B - \sum_{i \in \mathcal{M}} E\{V \mu \alpha_i^\delta(t) - H_{ij}(t) \alpha_i^\delta(t)|\Theta(t)\}
\]

\[
- \sum_{i \in \mathcal{M}} E\{H_{ij}(t) A_i^\delta(t) |\Theta(t)\} + \sum_{i \in \mathcal{M}, j \in \mathcal{N}} E\{Q_{ij}(t) A_i^\delta(t) |\Theta(t)\}
\]

\[
- \sum_{i \in \mathcal{M}, j \in \mathcal{N}} E\{s_i^\delta(t) |\Theta(t)\} + V \sum_{j \in \mathcal{N}} E\{D_j^\delta(t)|\Theta(t)\}.
\]

Since the optimal controls \( \alpha_i^\delta(t), s_i^\delta(t), A_i^\delta(t), A_i^\delta(t) \) are independent of \( \Theta(t) \), we can delete the condition \( \Theta(t) \) in both.
sides.

\[
\begin{align*}
&\mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t))\} - \mathbb{V}\{\varphi(t)\} \\
\leq & \sum_{i \in \mathcal{M}} \mathbb{E}\{V_i \alpha_i(t) - H_i(t) \alpha_i^2(t)\} \\
- & \sum_{i \in \mathcal{M}} \mathbb{E}\{H_i(t) \alpha_i^2(t)\} \sum_{j \in \mathcal{N}} \mathbb{E}\{\alpha_j \alpha_j(t)\} \\
- & \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{N}} Q_{ij}(t) \mathbb{E}\{s_{ij}(t)\} + \mathbb{V} \sum_{j \in \mathcal{N}} \mathbb{E}\{D_j(t)\}.
\end{align*}
\]

Then, plugging (32) into the right side, we have

\[
\begin{align*}
&\mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t))\} - \mathbb{V}\{\varphi(t)\} \leq B - V \varphi^*.
\end{align*}
\]

By summing the both sides over time \( \tau \in \{0, 1, \cdots, t-1\} \)

\[
\frac{\mathbb{E}\{L(\Theta(t)) - L(\Theta(0))\}}{t} - \frac{V \sum_{\tau=0}^{t-1} \mathbb{E}\{\varphi(\tau)\}}{t} \leq B - V \varphi^*.
\]

Considering the fact that \( L(\Theta(t)) \geq 0 \) and \( L(\Theta(0)) = 0 \), we have

\[
\frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\varphi(\tau)\} \geq \varphi^* - \frac{B}{V}.
\]

By letting \( t \to \infty \), we have

\[
\lim_{t \to \infty} \left\{ \mu \sum_{i \in \mathcal{M}} \overline{\alpha_i} - \sum_{j \in \mathcal{N}} \overline{\alpha_j} \right\} \geq \varphi^* - \frac{B}{V}.
\]

As virtual queue is proved to be stable, we have \( \overline{\alpha_i} \geq \overline{\alpha_i} \).

Thus, (29) is satisfied according to (34).

\section{IV. Performance Evaluation}

\subsection{A. Methodology and Simulation Setup}

We perform simulations to evaluate our online control algorithm. We compare our proposed algorithm with two baseline algorithms described as follows.

- **Online control algorithm without queuing (CAWOQ):**
  In this algorithm, there are no queues on smartphones. When a request arrives at the system, the platform finds an unoccupied smartphone who can obtain the largest performance gain for processing the task. Only when the obtained performance gain is positive, this request is admitted to the system for processing, and allocated to the corresponding smartphone.

- **Online control algorithm with fixed-length queues (CAFLQ):**
  In this algorithm, each smartphone maintains a queue with a fixed length for each type of tasks. The length of the queues is equal to the upper bound of the actual queues as computed in Theorem 1. When a request arrives at the system, it will be denied only if all queues of its type on all smartphones have already been full. If a request is admitted, it is allocated to the smartphone with the shortest queue. Finally, the algorithm determines the set of tasks to be processed by greedily selecting the tasks until the performance gain can no longer be increased.

In addition to system-wide performance, we also examine fairness among smartphones when evaluating the performance of our algorithm. Fairness is also an important requirement of a participatory sensing system. In general, it is highly desirable that the smartphones fairly share the load of processing sensing tasks. The fairness is computed as the variance of the time-averaged number of tasks processed by each smartphone. It is clear that a smaller value of the variance indicates better fairness among smartphones.

The default setting is as follows. There are 200 smartphones and the smartphone-specific positive coefficient \( \beta_j \) is set to one for all smartphones. There are 6 types of requests. The costs \( \nu_i \) of different types of sensing tasks are \([1, 2, 3, 4, 5, 6]\). The average arrival rate \( \overline{\tau_i} \) of each type of requests is set to \([40, 60, 40, 60, 80]\). The maximum number of arriving requests in each time slot \( R_{\text{max}}^i = 2\overline{\tau_i} \), and \( R_i(t) \) is uniformly distributed in \([0, R_{\text{max}}^i]\), \( \forall i \in \{1, \cdots, 6\} \). We choose an empirical value of \( \mu = 10 \). All simulations are performed over 10,000 time slots. All measurements are an average over ten independent runs.

\subsection{B. Case Study}

We first study two cases of our optimal online algorithm in response to varying request arrival rates. The change of the request arrival rates over time is shown in Fig. 3. The arrival rate of each type has three phases. Take Case 1 for example. In the first phase, each rate is equal to its default rate (i.e., \( \overline{\tau_i} \)). In the second phase, each rate abruptly rises to \( 2\overline{\tau_i} \). In the last phase, each rate drops suddenly to \( 0.5\overline{\tau_i} \).

Fig. 4 plots the instant system request processing throughput over time. We can see our online algorithm can quickly adapt to even dynamically varying request arrivals. Moreover, our online algorithm has stable performance in each phase of the varying arrival rates. Fig. 5 reports the average backlog of all actual queues on the smartphones in each time slot. We can find that all queues are strongly stable despite abruptly changing arrival rates.

\subsection{C. Impact of Arrival Rates}

We next study the performance of the three algorithms under different arrival rates. To vary the arrival rates of all types of tasks, we change the ratio of the arrival rate of each type to the default setting \( \tau_i \) as described in subsection IV-A from 0.25 to 1.5. The results are shown in Fig. 6 and Fig. 10.

Fig. 6 shows that our online algorithm is better than both CAWOQ and CAFLQ under all arrival rates. When the ratio is 1.5, the time-averaged system-wide performance of our online algorithm is 22.3% and 198.9% higher than CAFLQ and CAWOQ, respectively. We can also find that the increasing rates of performance of all the three algorithms become smaller as the arrival rates increase. This is because the performance gain contributed by increased request processing throughput is further limited by the total capacity of smartphones.

In Fig. 10, we can find that our online algorithm achieves better fairness than the other two algorithms. CAFLQ has the worst fairness. This demonstrates our online control algorithm allocates tasks to smartphones in a better balanced way.
D. Impact of Number of Smartphones

We compare the performance and fairness achieved by the three algorithms when the number of smartphones varies from 100 to 600. The results are reported in Fig. 7 and Fig. 11.

From Fig. 7, we can find that the time-averaged performance of both our algorithm and CAFLQ increases dimensionally as the number of smartphones increases. This is because more smartphones can obtain more performance gain by processing more tasks. When there are 600 smartphones, the time-averaged system-wide performance of our algorithm is 30.4% and 393.1% higher than CAFLQ and CAWOQ, respectively.

Our online algorithm can consistently achieve good fairness as the number of smartphones increases, as shown in Fig. 11. The variance of time-averaged throughput among all smartphones of our algorithm is always smaller than 0.05, while the highest variance of CAFLQ is higher than 0.5.

E. Impact of Number of Types

Next, we study the impact of the number of types. We have conducted simulations with three different numbers of types, i.e., {2, 4, 6}. The corresponding costs $\nu_i$ of different types of tasks are set to {3, 4}, {2, 3, 4, 5} and {1, 2, 3, 4, 5, 6}, respectively.

Fig. 8 and Fig. 12 show that our online algorithm outperforms CAFLQ and CAWOQ in terms of both system-wide performance and fairness. When there are six types, the performance of our online algorithm is 26.0% and 280% higher than CAFLQ and CAWOQ, respectively, and the variance is 5.7% and 9.1% of those produced by CAFLQ and CAWOQ, respectively.

F. Impact of $V$

Finally, we perform simulations to validate Theorem 1. Parameter $V$ trades off the system-wide performance and queue stability. Fig. 9 plots the time-averaged performance under different values of $V$. We can find that the performance increases when $V$ enlarges, and trends to be stable. This is because the gap compared with the optimum is $O(1/V)$ according to Theorem 1. Fig. 13 plots the time-averaged congestion captured by the Lyapunov function (14) under different $V$. As $V$ enlarges, the queue congestion of our online algorithm increases, which confirms Theorem 1.

V. RELATED WORK

Participatory sensing has recently received increasing research efforts. In this section we review related work in the following aspects, and compare our work with existing studies.

Applications and systems: A number of participatory sensing systems [14][15] have been constructed and many participatory sensing applications [4][3][16][17] have been developed. For example, a system called $S^2 a a S$ [15] is developed to provide sensing services for cloud users, and the application PIER [4] can be used to collect the sensing reports of personal environment impact via smartphones. These studies focusing on systems and applications usually assume that there are enough volunteer smartphones in the system for processing sensing tasks, and do not consider the problem of possible shortage of smartphones and the necessity of admission control.

Incentive mechanisms: There is a body of studies [6][7][8][9] dedicated on designing incentive mechanisms to
encourage smartphones, where participants selfishly try to maximize their own profits. In [6], the economic model for user participation is studied, and a reverse auction based on dynamic prices is designed. Considering location information of participants and a fixed cost budget, an incentive mechanism based on a greedy algorithm is proposed in [7]. Koutsopoulos [8] points out that the cost of a participant is private information and may be misrepresented. An monetary incentive mechanism is designed to minimize the total prices paid, while ensuring participants are truthful. All the above research efforts typically assume the sensing tasks to be processed are fixed and given, ignoring the serious problem of imbalanced demand and supply. The task allocation in their incentive mechanisms is performed in an offline manner. In comparison, our work specially focuses on the crucial problem of online request admission control, task allocation, and task scheduling on smartphones, all of which are performed on the fly. Note that our work studies participatory sensing systems with cooperative smartphones other than strategic smartphones.

VI. CONCLUSION

This paper has focused on the problem of maximizing the system-wide performance of a participatory sensing system. We have proposed the three-layer control framework which consists of (1) request admission control, (2) task allocation, and (3) task scheduling on smartphones. Based on the framework, we have proposed an optimal distributed online control approach. This approach has produced three effective control strategies for controlling a participatory sensing system: (1) We use a threshold-based strategy in admission control, trying to increase system throughput and avoid congestion; (2) all admitted requests of the same type are allocated to the smartphone with the shortest queue, balancing the loads on different smartphones; and (3) task scheduling is jointly performed by each smartphone. Comprehensive simulation results have confirmed the superiority of our online control approach.