# Efficient and Low-Overhead Uplink Scheduling for Large-Scale Wireless Internet-of-Things

Bin Li[1], Bo Ji[2], and Jia Liu[3]

[1]Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island
[2]Department of Computer and Information Sciences, Temple University
[3]Department of Computer Science, Iowa State University

***Abstract*—With the rapid growth of Internet of Things (IoT) applications in recent years, there is a strong need for wireless uplink scheduling algorithms that determine when and which subset of a large number of users should transmit to the central controller. Different from the downlink case, the central controller in the uplink scenario typically has very limited information about the users. On the other hand, collecting all such information from a large number of users typically incurs a prohibitively high communication overhead. This motivates us to investigate the development of an efficient and low-overhead uplink scheduling algorithm that is suitable for large-scale IoT applications with limited amount of coordination from the central controller. Specifically, we first characterize a capacity outer bound subject to the sampling constraint where only a small subset of users are allowed to use control channels for system state reporting and wireless channel probing. Next, we relax the sampling constraint and propose a joint sampling and transmission algorithm, which utilizes full knowledge of channel state distributions and instantaneous queue lengths to achieve the capacity outer bound. The insights obtained from this capacity-achieving algorithm allow us to develop an efficient and low-overhead scheduling algorithm that can strictly satisfy the sampling constraint with *asymptotically diminishing* throughput loss. Moreover, the throughput performance of our proposed algorithm is independent of the number of users, a highly desirable property in large-scale IoT systems. Finally, we perform extensive simulations to validate our theoretical results.**

## I. Introduction

Internet of Things (IoT) refers to the internetworking of a large number of heterogenous smart devices (e.g., smart phones, tablets, sensors and actuators) to meet the demands of the ever-increasing applications in personalized health care, smart home, ubiquitous environmental monitoring, smart manufacturing, etc. It is estimated that the global market for IoT will reach 20.4 billion devices by 2020 [1]. However, unlike traditional data communication networks where the predominant amount of data is transmitted in the downlink, a distinct feature of IoT applications is that a significant portion of the IoT data traffic is carried in the *uplink* (i.e., from user devices to the central controller). In most IoT applications, each device generates sparse or intermittent data traffic and transmits them to a central controller or access point (AP) for data processing, typically through wireless connections (referred to as wireless IoT uplink systems in this paper). As a result, to support the enormous amount of devices given limited spectral resources, there is a compelling need for efficient and low-overhead wireless IoT uplink scheduling algorithms that determine when and which devices (referred to as users in the rest of the paper) are allowed to transmit, with the goal of supporting as many users as possible, or equivalently, *throughput-optimal scheduling*.

Over the years, throughput-optimal scheduling has been extensively studied in the networking research community and many results are available. However, due to its unique features and applications, IoT uplink scheduling design is far more challenging compared to its counterparts in traditional wireless networks (see Section II for a detailed discussion). In particular, due to the sheer size of IoT systems, a well-designed uplink scheduling policy not only has to be near throughput-optimal, it also needs to be low-overhead. To this end, in this paper, we propose an efficient and simple design based on the key idea termed "pick and compare (PC)" (e.g., [22], [13], [4], [20]). Simply speaking, a PC scheme always stores the most congested user and compares its queue-length with a *randomly* selected user. It has been shown that the class of PC algorithms achieves the maximum throughput through gradually improved quality of transmission decisions over time. The simplicity, low-complexity, and scalability of PC algorithms motivate us to consider their deployments in large-scale IoT uplink systems, where extensive coordination from the AP is typically infeasible.

However, due to several technical challenges, developing a PC-based scheduling scheme for IoT uplink systems and conducting rigorous performance analysis is highly non-trivial. Traditionally, PC algorithms were developed for systems where the queue-length evolution processes are smooth. Unfortunately, in the presence of wireless channel fading, the time-varying channel rates could change rather abruptly. Consequently, most of the proof techniques used for establishing the throughput-optimality of PC-based algorithms fail under wireless channel fading settings. In fact, it remains an *open question* whether it is possible to design an efficient and low-overhead PC-based scheduling algorithm for IoT uplink systems with wireless channel fading. A key contribution of this paper is that we show that the answer is "*yes*." Our main results in this paper are summarized as follows:

This work is supported in part by NSF grants: CNS-1717108, ECCS-1818791, CCF-1758736, CNS-1758757, CNS-1446582, CNS-1651947 and CCF-1657162; and ONR grant: N00014-17-1-2417.

- First, we characterize a capacity outer bound for a wireless IoT uplink system subject to a sampling constraint, i.e., we limit the number of users that can use control channels for reporting system state information and probing uplink channels. This capacity outer bound lays the foundation for the performance analysis of our proposed algorithms.
- Next, we consider an efficient and low-overhead uplink scheduling design. In particular, we relax the sampling constraint and propose a two-stage MaxWeight-type scheduling algorithm that utilizes the full knowledge of channel state distributions and instantaneous queue lengths to achieve the capacity outer bound.
- Finally, building upon this capacity-achieving algorithm, we develop an iterative pick-and-compare (IPC) algorithm that strictly *satisfies* the sampling constraint with a *vanishing* throughput loss. More precisely, let $\Lambda(M,K)$ denote the capacity outer bound for an $N$-user $M$-channel uplink system with $K$-user sampling constraint. We show that our IPC algorithm can stabilize any arrival rates within the region $\Lambda(M,K-1)$. Further, the gap between $\Lambda(M,K)$ and $\Lambda(M,K-1)$ is vanishing as $K \uparrow N$, and thus proving the asymptotic throughput-optimality of our IPC algorithm. Moreover, the achievable rate region of our IPC algorithm is independent of the number of users, a highly desirable property in large-scale IoT systems.

To our knowledge, our work is the first to offer asymptotic throughput-optimality for IoT uplink scheduling. The remainder of this paper is organized as follows. Section II presents related work, putting this paper into perspective. Section III introduces the network model and problem formulation. Section IV provides a capacity outer bound characterization. Section V covers the key elements of our low-overhead uplink scheduling design. Section VI presents numerical results, and Section VII concludes this paper.

## II. Related Work

In this section, we provide a quick overview of throughput-optimal scheduling to put our work into perspective. In the literature, a well-known class of throughput-optimal scheduling algorithms is the family of MaxWeight policies, which date back to the seminal work by Tassiulas and Ephremides (e.g., [23], [24]) and consist of many follow-ups (e.g., [5], [14], [21], [15], [6]). The basic idea underpinning MaxWeight policies is to schedule users who have both good channel qualities and high congestion levels (evaluated by queue lengths or delays, etc.). However, it is exactly the requirement of both congestion and channel state information that renders MaxWeight policies *unsuitable* for IoT uplink scheduling. In many IoT applications, due to the large number of devices, congestion and channel state information is prohibitively expensive to collect. Exacerbating this problem is the fact that even those reduced channel-probing-overhead MaxWeight variants designed for large-size downlink systems (e.g., for energy minimization [11] or limited channel quality feedback [17]) cannot be directly applied in IoT uplink systems. The reason is that these MaxWeight variants still require queue-length information of all users, while the AP in IoT uplink systems usually has limited queueing information about their associated users. As a result, the assumption of accurate congestion level information for all users, which is usually valid for downlink scheduling, no longer holds in the uplink counterpart.

To overcome the limitations of MaxWeight policies, one solution is to adopt the carrier sense multiple access (CSMA) design (e.g., [8], [18], [16]), where each user judiciously selects its CSMA parameters to adapt to its congestion levels (e.g., the queue-lengths). However, most existing CSMA-based schemes assume that the weight of each user (typically some function of the queue-length) changes slowly over time, which makes their extensions to settings with wireless channel fading quite difficult. Indeed, in the presence of fading, the weight of each user (determined by the product of the queue-length and the current feasible service rate) is stochastic and can change rather abruptly. Although this restriction on slow time-varying weight has been relaxed in recent works (e.g., [26], [9]), a more fatal limitation of CSMA-based policies is that the average time required for successfully acquiring the channel in the channel contention grows *exponentially* with the number of users. This limitation renders CSMA-type schemes impractical for IoT applications that typically involve a huge number of users. As will be seen shortly, our proposed PC-based scheme avoids both the slow time-varying restriction and the scalability pitfall of CSMA.

## III. System Model

We consider an IoT uplink system with one access point (AP) and $N$ users, where each user transmits data to the AP through $M$ orthogonal channels. We assume that the system operates in slotted time with normalized slots $t \in \{1,2,\ldots\}$. We use $\mathbf{C}[t] = (C_{i,j}[t], i = 1,2,\ldots,N, j = 1,2,\ldots,M)$ to capture the wireless channel fading, where $C_{i,j}[t]$ denotes the maximum amount of service available for user $i$ to transmit packets on channel $j$ in slot $t$. Due to the finite number of modulation and coding schemes, we assume that each channel for each user has $R$ possible channel rates $c_1, c_2, ..., c_R$ with $0 = c_1 < c_2 < ... < c_R = c_{\max}$, which measure the maximum number of packets that can be delivered in one time slot. We assume that $\mathbf{C}[t]$ are independently distributed across both users and channels, and independently and identically distributed (i.i.d.) over time.

In the IoT uplink system, each user needs to send a control message to the AP in order for the AP to obtain its state information (e.g., congestion levels and channel rates). In particular, the AP fetches the congestion level of a user by decoding its control message, and obtains its channel rate by measuring the signal-to-noise ratio of the received control message. However, due to the restrictive amount of uplink resources, the AP usually has very limited information about the state of each user in a large-scale IoT uplink system. Therefore, we assume that the AP can at most sample $K$ ($K \ll N$) users to acquire their channel states and congestion levels on each channel at the beginning of each time slot.

We denote the sampling schedule as $\mathbf{X}[t] = (X_{i,j}[t], i = 1, 2, \ldots, N, j = 1, 2, \ldots, M)$, where $X_{i,j}[t] = 1$ if user $i$ is sampled to send a control message on channel $j$ such that the AP acquires its congestion level and the state of channel $j$ in time slot $t$, and $X_{i,j}[t] = 0$ otherwise. Let $\mathcal{X}(K)$ be the collection of all sampling schedules where $K$ users are sampled to send their control messages on each channel. To avoid interference, at most one user is allowed to transmit over each channel in each time slot. We call a schedule where at most one user is active on each channel in each time slot as a *feasible schedule* and denote it as $\mathbf{S}[t] = (S_{i,j}[t], i = 1, 2, \ldots, N, j = 1, 2, \ldots, M)$, where $S_{i,j}[t] = 1$ if user $i$ is scheduled on channel $j$ in time slot $t$ and $S_{i,j}[t] = 0$ otherwise. We use $\mathcal{S}$ to denote the collection of all feasible schedules. Here, we assume that only the sampled users are allowed to transmit in each time slot.

We assume that each user $i$ serves its own data traffic and maintains packets in a queue with $Q_i[t]$ denoting its queue length at the beginning of time slot $t$, which reflects its congestion level. The larger the $Q_i[t]$, the more congested the user $i$. Let $A_i[t]$ denote the number of packets arriving at user $i$ in time slot $t$, which is independently distributed across users and i.i.d. over time with mean $\lambda_i > 0$. We assume $A_i[t] < A_{\max}, \forall t \geq 0$, for some positive constant $A_{\max} < \infty$. This is a reasonable assumption since, as mentioned earlier, most IoT data traffic are low-rate and intermittent. Then, the evolution of user $i$'s queue can be described as follows:

$$Q_i[t+1] = \left( Q_i[t] + A_i[t] - \sum_{j=1}^{M} X_{i,j}[t] S_{i,j}[t] C_{i,j}[t] \right)^+$$

for $i = 1, 2, \ldots, N$, where $(x)^+ \triangleq \max\{x, 0\}$. In this paper, we are interested in developing an efficient uplink scheduling algorithm with limited coordination from the AP. In particular, our goal is to find a joint sampling and transmission schedule $\{\mathbf{X}[t], \mathbf{S}[t]\}_{t \geq 1}$ such that, in each time slot and for each channel, (i) $K$ users are allowed to send their control messages in order for the AP to acquire their queue-lengths and channel rates; and (ii) at most one user can be scheduled among these $K$ sampled users. A key difficulty of this joint sampling and scheduling problem is that the information available for making transmission scheduling decision $\mathbf{S}[t]$ heavily relies on the sampling decision $\mathbf{X}[t]$.

We consider the class $\mathcal{P}$ of stationary sampling and transmission policies that first decide the sampling schedule $\mathbf{X}[t]$ at slot $t$ based on the available information $\mathbf{Q}[t] = (Q_1[t], \ldots, Q_N[t])$ and channel state distributions, and then determine the transmission schedule $\mathbf{S}[t]$ based on ($\mathbf{Q}[t] \otimes \mathbf{X}[t]$, $\mathbf{C}[t] \otimes \mathbf{X}[t]$), where $\otimes$ stands for component-wise multiplication. In other words, a joint sampling and transmission policy in $\mathcal{P}$ is a two-stage mapping where it first maps from the space of $\mathbf{Q}[t]$ to the space of sampling schedules $\mathcal{X}(K)$ during the sampling stage and then maps from the space of ($\mathbf{Q}[t] \otimes \mathbf{X}[t]$, $\mathbf{C}[t] \otimes \mathbf{X}[t]$) to the space of feasible schedules $\mathcal{S}$ during the transmission stage. Under any policy in $\mathcal{P}$, the queue length process $\{\mathbf{Q}[t]\}_{t \geq 1}$ forms a Markov Chain.

We say that queue $i$ is *strongly stable* if $\limsup_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[Q_i[t]] < \infty$. The system is *stable* if all queues in the system are strongly stable. We let $\Theta(M, K)$ denote the *capacity region* for an IoT uplink system with $M$ orthogonal channels and $K$ allowed users in sampling for each channel in each time slot, which represents the maximum set of arrival rate vectors $\boldsymbol{\lambda} = (\lambda_i)_{i=1}^N$ for which the system is stable under some policy. We call an algorithm *optimal* if it keeps the system stable for any arrival rate vector that lies strictly inside $\Theta(M, K)$. Note that it is in general difficult to directly characterize $\Theta(M, K)$. To that end, we first derive an outer bound for the capacity region in the next section.

## IV. Capacity Outer Bound Characterization

In this section, we characterize an outer bound of the capacity region $\Theta(M, K)$ for a wireless IoT uplink system with $M$ orthogonal channels under the sampling constraint that $K$ users are allowed to send their control messages on each channel in order for the AP to acquire their channel rates and queue-lengths and the scheduling constraint that at most one user is allowed to transmit on each channel in each time slot.

*Proposition 1 (Capacity Outer Bound):* $\Theta(M, K)$ is contained in the rate region $\Lambda(M, K)$ (i.e., $\Theta(M, K) \subseteq \Lambda(M, K)$), where $\Lambda(M, K)$ is defined as the set of arrival rate vectors $\boldsymbol{\lambda} = (\lambda_i)_{i=1}^N$ for which there exist non-negative numbers $\alpha(\mathbf{x})$ and $\beta(\mathbf{x}, \mathbf{c}; \mathbf{s})$ such that the following expressions are satisfied:

$$\lambda_i \leq \sum_{\mathbf{x} \in \mathcal{X}(K)} \alpha(\mathbf{x}) \sum_{\mathbf{c}} p(\mathbf{c}) \sum_{\mathbf{s} \in \mathcal{S}} \beta(\mathbf{x}, \mathbf{c}; \mathbf{s}) \sum_{j=1}^{M} x_{i,j} c_{i,j} s_{i,j}, \forall i, \quad (1)$$

$$\sum_{\mathbf{s} \in \mathcal{S}} \beta(\mathbf{x}, \mathbf{c}; \mathbf{s}) = 1, \forall \mathbf{x}, \mathbf{c}, \text{ and } \sum_{\mathbf{x} \in \mathcal{X}(K)} \alpha(\mathbf{x}) = 1, \quad (2)$$

where $p(\mathbf{c}) \triangleq \Pr\{\mathbf{C}[t] = \mathbf{c}\}$ denotes the probability that the channel state is $\mathbf{c}$, and $\alpha(\mathbf{x})$ and $\beta(\mathbf{x}, \mathbf{c}; \mathbf{s})$ denote the probabilities of selecting the sampling schedule $\mathbf{x} \in \mathcal{X}(K)$ and selecting the transmission schedule $\mathbf{s} \in \mathcal{S}$ given the sampling schedule $\mathbf{x}$ and channel state $\mathbf{c}$, respectively.

*Remark*: In (1), the right-hand-side is the total average service rate provided for each user, and the left-hand-side is the user's average arrival rate. Thus, in order to ensure that the system is stable under some policy, (1) must be satisfied.

*Proof:* The proof follows a similar line of analysis as that in [14] and is omitted due to space limitation. We refer readers to our technical report [10] for proof details. ■

Having established the capacity outer bound $\Lambda(M, K)$, we are now in a position to develop an efficient and low-overhead uplink scheduling algorithm.

## V. Low-Overhead Uplink Scheduling Design

In this section, we consider an efficient and low-overhead uplink scheduling design based on the idea of pick-and-compare (PC). In particular, we first develop an iterative sampling and transmission algorithm with full information to achieve the capacity outer bound $\Lambda(M, K)$, which motivates

the development of an efficient and low-overhead uplink scheduling design. Then, we discuss the throughput deficiency of a natural variant of the proposed algorithm in the single-channel system under the stringent constraint that $K$ users are allowed to send their control messages. Finally, we propose a PC-based efficient low-complexity iterative sampling and transmission algorithms under the strict sampling constraint.

### *A. Iterative Sampling and Transmission Algorithm Design*

In this subsection, we develop an efficient joint sampling and transmission algorithm that can achieve the capacity outer bound $\Lambda(M,K)$. Although this algorithm requires full knowledge of channel state distributions and instantaneous queue lengths of all users, it provides a guideline for our design with the desired sampling constraint. We consider $M+1$ rounds in each time slot. By slightly abusing the notations, we use $Q_{i,j}[t]$ to denote the (virtual) queue-length of user $i$ at the end of round $j$, where $j=0,1,2,\ldots,M$, and $Q_{i,0}[t]=Q_i[t]$.

---

**Iterative Joint Sampling and Transmission (IJST) Algorithm:** In each time slot $t$, all users report their queue-lengths, i.e., $\mathbf{Q}[t]=(Q_i[t])_{i=1}^N$. Then, for each round $j=1,2,\ldots,M$, perform the following:

(1) *Sampling Decision*: Set the sampling vector $\mathbf{X}_j^*[t]$ as:

$$\mathbf{X}_j^*[t] \in \underset{\mathbf{X}_j\in\mathcal{X}_j(K)}{\arg\max}\ \mathbb{E}\left[\max_i Q_{i,j-1}[t]X_{i,j}C_{i,j}[t]\right], \qquad (3)$$

where $\mathbf{X}_j$ is the $j$-th column of a $N\times M$ matrix $\mathbf{X}$ and $\mathcal{X}_j(K)$ denotes the collection of sampling schedules on channel $j$ under the constraint that $K$ users send their control messages on channel $j$. Users with $X_{i,j}^*[t]=1$ are also required to send their control messages on channel $j$.

(2) *Transmission Scheduling Decision*: Schedule the transmission of user $i_j^*[t]$ on channel $j$ that satisfies:

$$i_j^*[t] \in \arg\max_i Q_{i,j-1}[t]X_{i,j}^*[t]C_{i,j}[t]. \qquad (4)$$

(3) *(Virtual) Queue-length Update*:

$$Q_{i_j^*[t],j}[t+1]=\left(Q_{i_j^*[t],j-1}[t]-C_{i_j^*[t],j}[t]\right)^+. \qquad (5)$$

After $M$-round decision making, users $\{i_j^*[t]\}_{j=1}^M$ transmit on their corresponding channels in the rest of time slot $t$.

---

Here, the IJST Algorithm uses the idea of iterative scheduling that is similar to that of [3], [7] in order to improve delay performance. This is due to the fact that users with the larger queue-lengths may have priority over multiple channels, and thus users with slightly smaller queue-lengths suffer from poor delay performance (see [2], [7]), especially when the number of channels is large. In the IJST Algorithm, all users need to report their queue-length information at the beginning of each time slot. Then, in the $j$-th round of the IJST Algorithm, (i) we first solve the optimization problem (3) to get the optimal sampling schedule $\mathbf{X}_j^*[t]$; (ii) users with $X_{i,j}^*[t]=1$ send their control message in order for the AP to acquire their channel state information; (iii) After collecting both queue-length and channel state information from users, user with the maximum product of queue-length and channel rate is selected for data transmission on channel $j$, and then the AP virtually updates the queue-length of the selected user. After $M$-round decision making, the selected users $\{i_j^*[t]\}_{j=1}^M$ are allowed for data transmission in the rest of the time slot $t$. The next proposition shows that the proposed IJST Algorithm can stabilize the system for any arrival rate vector strictly within the capacity outer bound $\Lambda(M,K)$.

*Proposition 2:* The IJST Algorithm achieves the capacity outer bound $\Lambda(M,K)$, i.e., for any arrival rate vector $\boldsymbol{\lambda}$ that is strictly inside $\Lambda(M,K)$, the IJST Algorithm stabilizes the system subject to the constraints of $K$ allowed sampling users on each channel.

*Proof:* Select the Lyapunov function $V(\mathbf{Q})=\frac{1}{2}\sum_{i=1}^N Q_i^2$ and follows the standard Lyapunov arguments. The details can be found in our technical report [10]. ■

Note that the IJST Algorithm incurs a large amount of communication overhead that is linearly increasing with the number of users $N$ before each data transmission. This is because the AP needs to know queue-length information of all $N$ users to solve the optimization problem in (3) to obtain the optimal sampling schedule $\mathbf{X}^*[t]$. This motivates us to investigate whether there exist efficient policies that only allow $K$ users to send their control messages on each channel, which significantly reduces the amount of communication overhead. Next, we provide an example to illustrate a non-trivial design of such policies starting from the single channel setting for the ease of exposition.

### *B. A Motivating Example of Low-Overhead Uplink Scheduling: From "Power-of-$K$-Choices" to "Pick-and-Compare"*

One way to reduce the amount of coordination by the AP in a single-channel wireless IoT uplink system works as follows: the AP randomly samples $K$ users and asks them to send their control messages at the beginning of each time slot, and then selects the user with the maximum product of queue-length and channel rate for data transmission in the rest of the time slot. This algorithm is called *Power-of-$K$-Choices*. Similar ideas have been explored in the context of load-balancing algorithms (e.g., [25], [12]) in data centers that distribute arriving jobs across servers with the goal of minimizing job waiting time. However, this algorithm suffers from a large throughput performance loss in wireless IoT uplink systems even in the single-channel setting.

To see the throughput inefficiency of the Power-of-$K$-Choices policy, we consider a single-channel uplink example with two groups of users without channel fading, where the first group has $\lceil \phi N\rceil$ users with the same mean arrival rate of $0.5/\lceil \phi N\rceil$ and the other has $N-\lceil \phi N\rceil$ users with the same mean arrival rate of $\lambda$, where $\phi\in(0,1)$ and $\lceil x\rceil$ denotes the minimum integer no smaller than $x$. Here, it is easy to see that the capacity region is $\{\lambda:(N-\lceil \phi N\rceil)\lambda<0.5\}$. For the Power-of-Two-Choices policy (i.e., when $K=2$),

the probability that at least one user sampled from the second group is $1-\binom{\lceil\phi N\rceil}{2}/\binom{N}{2}$. Therefore, the Power-of-Two-Choices policy can at most support the throughput region: $\left\{\lambda:(N-\lceil\phi N\rceil)\lambda<1-\binom{\lceil\phi N\rceil}{2}/\binom{N}{2}\right\}$. Thus, the second group of users suffer throughput loss by at least:

$$\frac{0.5-\left(1-\binom{\lceil\phi N\rceil}{2}/\binom{N}{2}\right)}{0.5}\times 100\%, \tag{6}$$

which amounts to $61.82\%$ when $N=100$ and $\phi=0.9$. This simple example shows that the Power-of-$K$-Choices policy suffers from large throughput degradation even in the single-channel and non-fading case, let alone in general settings with multiple channels and wireless channel fading. This is because the congested or heavily loaded users may not have an opportunity to be sampled and hence are not able to obtain service under the Power-of-$K$-Choices policy.

Interestingly, in the single-channel non-fading case, there is a variant of the Power-of-Two-Choices policy, known as the Pick-and-Compare (PC) algorithm (e.g., [22], [13], [4], [20]), which is known to be *throughput-optimal*. A PC-based scheme keeps track of the most congested user in the memory and compares its weight with a randomly selected user. The PC algorithm achieves the maximum throughput by gradually improving the scheduling decisions over time. However, we note that the PC algorithm in the literature only works under non-fading setting, while fading is the one of the key features in wireless communication channels. *So far, it remains unclear how to generalize the PC algorithm to the fading settings and still achieve throughput performance guarantee.* The main challenge in developing the PC algorithm for fading settings lies in the fact that the channel rates are time-varying and can change abruptly. This is very different from the smooth evolution of the queue-length process. In the next subsection, we will address this challenge and propose an efficient and low-overhead uplink scheduling algorithm. Moreover, this algorithm works for *general* multi-channel settings with fading.

### C. Iterative Pick-and-Compare Algorithm Design

In this subsection, we focus on the efficient uplink scheduling design under the stringent constraints that $K$ users are allowed to transmit their control messages on each channel. The key element in our approach is to decouple the optimization problem (3) such that it can be solved by only considering a small subset of users. To that end, we assume that the wireless fading channels satisfy the following assumption.

*Assumption 1:* For any given non-negative numbers $n_1,n_2,\ldots,n_N$, there exists a stochastic order among random variables $n_1C_{1,j},n_2C_{2,j},\ldots,n_NC_{N,j}$, i.e., there exists a permutation $(m_1,m_2,\ldots,m_N)$ of $(1,2,\ldots,N)$ such that

$$n_{m_1}C_{m_1,j}\geq_{st} n_{m_2}C_{m_2,j}\geq_{st}\ldots\geq_{st} n_{m_N}C_{m_N,j}, \tag{7}$$

where $j=1,2,\ldots,M$. Here, $Z_1\geq_{st}Z_2$ means that random variable $Z_1$ is stochastically greater than random variable $Z_2$ (see [19]), i.e., $\Pr\{Z_1>z\}\geq\Pr\{Z_2>z\},\forall z\in\mathbb{R}$.

*Remark*: If channel states are i.i.d., then (7) trivially holds.

Assumption 1 provides an opportunity for decoupling the optimization problem (3) by only allowing a small portion of users to be sampled in order for the AP to obtain system state information of sampled users. *Indeed, if Assumption 1 does not hold, it is almost impossible to obtain the optimal value of* (3) *by only collecting information from a small subset of users due to the abrupt changes of channel rates.* Next, we incorporate wireless channel fading to generalize the traditional PC algorithmic design in the general multi-channel systems. Similar to the IJST Algorithm, we use $Q_{i,j}[t]$ to denote the queue-length of user $i$ at the end of round $j$, $j=0,1,2,\ldots,M$, and $Q_{i,0}[t]=Q_i[t]$. Also, we use $C_{i,j}[t]$ to denote the $j$-th channel rate of user $i$ in time slot $t$, while $C_{i,j}$ without time index $t$ denotes a random variable with the same distribution as the $j$-th channel rate of user $i$.

---

**Iterative Pick and Compare (IPC) Algorithm**: In each time slot $t$, given users $(\hat{i}_{k,j}[t-1],k=1,2,\ldots,K-1,j=1,2,\ldots,M)$ selected by the IPC Algorithm in time slot $t-1$, perform the following: For each channel $j=1,2,\ldots,M$,

(1) *Pick Phase*: Randomly pick one user $r_j[t]$, and ask it to report its current queue-length $Q_{r_j[t]}[t]$ and channel state $C_{r_j[t],j}[t]$ on channel $j$ to the AP.

(2) *Report Phase*: Ask users $(\hat{i}_{k,j}[t-1])_{k=1}^{K-1}$ to report their queue-lengths and channel states of channel $j$ to the AP.

(3) *Compare Phase (Transmission Scheduling)*: Determine the transmission schedule of user $i$ on channel $j$ that satisfies:

$$\widetilde{i}_j[t]\in\arg\max_i\left\{Q_{i,j-1}[t]\hat{X}_{i,j}[t]C_{i,j}[t]\right\}, \tag{8}$$

where $\hat{X}_{i,j}[t]=1$ if $i\in\left\{r_j[t],\hat{i}_{k,j}[t-1],\forall k=1,2,\ldots,K-1\right\}$, and $\hat{X}_{i,j}[t]=0$ otherwise.

(4) *Update Phase*: Select users $(\hat{i}_{k,j}[t])_{k=1}^{K-1}$ that achieve the $K-1$ largest $Q_l[t]C_{l,j}$ among users $(\hat{i}_{k,j}[t-1])_{k=1}^{K-1}$ and the newly reporting user $r_j[t]$ in the stochastic ordering sense, i.e.,

$$(\hat{i}_{k,j}[t])_{k=1}^{K-1}\in\underset{l\in\{(\hat{i}_{k,j}[t-1])_{k=1}^{K-1},r_j(t)\}}{\arg\max}\{Q_l[t]C_{l,j}\}, \tag{9}$$

where $Q_{\hat{i}_{1,j}[t]}[t]C_{\hat{i}_{1,j}[t],j}\geq_{st}\ldots\geq_{st}Q_{\hat{i}_{K-1,j}[t]}[t]C_{\hat{i}_{K-1,j}[t],j}$.

(5) *(Virtual) Queue-length Update*:

$$Q_{\widetilde{i}_j[t],j}[t+1]=\left(Q_{\widetilde{i}_j[t],j-1}[t]-C_{\widetilde{i}_j[t],j}[t]\right)^+. \tag{10}$$

After $M$-round decision making, users $\{\widetilde{i}_j[t]\}_{j=1}^M$ transmit on their corresponding channels in the rest of the time slot $t$.

---

In the IPC Algorithm, the AP exactly requires $K$ sampling users $\left\{(\hat{i}_k[t-1])_{k=1}^{K-1},r(t)\right\}$ on each channel. This significantly reduces the amount of coordination from the AP compared to the IJST Algorithm. Next, we will show that the IPC Algorithm still possesses excellent throughput performance.

*Proposition 3:* Suppose that the channel state $\mathbf{C}_j=(C_{i,j})_{i=1}^N$ on each channel $j$ satisfies Assumption 1. Then, for any arrival rate vector $\boldsymbol{\lambda}=(\lambda_i)_{i=1}^N$ that is strictly inside the rate region $\Lambda(M,K-1)$, the IPC Algorithm stabilizes the

system subject to the constraints that $K$ users are allowed to send control messages on each channel.

*Proof:* The key step is to establish that the IPC Algorithm performs similarly as its centralized counterpart (i.e., the IJST Algorithm) does, i.e.,

$$\Pr\Bigg\{\mathbb{E}\left[\max_{k=1,2,\ldots,K-1} Q_{\hat{i}_{k,j}[t],j}[t]C_{\hat{i}_{k,j}[t],j}\middle|\mathbf{Q}[t]\right] \geq \mathbb{E}\left[\max_{k=1,2,\ldots,K-1} Q_{i^*_{k,j}[t],j}[t]C_{i^*_{k,j}[t],j}\middle|\mathbf{Q}[t]\right] - G_{\gamma,j}\Bigg\} \geq 1-\gamma, \forall j=1,2,\ldots,M, \quad (11)$$

holds for any $\gamma \in (0,1)$, where $G_{\gamma,j} > 0$ is some constant. The rest of the proof follows a Lyapunov analysis for which the detailed proof can be found in Appendix A. ∎

Note that under the constraint of $K$ sampling users for each channel, the maximum throughput region that can be achieved by any feasible algorithm is at most $\Lambda(M,K)$, while our IPC Algorithm can achieve the throughput region $\Lambda(M,K-1)$ with the same amount of communication overhead. Here, it is worth pointing out that the throughput region $\Lambda(M,K-1)$ is *independent* of the number of users in the system. This yields a significant advantage over the IJST Algorithm especially for a large number of users, which is the typical case in most IoT applications.

Moreover, the achieved throughput region $\Lambda(M,K-1)$ is close to the capacity outer bound $\Lambda(M,K)$ even for a small $K$ when the number of channels $M$ is small. For example, in a single-channel case of $N$ users with i.i.d. ON-OFF fading with $p = \Pr\{C[t]=1\}$, then $\Lambda(1,K-1) = \{\lambda : N\lambda < 1-(1-p)^{K-1}\}$ and $\Lambda(1,K) = \{\lambda : N\lambda < 1-(1-p)^{K}\}$. Thus, the throughput performance loss is at most $\left(1-\left(1-(1-p)^{K-1}\right)/\left(1-(1-p)^{K}\right)\right)\times 100\%$. Fig. 1 shows the throughput performance loss percentage under the IPC Algorithm when $p=0.8$. We can observe from Fig. 1 that the throughput loss decays *exponentially* fast with the increase of the number of allowed sampling users $K$, and is at most $3.23\%$ even when $K=3$. Therefore, the throughput performance loss under the IPC Algorithm is small.

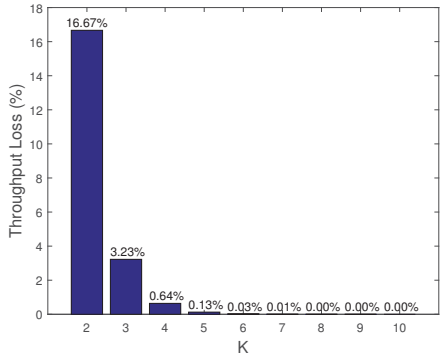
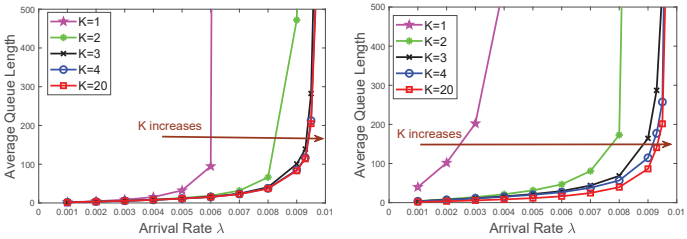



Fig. 1: Throughput loss in a single-channel system

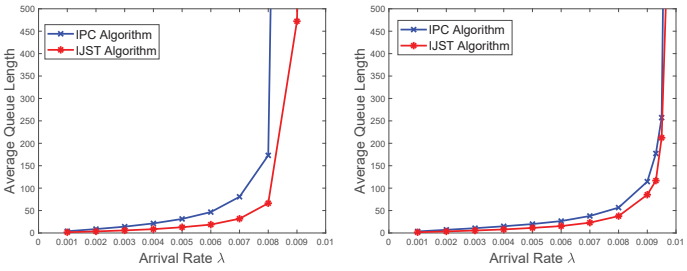
## VI. Numerical Results

In this section, we perform simulations for our proposed low-overhead IPC Algorithm and compare it to the IJST Algorithm in both single-channel and multi-channel cases. In the simulations, we consider $N=20$ users. We assume that the number of arrivals occurring at each user in each time slot follows a Bernoulli distribution with mean $\lambda$. In order to capture the burstiness feature of IoT traffic, we assume that each incoming arrival brings $F$ packets, where $F$ is equal to 20 with probability $4/19$ and 1 otherwise. Therefore, the expected number of packets that each arrival carries is equal to 5, i.e., $\mathbb{E}[F]=5$. We consider ON-OFF channel fading models that are independently distributed over users and i.i.d. over time, where the first ten users have channel availability probability of 0.9 and all others have probability of 0.5. We assume that all $M$ channels have the same channel fading model.




(a) IJST Algorithm (b) IPC Algorithm

Fig. 2: Impact of number of sampling users $K$: single channel



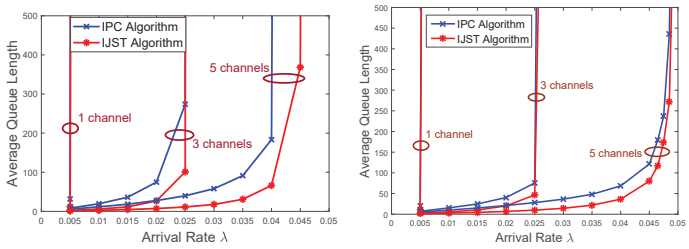

(a) $K=2$ (b) $K=4$

Fig. 3: Performance comparison: single channel case

Fig. 2 shows the impact of the number of sampling users $K$ on the system performance of the IJST Algorithm and the IPC Algorithm in the single-channel case. From Figs. 2a and 2b, we can observe that as $K$ increases, both throughput and delay performance of these algorithms improve. Especially, we can see that $K=4$ sampling users are sufficient for both algorithms to almost achieve the maximum throughput (i.e., when $K=20$). Moreover, the delay performance under the IPC Algorithm is only slightly worse than that under the IJST Algorithm, and their gap becomes smaller as $K$ increases, as shown in Fig. 3b. This indicates that in the single channel case, the IPC Algorithm with only four sampling users can achieve almost the same throughput and delay performance as the IJST Algorithm, which requires all queue-length information available before each data transmission and thus requires a significant amount of communication overhead. Hence, our proposed IPC Algorithm dramatically reduces the communication overhead with a negligible performance loss.

In Fig. 4, we study the performance of our proposed IPC Algorithm in a multi-channel case and compare it to the IJST

Algorithm. From Figs. 4a and 4b, we can observe that our proposed IPC Algorithm still performs well in both three and five channel cases compared to the IJST Algorithm when the number of allowable sampling users is four. This indicates that the IPC Algorithm is quite robust to the number of channels, which is significantly more advantageous in large-scale multi-channel IoT uplink systems.




(a) $K = 2$ (b) $K = 4$

Fig. 4: Performance comparison: multi-channel case

## VII. Conclusions

In this paper, we considered the design of efficient and low-overhead uplink scheduling algorithms for large-scale IoT applications, where the central controller has a limited amount of information about the users. We first derived a capacity outer bound under the sampling constraint, where only a small subset of users are allowed to use control channels for system state reporting and channel probing. Then, we proposed a joint sampling and transmission algorithm with full information before each transmission and show that it achieves the capacity outer bound. However, this algorithm incurs a huge amount of communication overhead before each data transmission. To that end, we developed an efficient and low-overhead uplink scheduling algorithm that is suitable for large-scale IoT applications. Finally, we validated our theoretical results through extensive simulations.

## Appendix A
## Proof of Proposition 3

Since the channel states on each channel satisfy Assumption 1, the IJST Algorithm selects the $K$ largest $Q_{i,j}[t]C_{i,j}$ among $N$ users on channel $j+1$ ($j = 0, 1, \ldots, M-1$) in the stochastic order sense given $\mathbf{Q}[t] = (Q_i[t])_{i=1}^N$, i.e.,

$$Q_{i^*_{1,j}[t],j}[t]C_{i^*_{1,j}[t],j} \geq_{st} \cdots \geq_{st} Q_{i^*_{K,j}[t],j}[t]C_{i^*_{K,j}[t],j} \geq_{st} Q_{i,j}[t]C_{i,j}, \quad (12)$$

where $\{i^*_{k,j}[t]\}_{k=1}^K$ are the users selected by the IJST Algorithm for reporting their channel states and $i \notin \{i^*_{k,j}[t]\}_{k=1}^K$.

Since the IPC Algorithm independently picks a user at uniformly random on each channel in each time slot, for any given $\gamma > 0$, there exists a $D_{\gamma,j} > 0$ such that

$$\Pr\left\{r_j[\tau_k] = i^*_{k,j}[t] \text{ for some } \tau_k \in \{t - D_{\gamma,j}, \ldots, t-1\}, \forall k = 1, 2 \ldots, K-1\right\} \geq 1 - \gamma. \quad (13)$$

Under the IPC Algorithm, we have

$$\begin{aligned}
&\max_{k=1,2,\ldots,K-1} Q_{\hat{i}_{k,j}[t],j}[t]C_{\hat{i}_{k,j}[t],j} \\
&\overset{(a)}{\geq}_{st} \max_{k=1,2,\ldots,K-1} Q_{\hat{i}_{k,j}[t-1],j}[t]C_{\hat{i}_{k,j}[t-1],j} \\
&\overset{(b)}{\geq}_{st} \max_{k=1,2,\ldots,K-1} \left(Q_{\hat{i}_{k,j}[t-1],j}[t-1] - c_{\max}\right) C_{\hat{i}_{k,j}[t-1],j} \\
&\geq_{st} \max_{k=1,2,\ldots,K-1} Q_{\hat{i}_{k,j}[t-1]}[t-1]C_{\hat{i}_{k,j}[t-1],j} - c^2_{\max}, \quad (14)
\end{aligned}$$

where step $(a)$ follows the definition of the IPC Algorithm, and $(b)$ uses the fact that at most $c_{\max}$ packets can be served on each channel in each time slot.

Without loss of generality, we assume that $\tau_{m_1} > \tau_{m_2} > \ldots > \tau_{m_{K-1}}$, where $(m_1, m_2, \ldots, m_{K-1})$ is a permutation of $(1, 2, \ldots, K-1)$. Hence, we have

$$\begin{aligned}
&\max_{k=1,2,\ldots,K-1} Q_{\hat{i}_{k,j}[t],j}[t]C_{\hat{i}_{k,j}[t],j} \\
&\overset{(a)}{\geq}_{st} \max_{k=1,2,\ldots,K-1} Q_{\hat{i}_{k,j}[\tau_{m_1}],j}[\tau_{m_1}]C_{\hat{i}_{k,j}[\tau_{m_1}],j} - (t - \tau_{m_1})c^2_{\max} \\
&\overset{(b)}{\geq}_{st} \max\left\{ \max_{k=1,2,\ldots,K-2} Q_{\hat{i}_{k,j}[\tau_{m_1}-1],j}[\tau_{m_1}]C_{\hat{i}_{k,j}[\tau_{m_1}-1],j}, \right. \\
&\qquad \left. Q_{r_j[\tau_{m_1}],j}[\tau_{m_1}]C_{r_j[\tau_{m_1}],j} \right\} - (t - \tau_{m_1})c^2_{\max} \\
&\overset{(c)}{\geq}_{st} \max\left\{ \max_k \left(Q_{\hat{i}_{k,j}[\tau_{m_1}-1],j}[\tau_{m_1}-1] - c_{\max}\right) C_{\hat{i}_{k,j}[\tau_{m_1}-1],j}, \right. \\
&\qquad \left. Q_{r_j[\tau_{m_1}],j}[\tau_{m_1}]C_{r_j[\tau_{m_1}],j} \right\} - (t - \tau_{m_1})c^2_{\max} \\
&\geq_{st} \max\left\{ \max_k Q_{\hat{i}_{k,j}[\tau_{m_1}-1],j}[\tau_{m_1}-1]C_{\hat{i}_{k,j}[\tau_{m_1}-1],j} - c^2_{\max}, \right. \\
&\qquad \left. Q_{r_j[\tau_{m_1}],j}[\tau_{m_1}]C_{r_j[\tau_{m_1}],j} \right\} - (t - \tau_{m_1})c^2_{\max} \\
&\geq_{st} \max\left\{ \max_{k=1,2,\ldots,K-2} Q_{\hat{i}_{k,j}[\tau_{m_1}-1]}[\tau_{m_1}-1]C_{\hat{i}_{k,j}[\tau_{m_1}-1],j}, \right. \\
&\qquad \left. Q_{r_j[\tau_{m_1}],j}[\tau_{m_1}]C_{r_j[\tau_{m_1}],j} \right\} - (t - \tau_{m_1} + 1)c^2_{\max}, \quad (15)
\end{aligned}$$

where step $(a)$ iteratively uses (14); $(b)$ follows the definition of the IPC Algorithm; $(c)$ uses the fact that at most $c_{\max}$ packets can be delivered on each channel in each time slot.

By using the similar argument in deriving (15), we can show

$$\begin{aligned}
&\max_{k=1,2,\ldots,K-l} Q_{\hat{i}_{k,j}[\tau_{m_l}-1],j}[\tau_{m_l}-1]C_{\hat{i}_{k,j}[\tau_{m_l}-1],j} \\
&\geq_{st} \max\left\{ Q_{r_j[\tau_{m_{l+1}}],j}[\tau_{m_{l+1}}]C_{r_j[\tau_{m_{l+1}}],j}, \right. \\
&\qquad \left. \max_{k=1,\ldots,K-(l+1)} Q_{\hat{i}_{k,j}[\tau_{m_{l+1}}-1],j}[\tau_{m_{l+1}}-1]C_{\hat{i}_{k,j}[\tau_{m_{l+1}}-1],j} \right\} \\
&\qquad - (\tau_{m_l} - \tau_{m_{l+1}})c^2_{\max}, \quad \forall l = 1, 2, \ldots, K-2. \quad (16)
\end{aligned}$$

By using (15) and (16), we have

$$\begin{aligned}
&\max_{k=1,2,\ldots,K-1} Q_{\hat{i}_{k,j}[t],j}[t]C_{\hat{i}_{k,j}[t],j} \\
&\geq_{st} \max_{k=1,2,\ldots,K-1} Q_{r_j[\tau_k],j}[\tau_k]C_{r_j[\tau_k],j} - (t - \tau_{m_{K-1}} + 1)c^2_{\max} \\
&\overset{(a)}{=}_{st} \max_{k=1,2,\ldots,K-1} Q_{i^*_{k,j}[t],j}[\tau_k]C_{i^*_{k,j}[t],j} - (t - \tau_{m_{K-1}} + 1)c^2_{\max}
\end{aligned}$$

$$
\begin{aligned}
&\overset{(b)}{\geq}_{st} \max_{k=1,2,\ldots,K-1} \left(Q_{i^*_{k,j}[t],j}[t] - (t-\tau_k)A_{\max}\right) C_{i^*_{k,j}[t],j} \\
&\qquad - (t-\tau_{m_{K-1}}+1)c^2_{\max} \\
&\overset{(c)}{\geq}_{st} \max_{k=1,2,\ldots,K-1} Q_{i^*_{k,j}[t],j}[t] C_{i^*_{k,j}[t],j} \\
&\qquad - (t-\tau_{m_{K-1}}+1)\left(A_{\max}+c_{\max}\right)c_{\max} \\
&\geq_{st} \max_{k=1,2,\ldots,K-1} Q_{i^*_{k,j}[t],j}[t] C_{i^*_{k,j}[t],j} - G_{\gamma,j}, \qquad (17)
\end{aligned}
$$

where step $(a)$ uses the definition of $\tau_k, \forall k = 1, 2, \ldots, K-1$; $(b)$ follows the fact that at most $A_{\max}$ packets arrives at each queue in each time slot; $(b)$ is true since the maximum channel rate is $c_{\max}$; $(c)$ is true for $G_{\gamma,j} \triangleq (D_{\gamma,j}+1)\left(A_{\max}+c_{\max}\right)c_{\max}$.

According to the property of the stochastic ordering, (17) implies that

$$
\begin{aligned}
&\mathbb{E}\left[\max_{k=1,2,\ldots,K-1} Q_{\hat{i}_{k,j}[t],j}[t] C_{\hat{i}_{k,j}[t],j} \middle| \mathbf{Q}[t]\right] \\
\geq &\mathbb{E}\left[\max_{k=1,2,\ldots,K-1} Q_{i^*_{k,j}[t],j}[t] C_{i^*_{k,j}[t],j} \middle| \mathbf{Q}[t]\right] - G_{\gamma,j}. \qquad (18)
\end{aligned}
$$

By combining (18) and (13), we have

$$
\begin{aligned}
&\Pr\Bigg\{\mathbb{E}\left[\max_{k=1,2,\ldots,K-1} Q_{\hat{i}_{k,j}[t],j}[t] C_{\hat{i}_{k,j}[t],j} \middle| \mathbf{Q}[t]\right] \\
&\qquad \geq \mathbb{E}\left[\max_{k=1,2,\ldots,K-1} Q_{i^*_{k,j}[t],j}[t] C_{i^*_{k,j}[t],j} \middle| \mathbf{Q}[t]\right] - G_{\gamma,j}\Bigg\} \\
&\geq 1-\gamma, \qquad (19)
\end{aligned}
$$

which implies that

$$
\begin{aligned}
&\sum_{j=1}^{M} \mathbb{E}\left[\max_{k=1,2,\ldots,K-1} Q_{\hat{i}_{k,j}[t],j}[t] C_{\hat{i}_{k,j}[t],j} \middle| \mathbf{Q}[t]\right] \\
&\overset{(a)}{\geq}(1-\gamma)\sum_{j=1}^{M} \mathbb{E}\left[\max_{k=1,2,\ldots,K-1} Q_{i^*_{k,j}[t],j}[t] C_{i^*_{k,j}[t],j} \middle| \mathbf{Q}[t]\right] - \overline{B}_1 \\
&\overset{(b)}{=}(1-\gamma)\sum_{j=1}^{M} \mathbb{E}\left[\max_{i} Q_{i,j}[t] X^*_{i,j} C_{i,j} \middle| \mathbf{Q}[t]\right] - \overline{B}_1,
\end{aligned}
$$

where step $(a)$ is true for $\overline{B}_1 \triangleq (1-\gamma)\sum_{j=1}^{M} G_{\gamma,j}$ and follows from the fact that $\mathbb{E}\left[\max_k Q_{i^*_{k,j}[t],j}[t] C_{i^*_{k,j}[t],j} \middle| \mathbf{Q}[t]\right]$ is a fixed value; $(b)$ is true since $X^*_{i,j} = 1$ if $i \in \{i^*_{k,j}, k = 1, 2, \ldots, K-1\}$. This indicates that the selected weight by the IPC Algorithm is very close to the IJST Algorithm. The rest of the proof follows the standard Lyapunov arguments. The detailed proof is available in our technical report [10].

## REFERENCES


[1] Gartner newsroom, February 2017. http://www.gartner.com/newsroom/id/3598917.

[2] S. Bodas, S. Shakkottai, L. Ying, and R. Srikant. Scheduling in multi-channel wireless networks: Rate function optimality in the small-buffer regime. In *ACM Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems (SIGMETRICS)*, pages 121–132, 2009.

[3] S. Bodas, S. Shakkottai, L. Ying, and R. Srikant. Low-complexity scheduling algorithms for multichannel downlink wireless networks. *IEEE/ACM Transactions on Networking (TON)*, 20(5):1608–1621, 2012.

[4] A. Eryilmaz, A. Ozdaglar, and E. Modiano. Polynomial complexity algorithms for full utilization of multi-hop wireless networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 499–507. IEEE, 2007.

[5] A. Eryilmaz, R. Srikant, and J. R. Perkins. Stable scheduling policies for fading wireless channels. *IEEE/ACM Transactions on Networking*, 13:411–425, April 2005.

[6] J. Huang, V. G. Subramanian, R. Agrawal, and R. Berry. Joint scheduling and resource allocation in uplink ofdm systems for broadband wireless access networks. *IEEE Journal on Selected Areas in Communications*, 27(2):226–234, 2009.

[7] B. Ji, X. Lin, and N. B. Shroff. Advances in multi-channel resource allocation: Throughput, delay, and complexity. *Synthesis Lectures on Communication Networks*, 9(1):1–130, 2016.

[8] L. Jiang and J. Walrand. A distributed csma algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Transactions on Networking (ToN)*, 18(3):960–972, 2010.

[9] B. Li and A. Eryilmaz. Distributed channel probing for efficient transmission scheduling in wireless networks. *IEEE Transactions on Mobile Computing*, 14(6):1176–1188, 2015.

[10] B. Li, B. Ji, and J. Liu. Efficient and low-overhead scheduling design for large-scale wireless uplink systems. Technical Report. Available at http://www.ele.uri.edu/faculty/binli/papers/WiOPT2018UplinkScheduling.pdf.

[11] C.-p. Li and M. J. Neely. Energy-optimal scheduling with dynamic channel acquisition in wireless downlinks. *IEEE Transactions on Mobile Computing*, 9(4):527–539, 2010.

[12] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.

[13] E. Modiano, D. Shah, and G. Zussman. Maximizing throughput in wireless networks via gossiping. In *ACM SIGMETRICS/IFIP Performance*, 2006.

[14] M. Neely, E. Modiano, and C. Rohrs. Dynamic power allocation and routing for time varying wireless networks. In *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, San Francisco, CA, April 2003.

[15] M. J. Neely. Order optimal delay for opportunistic scheduling in multi-user wireless uplinks and downlinks. *IEEE/ACM Transactions on Networking*, 16(5):1188–1199, 2008.

[16] J. Ni, B. Tan, and R. Srikant. Q-csma: Queue-length-based csma/ca algorithms for achieving maximum throughput and low delay in wireless networks. *IEEE/ACM Transactions on Networking*, 20(3):825–836, 2012.

[17] M. Ouyang and L. Ying. Approaching throughput optimality with limited feedback in multichannel wireless downlink networks. *IEEE/ACM Transactions on Networking (TON)*, 21(6):1827–1838, 2013.

[18] S. Rajagopalan, D. Shah, and J. Shin. Network adiabatic theorem: an efficient randomized protocol for contention resolution. In *ACM SIGMETRICS performance evaluation review*, volume 37, pages 133–144. ACM, 2009.

[19] S. M. Ross. *Stochastic Processes*. John Wiley & Sons, 1995.

[20] S. Sanghavi, L. Bui, and R. Srikant. Distributed link scheduling with constant overhead. *ACM SIGMETRICS Performance Evaluation Review*, 35(1):313–324, 2007.

[21] S. Shakkottai and A. Stolyar. Scheduling for multiple flows sharing a time-varying channel: The exponential rule. *American Mathematical Society Translations, Series 2*, 207:185–202, 2002.

[22] L. Tassiulas. Linear complexity algorithms for maximum throughput in radio networks and input queued switches. In *Proceedings of IEEE Infocom*, pages 533–539, 1998.

[23] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 36:1936–1948, December 1992.

[24] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory*, 39:466–478, March 1993.

[25] N. D. Vvedenskaya, R. L. Dobrushin, and F. I. Karpelevich. Queueing system with selection of the shortest of two queues: An asymptotic approach. *Problemy Peredachi Informatsii*, 32(1):20–34, 1996.

[26] S.-Y. Yun, J. Shin, and Y. Yi. Csma over time-varying channels: optimality, uniqueness and limited backoff rate. In *Proceedings of the fourteenth ACM international symposium on Mobile ad hoc networking and computing*, pages 137–146. ACM, 2013.