Computationally Efficient, Stable Scheduling for Wireless Systems with Limited Probing

Joseph Lubars*, R. Srikant*, and Lei Ying†
*University of Illinois at Urbana-Champaign {lubars2, rsrikant}@illinois.edu
†Arizona State University lei.ying.2@asu.edu

***Abstract*—Modern cellular base stations can transmit over multiple frequencies, and further choose to transmit to different users over different frequencies. In much of the prior literature, it is assumed that the channel state of each user over each frequency is known. However, to get such channel state information for each user-channel pair requires a large overhead. Here, we consider the problem of computationally efficient and throughput-optimal scheduling in networks where the base station ensures a small probing overhead by limiting the number of allowed probe packets per time slot. We first argue that a naive optimization-based MaxWeight algorithm is combinatorially infeasible to implement, and then design a low-complexity algorithm that achieves the same throughput as the naive MaxWeight algorithm. Through simulations, we also investigate further improvements to achieve very small packet delays.**

I. Introduction

In a wireless cellular system, a base station must transmit data to a large number of users as efficiently as possible. Each base station has access to many carrier frequencies, which can each independently be used to transmit to a particular user. The overall throughput of the data transmission is affected greatly by the scheduling algorithm used to assign a user to each frequency for a given time slot. These time slots are also short, so it is necessary to have an efficiently computable scheduling algorithm.

It is well known that the state of a wireless channel can be time-varying due to fading and shadowing effects. In order to better plan a schedule for each time slot, the base station can send a probing signal to its users, using the response to gauge the channel quality. However, because only one user at a time can respond over each frequency, the time to obtain feedback from all the users can be large, leading to reduced network throughput. One solution is to limit the number of probes allowed in each time slot, which is the model used in this paper. If the number of allowed probes is limited, then the scheduling problem becomes one of probing and scheduling, i.e., one has to first decide which user-frequency pairs to probe, and based on the feedback received, then decide which users to schedule over the different frequencies.

In this paper, each channel is an on-off channel with an underlying probability of being on, representing the channel quality. Each probe reveals the hidden state of the channel for that particular time slot. Packets arrive at the base station and are queued for transmission to particular users. The goal is to find a throughput-optimal scheduling algorithm given a fixed number of probes per frequency. It is well known that the MaxWeight algorithm can provide throughput-optimal algorithms for such systems. However, the MaxWeight algorithm may not always be efficiently computable. We will show how the throughput-optimal algorithm provided by the MaxWeight framework can be reframed into something that is much more computationally tractable.

The main contributions of the paper are as follows:

- We design a computationally efficient and throughput-optimal joint probing and scheduling algorithm for cellular networks in which limited probing is allowed per time slot.
- In the traditional MaxWeight algorithm where all user-frequency pairs are assumed to be probed, one does not require any knowledge of the statistics of the channel state (i.e., the probability that the channel is in an ON state). Similarly, when no probing is required, again it can be shown that channel statistics are not needed to design a throughput -optimal algorithm. However, when only limited probing is allowed, we present an example to show that the knowledge of channel statistics is needed to achieve throughput optimality.
- The basic version of the computationally efficient, throughput-optimal algorithm can be further modified to maintain throughput optimality while reducing the observed packet delays. We investigate two modifications through simulations and show that they significantly reduce packet delays.
- Through simulations, we also investigate the amount of channel state information required to achieve very small packet delays. Somewhat surprisingly, our simulations show that a small number of channel probes is sufficient to achieve very small packet delays.

A. Related Work

The MaxWeight algorithm was originally designed in [1], and its connection to convex optimization was shown in [2]–[5]; see [6] for a summary. Several modifications to the basic algorithm were introduced in [7]–[9] for the case of multiple-frequency systems; see [10] for a survey. However, all of these papers assume perfect channel state feedback. In the context of our current paper, this essentially means that there is no limitation on the number of probes allowed per time slot.

The problem where channel states are unknown and one has to use probing to learn them has been considered in [11], [12]. However, our focus is on the computational complexity of the resulting algorithm, which is not considered in either work. Instead, [11] focuses on the question of whether channel statistics are necessary for scheduling purposes, but in our case estimating the channel statistics is easy, so we do not emphasize this issue in our paper. In [12], the authors focus on the communication overhead of communicating queue lengths.

We also note that there is other related work on partial channel state information (see [13] and references within), but the results there are asymptotic in the size of the system whereas we focus on exact throughput optimality.

II. The Case of One Carrier Frequency

As mentioned in the introduction, the reason for limiting the number of probes in each time slot is to limit the time overhead needed to collect feedback from all the users in the network. This overhead issue arises even where there is only one frequency and there are a large number of users. Therefore, in this section, we consider the case of a single carrier frequency and generalize the result for this case to the case of multiple carrier frequencies in a subsequent section.

A. Problem Statement

We assume for now that there are n users and one carrier frequency. At each time slot, the channel from the base station to user i is ON with probability μ_i (allowing one packet to be transmitted), or OFF otherwise. Each user i generates packets in each time slot according to some stochastic process, with a mean rate λ_i and some finite variance. It is assumed that the packet arrival processes for the users are independent. We denote the vector of arrival rates by λ. The base station can probe k users (channels) at each time slot, and then choose to transmit to one user. We would like to find the capacity region of this system, and an algorithm that achieves this capacity region.

In each time slot, the action chosen by a joint probing and scheduling algorithm is as follows: Choose k different channels to probe, in order. Call these channels $\mathbf{u} = (u_1, \ldots, u_k)$. Given a result $r \in \{0,1\}^k$, where r_i encodes whether probe i was successful, the algorithms decides to transmit on channel t_r. Therefore, an action can be written as:

$$(\mathbf{u}, t), \text{where } \mathbf{u} \in \binom{[n]}{k}, t \in [n]^{2^k}$$

For ease of notation, we will let $\mathscr{A} = \binom{[n]}{k} \times [n]^{2^k}$ denote the set of possible actions, where $[n] = \{1, \ldots, n\}$.

B. The Capacity Region

Suppose action $(\mathbf{u}, t)$ is used. Then, the probability of each result $p(r|\mathbf{u})$ is $\prod_i \mu_{u_i}^{r_i}(1-\mu_{u_i})^{1-r_i}$. For that result, define $\tilde{\mu}_{t_r}$ for the transmission channel t_r as follows:

$$\tilde{\mu}_{t_r} = \begin{cases} r(t_r) & : t_r \in \mathbf{u} \\ \mu_{t_r} & : t_r \notin \mathbf{u} \end{cases}$$

We define $r(x)$ to be the element of r corresponding to channel x. The probability of successful transmission on t_r with result r is then:

$$p(r|\mathbf{u})\tilde{\mu}_{t_r}$$

And therefore, if we construct the n-dimensional throughput vector $v(\mathbf{u}, t)$ where the i-th component of v contains the expected number of packets transmitted to user i, as a combination of identity vectors e_{t_r}, we get

$$v(\mathbf{u}, t) = \sum_{r \in \{0,1\}^k} p(r|\mathbf{u})\tilde{\mu}_{t_r} e_{t_r}.$$

For any $(\mathbf{u}, t)$, it is easy to show that $\lambda < v(\mathbf{u}, t)$ is a necessary condition for stability. Furthermore, available throughput is linear in the set of actions. For any $\mathbf{u}', t', \alpha$, we must have $\lambda < \alpha v(\mathbf{u}, t) + (1-\alpha)v(\mathbf{u}', t')$. We omit the proof which is standard (see [6], for example), but following this reasoning, the capacity region is the convex hull of the throughput vectors:

$$\mathcal{C} = Co(\{v(\mathbf{u}, t) : (\mathbf{u}, t) \in \binom{[n]}{k} \times [n]^{2^k}\})$$

C. Main Results

Given a capacity region in the form of a convex hull of throughput vectors, the MaxWeight policy can be easily derived as maximizing the inner product of the expected throughput vector and the queue length vector over the set of possible actions. We state the following theorem without proof, which states the throughput optimality of the MaxWeight policy (for details, see [6], for example):

Theorem 1. *For any time slot, let $\mathbf{q} = (q_1, \ldots, q_n)$ be the current vector of queue lengths. Let $\mathbf{u}^*$ and t^* be the solutions to*

$$(\mathbf{u}^*, t^*) = \underset{\mathbf{u} \in \binom{[n]}{k}, t \in [n]^{2^k}}{\arg\max} \langle v(\mathbf{u}, t), \mathbf{q} \rangle \tag{1}$$

Then, the policy which probes the channels $\mathbf{u}^*$, obtains a result r, and transmits on channel t_r^* is throughput optimal. $\diamond$

Although this policy is throughput optimal, it is computationally intractable. Since there are 2^k possible values for r, naively performing the optimization in (1) requires $\Omega(n^{2^k}2^k)$ time. With about one hundred users connecting to the base-station, this is unreasonable for even a small number of probes. Ideally, we would like the computational complexity to be nearly linear in terms of number of users and probes.

In fact, a much lower complexity is achievable. We will show that the MaxWeight algorithm can take the form of a much simpler optimization problem, and this problem can be solved in $\tilde{O}(nk)$ time:

Theorem 2. *For any time slot, let $\mathbf{q} = (q_1, \ldots, q_n)$ be the current vector of queue lengths. Let $(u_1, \ldots, u_{k+1})$ solve:*

$$\max_{(u_1, \ldots, u_{k+1})} \sum_{i=1}^{k+1} \left(\mu_{u_i} q_{u_i} \prod_{m=1}^{i-1} (1 - \mu_{u_m}) \right), \tag{2}$$

where the maximization is taken over the set of $(u_1, \ldots, u_{k+1}) \in \binom{n}{k+1}$ *such that* $q_{u_1} \geq q_{u_2} \geq \ldots \geq q_{u_{k+1}}$. *Then, consider the policy which probes channels* $u_1, \ldots, u_k$ *in that order, transmitting on the first successfully probed channel, or on channel* u_{k+1} *if all probes are unsuccessful. This policy is throughput optimal.* $\diamond$

Immediately, we can see computational gains. Because the channels used for transmission are a deterministic function of the channels being probed, the maximization is now over only $O(n^{k+1})$ values of $(u_1, \ldots, u_{k+1})$, a dramatic improvement over n^{2^k}. Fortunately, we can do much better:

Theorem 3. *For the policy defined in Theorem 2, the vector* $(u_1, \ldots, u_{k+1})$ *can be computed in* $O(nk \log(n))$ *time, using Algorithm 2 (see the following section).* $\diamond$

The proofs of the previous theorems are presented in a later section.

D. Algorithms

By extensively manipulating the baseline MaxWeight algorithm, we have arrived at an optimization problem that is amenable to a dynamic programming approach. We will show how to efficiently calculate the solution to (2).

First, it is instructive to rewrite (2) in the form of a nested product:

$$\max_{(u_1,\ldots,u_{k+1})} \mu_{u_1} q_{u_1} + (1-\mu_{u_1})(\mu_{u_2} q_{u_2} + (1-\mu_{u_2})(\ldots + (1-\mu_{u_k})(\mu_{u_{k+1}} q_{u_{k+1}})))$$

Letting $c_{k+1} = \mu_{u_{k+1}} q_{u_{k+1}}$ and $c_i = q_{u_i}\mu_{u_i} + (1-\mu_{u_i})(c_{i+1})$ for $i = 1, \ldots, k$, our objective function reduces to c_1. Furthermore, each c_i only depends on $u_i, \ldots, u_{k+1}$, so if $c(i,j) := \max_{u_i,\ldots,u_{k+1}:u_i=j} c_i$ is the maximum value of a truncation of the objective above, then each $c(i,j)$ can be computed easily, given $c(i+1,\cdot)$:

$$c(i,j) = q_j\mu_j + (1-\mu_j) \max_{x:q_x \leq q_j} c(i+1,x)$$

This gives rise to Algorithm 1 below, in which we iteratively compute all the values of $c(i,j)$, then find $c_1 = \max_j c(1,j)$. The sorting takes $O(n \log n)$ time, the first loop takes $O(n)$ time, the second loop takes $O(kn^2)$ time ($O(n)$ to compute $c(i,j)$), and the $\arg\max$ and final loop both take $O(n)$ time, for a total complexity of $O(kn^2)$.

Although $O(kn^2)$ is much better than the exponential complexity we started with, it can still be improved further with one small optimization. To achieve even lower complexity, modify second loop as follows: Sort $c(i-1,\cdot)$ and iterate through the sorted list to solve $\max_{x<j} c(i-1,x)$, storing your place in the sorted list instead of performing the entire maximization for every j. The modified algorithm can be found in Algorithm 2.

The most complex operation in the modified loop is the sort, which takes $O(n \log n)$ time. Because this loop is performed k times and everything outside of this loop had complexity $O(n \log n)$ before modification, the new algorithmic complexity is $O(kn \log n)$.

Algorithm 1: Polynomial time algorithm to find $\mathbf{u}$

Input: (q, μ, k)
Sort q and μ in descending order of q_i.
for *j = 1 to* n **do**
  $c(k+1, j) := q_j \mu_j$
for *i =* k *to* 1 **do**
  for *j = 1 to* n **do**
    $c(i,j) := q_j\mu_j + (1-\mu_j)\max_{x<j} c(i+1,x)$
    $d(i,j) :=$ the x that maximized the previous quantity
$u_1 := \arg\max_j c(1,j)$
for *i =* 1 *to* k **do**
  $u_{i+1} := d(i, u_i)$
Return $(u_1, \ldots, u_{k+1})$

Algorithm 2: A more efficient algorithm

Input: (q, μ, k)
Sort q and μ in descending order of q_i.
for *j = 1 to* n **do**
  $c(k+1, j) := q_j \mu_j$
for *i =* k *to* 1 **do**
  $s :=$ $[n]$, sorted in descending order of $c(i+1,x)$
  $y := 0$
  for *j = 1 to* n **do**
    while $s[y] < j$ **do**
      y += 1
    $x := s[y]$
    $c(i,j) = q_j\mu_j + (1-\mu_j)c(i+1,x)$
    $d(i,j) = x$
$u_1 := \arg\max_j c(1,j)$
for *i =* 1 *to* k **do**
  $u_{i+1} := d(i, u_i)$
Return $(u_1, \ldots, u_{k+1})$

E. Proofs

To prove Theorem 2, we will start with the policy from Theorem 1, and show it is equivalent to solving some much simpler optimization problems. To start, we reproduce the throughput-optimal probing and transmission channels $\mathbf{u}^*$ and t^* from Theorem 1, after expanding the inner product and using the definition of $v(\mathbf{u}, t)$. $\mathbf{u}^*$ and t^* are the solutions to the following maximization problem:

$$\max_{\mathbf{u}} \max_{t} \sum_{r} p(r|\mathbf{u}) \tilde{\mu}_{t_r} q_{t_r} \tag{3}$$

We present a series of lemmas:

Lemma 4. *Given any series of probing channels* $\mathbf{u}$ *and any probing outcome* r*, an optimal choice* t_r^* *for* t_r *must satisfy:*

$$t_r^* \in \arg\max_{i \in \mathbf{u}} q_i r(i)$$

if $\max_{i\in\mathbf{u}} q_i r(i) \geq \max_{j\notin\mathbf{u}} q_j\mu_j$*. Otherwise, the optimal choice of* t_r *is:*

$$t_r^* \in \arg\max_{j \notin \mathbf{u}} q_j\mu_j$$

The optimal $\mathbf{u}$ *must then maximize:*

$$\max_{\mathbf{u}} \max_{j\notin\mathbf{u}} \sum_r p(r|\mathbf{u}) \max\{\max_{i\in\mathbf{u}} r(i)q_i, q_j\mu_j\} \tag{4}$$

Lemma 5. *For the optimal choice of* $\mathbf{u}$ *and for any choice of* $j \notin \mathbf{u}$*, whenever* $r(i) = 1$ *for some* $i \in \mathbf{u}$*,* $\max_{i\in\mathbf{u}:r(i)=1} q_i \geq q_j\mu_j$*. In other words, we should always transmit on a channel that was sampled to be ON.*

Lemma 6. *If* $q_{u_1} \geq q_{u_2} \geq \ldots \geq q_{u_k}$*, then* $\max_{i\in\mathbf{u}:r(i)=1} q_i = \min_{i\in[k]:r_i=1} q_i$*. Therefore, if we probe the channels in decreasing order of* q_i*, we should always transmit on the first successfully probed channel.*

Proof of Lemma 4: First, we observe that each element of the sum in (3) depends on t only through t_r. Therefore, we can reduce the whole maximization to maximizations over the individual elements of t. This can be written as:

$$\max_{\mathbf{u}} \sum_r p(r|\mathbf{u}) \max_{t_r} q_{t_r}\tilde{\mu}_{t_r} \tag{5}$$

Using the definition of $\tilde{\mu}_{t_r}$, we have:

$$\max_{t_r} q_{t_r}\tilde{\mu}_{t_r} = \max\{\max_{i\in\mathbf{u}} q_i r(i), \max_{j\notin\mathbf{u}} q_j\mu_j\}$$

Therefore, the optimal choices of t_r are as stated in the lemma. We plug the expanded version of $\max_{t_r} q_{t_r}\tilde{\mu}_{t_r}$ into (5):

$$\max_{\mathbf{u}} \sum_r p(r|\mathbf{u}) \max\{\max_{i\in\mathbf{u}} q_i r(i), \max_{j\notin\mathbf{u}} q_j\mu_j\}$$

Now, note that $\max_{j\notin\mathbf{u}} q_j\mu_j$ does not depend on r. Therefore, we can move the maximization over j outside of the sum, obtaining the desired expression:

$$\max_{\mathbf{u}} \max_{j\notin\mathbf{u}} \sum_r p(r|\mathbf{u}) \max\{\max_{i\in\mathbf{u}} r(i)q_i, q_j\mu_j\}$$

Proof of Lemma 5: Consider the sum from (4). We note that this has the form of an expectation, taken over the states of the channels in $\mathbf{u}$ and the channel j (denote the state of channel j by $r(j)$). First, we replace $\max\{\max_{i\in\mathbf{u}} r(i)q_i, q_j\mu_j\}$ with appropriate indicator variables. Let $A_x = \mathbb{I}(r(x) = 1)$, the indicator that channel x is on. Define $B_{\mathbf{u},j}$ to be the indicator that channel j is not chosen for transmission, i.e., $q_j\mu_j < \max_{i\in\mathbf{u}} r(i)$. Note that both are functions of r:

$$B_{\mathbf{u},j} = \mathbb{I}\left(q_j\mu_j < \max_{i\in\mathbf{u}} q_i A_i\right)$$

Now, we can write (4) as:

$$\max_{\mathbf{u},j} \sum_r p(r|\mathbf{u}) \left(B_{\mathbf{u},j} \max_{i\in\mathbf{u}} q_i A_i + (1 - B_{\mathbf{u},j})q_j\mathbb{P}(r(j) = 1)\right)$$

$$= \max_{\mathbf{u},j} \mathbb{E}[B_{\mathbf{u},j} \max_{i\in\mathbf{u}} q_i A_i + (1 - B_{\mu,j})q_j A_j|\mathbf{u}, j]$$
$$= \max_{\mathbf{u},j} \mathbb{E}[h(r|\mathbf{u}, j)|\mathbf{u}, j]$$

Here, we have denoted the inside of the expectation by $h(r|\mu, j)$. Now, consider any choice of $k + 1$ channels $x_1, \ldots, x_{k+1}$ for $\mathbf{u}$ and j, with $q_{x_1} \geq q_{x_2} \geq \ldots \geq q_{x_k}$. If we are to choose which of these $k+1$ channels are assigned to $\mathbf{u}$ and which is assigned to j, we claim that assigning $j = x_{k+1}$, the channel value of q, will maximize the above expectation.

To see this, fix a sample path, consisting of the channel states r for all of the channels in the system. Suppose there exists $k < k + 1$ such that $r(x_i) = 1$. The maximum possible value of $h(r|\mu, j)$ is $\max_{i:r(x_i)=1} q_i$. This can be achieved when $j = x_{k+1}$. If instead, $r(x_i) = 0$ for all $i < k + 1$, then the maximum possible value of $h(r|\mu, j)$ is $q_{k+1}r(k + 1)$, which is achieved with any choice of μ and j. Therefore, $h(r|\mu, j)$ is maximized for any choice of r by taking $j = x_{k+1}$. Taking the expectation over r, we have that the same holds for $\mathbb{E}[h(r|\mathbf{u}, j)|\mathbf{u}, j]$, and therefore for (4).

Finally, because $x_1, \ldots, x_{k+1}$ were arbitrary, for the optimal choice of $\mathbf{u}$ and any $j \notin \mathbf{u}$, $q_i \geq q_j \geq q_j\mu_j$ for all $i \in \mathbf{u}$. Therefore, if $r(i) = 1$ for some $i \in \mathbf{u}$, $\max_{i\in\mathbf{u}:r(i)=1} q_i \geq q_j\mu_j$.

Proof of Lemma 6: Trivial.

Proof of Theorem 2: Let C' be the set of $(\mathbf{u}, j) = (u_1, \ldots, u_{k+1}) \in \binom{n}{k+1}$ such that $q_{u_1} \geq \ldots \geq q_{u_{k+1}}$. Applying Lemmas 4, 5, and 6, we have the following throughput-optimal policy: Probe the channels $u_1, \ldots, u_k$, transmitting on the first successfully probed channel, or on channel $j = u_{k+1}$ if all probes are unsuccessful. $(\mathbf{u}, j)$ are given by a solution to:

$$\max_{(\mathbf{u},j)\in C'} \sum_r p(r|\mathbf{u}) \max\{\max_{i\in\mathbf{u}} r(i)q_i, q_j\mu_j\}$$

By Lemma 5, $q_j\mu_j$ is never chosen over $\max_{i\in\mathbf{u}} q_i r(i)$ unless $r(i) = 0\ \forall i \in \mathbf{u}$. Therefore, we can simplify further:

$$\max_{(\mathbf{u},j)\in C'} \sum_{r\neq 0} \left(p(r|\mathbf{u}) \max_{i\in\mathbf{u}} r(i)q_i\right) + p(r = 0|\mathbf{u})q_j\mu_j$$

We now apply Lemma 6:

$$\max_{(\mathbf{u},j)\in C'} \sum_{r\neq 0} \left(p(r|\mathbf{u}) \min_{i:r_i=1} q_{u_i}\right) + p(r = 0|\mathbf{u})q_j\mu_j$$

Taking advantage of the structure of $p(r|\mathbf{u})$, this becomes:

$$\max_{(\mathbf{u},j)\in C'} \sum_{i=1}^{n} \left(\mathbb{P}(\min_i r_i = 1|\mathbf{u})q_{u_i}\right) + \mathbb{P}(r = 0|\mathbf{u})q_j\mu_j \tag{6}$$

By independence and the definition of r, we have:

$$\mathbb{P}(\min_i r_i = 1|\mathbf{u}) = \mathbb{P}(r_1 = 0, \ldots, r_{i-1} = 0, r_i = 1|\mathbf{u}) = \mu_{u_i} \prod_{m=1}^{i-1} (1 - \mu_{u_m})$$

and

$$\mathbb{P}(r = 0|\mathbf{u}) = \prod_{m=1}^{k} (1 - \mu_{u_m})$$

Therefore, (6) can be expressed as:

$$\max_{(\mathbf{u},j)\in C'} \sum_{i=1}^{k} \left(\mu_{u_i} q_{u_i} \prod_{m=1}^{i-1} (1 - \mu_{u_m}) \right) + q_j \mu_j \prod_{m=1}^{k} (1 - \mu_{u_i})$$

This is further simplified using the notation $j = u_{k+1}$, obtaining the desired result:

$$\max_{(u_1,\ldots,u_{k+1})} \sum_{i=1}^{k+1} \left(\mu_{u_i} q_{u_i} \prod_{m=1}^{i-1} (1 - \mu_{u_m}) \right),$$

F. How much information is needed for throughput optimality?

When the number of probes k is 0, $n-1$, or n, it is known that a scheduling policy does not need knowledge of the channel state μ to achieve throughput optimality. When $k = 0$, any scheme that transmits on a channel with a non-empty queue whenever possible is throughput optimal. If we can probe n or $n-1$ channels, we also do not need to know the values of μ_i. When $k = n$, it is well known that transmitting from the longest active queue is throughput optimal. Similarly, when $k = n - 1$, one can still transmit on the longest active queue by probing all channels except for one with the shortest queue. Knowledge of the state of this remaining channel is not needed, as you would only transmit on this channel if all others were found to be inactive, anyway.

These algorithms for $k \in \{0, n-1, n\}$ all achieve throughput optimality using only a portion of the information about the state of the system. When $k = 0$, we need only identify non-empty queues. When $k \geq n - 1$, we can follow the longest-queue-first heuristic: probe the k channels with the longest queue length, transmit on the first successfully probed channel, or transmit on the unprobed channel with the longest queue length otherwise. In order to study the case where $0 < k < n - 1$, we therefore ask the following three questions:

- Is the information about the emptiness of a queue sufficient for throughput optimality?
- Is the longest-queue-first heuristic sufficient for throughput optimality?
- Is knowledge of μ required for throughput optimality?

We conjecture that the answer to the first two questions is no, and that knowledge of μ is required, in the following sense: no joint probing-scheduling algorithm which makes decisions as a function of only queue lengths can be throughput optimal when $0 < k < n-1$. While we do not yet have a proof of this conjecture, we will provide an example to show that a heuristic combining the longest-queue-first heuristic for probing and the nonempty queue heuristic for transmission is not sufficient.

Consider the following joint probing-scheduling algorithm for $k = 1$ and $n = 3$, which combines the longest-queue-first and nonempty queue heuristics: Probe on the channel with the longest queue and transmit on that channel if the probe is successful. Otherwise, transmit on a random different channel with a non-empty queue (if one exists). We will show that this algorithm is not throughput optimal. To do so, consider the departure rates $\mu = (1, 0.5, 0.5)$ and the arrival rates $\lambda = (1/2 - \epsilon, 1/4 - \epsilon, 1/4 - \epsilon)$, for some $\epsilon > 0$ to be chosen later. Under these arrival and departure rates, the queues can be stabilized by probing channels 2 and 3 with probability 0.5 each, transmitting on the channel probed if the probe is successful, and transmitting on channel 1 otherwise. Intuitively, one should never probe channel 1, because such a probe is always successful and loses the potential information gained by probing channel 2 or 3.

Suppose, with this choice of λ and μ, that the max queue length heuristic spends a fraction p_{ij} of time steps using the strategy of probing channel i and transmitting on j if the probe is unsuccessful. Then, we require the following conditions for stability:

$$(p_{12} + p_{13}) + \frac{1}{2}(p_{21} + p_{31}) > \lambda_1 \quad (7)$$

$$\frac{1}{2}(p_{21} + p_{23}) + \frac{1}{4}(p_{32}) > \lambda_2 \quad (8)$$

$$\frac{1}{2}(p_{31} + p_{32}) + \frac{1}{4}(p_{23}) > \lambda_3 \quad (9)$$

We provide a brief explanation for (7); the reasoning behind (8) and (9) are similar. If channel 1 is probed, it will always be ON and hence transmission will occur on this channel. Thus, $p_{12} + p_{13}$ is the fraction of time spent on transmitting over this channel when strategy 12 or 13 is used. When channels 2 or 3 are probed, then the probe will find the channel being ON with probability half. Thus, the probability of transmitting over channel 1 in this case is $(p_{21} + p_{31})/2$. Combining the above inequalities gives the following:

$$(p_{12} + p_{13}) + p_{21} + p_{31} + \frac{3}{4}(p_{23} + p_{32}) > \lambda_1 + \lambda_2 + \lambda_3$$

Using $\lambda_1 + \lambda_2 + \lambda_3 = 1 - 3\epsilon$ and the fact that $\sum_{ij} p_{ij} = 1$, we have $p_{23} + p_{32} < 12\epsilon$. Combining (8) and (9) also gives:

$$\frac{1}{2}(p_{21} + p_{31}) + \frac{3}{4}(p_{23} + p_{32}) > \frac{1}{2} - 2\epsilon,$$

from which we can conclude that $p_{21} + p_{31} > 1 - 22\epsilon$, so $p_{12} + p_{13} + p_{23} + p_{32} < 22\epsilon$. For some constant c, the system must therefore spend at most 44ϵ fraction of time total in the following states:

1) Queue 1 is longest

2) Queue 2 is longest, but queue 3 is nonempty

3) Queue 3 is longest, but queue 2 is nonempty

Let x be the fraction of time the system spends in a state not listed above, i.e., where either queue 2 is longest but queue 3 is empty, or queue 3 is longest but queue 2 is empty. In such a state, if an arrival occurs to queues 2 and 3, but channels 2 and 3 are OFF, at the next time step, queues 2 and 3 will both be non-empty. Therefore, we have:

$$1 - x \geq x\left(\frac{1}{4} - \epsilon\right)\left(\frac{1}{4} - \epsilon\right)\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)$$

Letting $\epsilon = \frac{1}{10000}$, we get $x \leq 0.985 < 1 - 44\epsilon$, a contradiction. Therefore, the empty queue heuristic we have considered is not throughput optimal.

III. Multiple Frequencies

So far, all of the analysis we have done has been for the special case of one carrier frequency. Now, assume there are m frequencies available for transmission. Each channel, for user i and frequency j, is ON with probability μ_{ij}. We assume that in each time slot, we now simultaneously probe k channels $\mathbf{u}_i$ for each frequency i. Given a result r^i, we then transmit to user t_{r^i}.

We assume independence between the channel states on each frequency band. Then, for long enough queues, we have for every set of probing channels $\{\mathbf{u}_i\}$ that the probability of channel statuses decomposes:

$$p(\{r^i\}|\{\mathbf{u}_i\}) = \prod_i p(r^i|\mathbf{u}_i) \tag{10}$$

So then, the throughput vector $v(\{\mathbf{u}_i\}, \{t\})$ is:

$$\begin{aligned}
v(\{\mathbf{u}_i\}, \{t\}) &= \sum_{\{r\}} \prod_i \left(p(r^i|\mathbf{u}_i)\right) \sum_j (e_{t_{r^j}} \tilde{\mu}_{t_{r^j}}) \\
&= \sum_j \sum_{\{r\}} (e_{t_{r^j}} \tilde{\mu}_{t_{r^j}}) \prod_i \left(p(r^i|\mathbf{u}_i)\right) \\
&= \sum_j \sum_{r^j} (e_{t_{r^j}} \tilde{\mu}_{t_{r^j}}) p(r^j|\mathbf{u}_j) \sum_{\{r_x\}_{x \neq j}} \prod_i p(r^i|\mathbf{u}_i) \\
&= \sum_j \sum_{r^j} (e_{t_{r^j}} \tilde{\mu}_{t_{r^j}} p(r^j|\mathbf{u}_j)) \\
&= \sum_j v(\mathbf{u}_j, t_j)
\end{aligned}$$

The multiple-frequency problem therefore decomposes into the single-frequency problem, and the MaxWeight algorithm for multiple frequencies involves maximizing the sums of the $\langle v(\mathbf{u}_i, t_i), q(k)\rangle$. This can be solved by maximizing each sum individually. In order to do this, we can use Algorithm 2, and run it m times, once for each frequency. The overall complexity is just $kmn\log(n)$. The simple algorithm is reproduced as Algorithm 3, where $SingleFrequency$ is the procedure from Algorithm 2.

Algorithm 3: Simple extension of the single-frequency case

Input: (q, μ, k)
for $f = 1$ *to* m **do**
  $\mathbf{u}(f, \cdot) = SingleFrequency(q, \mu_f, k)$
Return $\mathbf{u}$

A. Algorithmic Improvements to Reduce Delay

Although our algorithm is throughput optimal for multiple frequencies, it may not have optimal delay performance. In particular, because all of the channels for probing and transmission are independently chosen for each frequency, there will often be overlap in the channels chosen. If the queue lengths are small enough, then too many packets may be scheduled for transmission from the same queue, causing other packets to wait unnecessarily. To combat this effect, we introduce two improvements to our algorithm.

This first improvement aims to avoid probing the same channel too many times by introducing a "virtual queue length reduction" after scheduling the probes for each frequency. After the channels $\mathbf{u}_1$ are determined, instead of passing (q, μ_2) to our MaxWeight algorithm for frequency 2, q is first decreased by an amount equal to the expected number of packets transmitted over the first frequency. For example, if $u_{1,1}$ and $u_{1,2}$ are the first two channels probed on frequency 1, then $q_{u_{1,2}}$ is decreased by $\mu_{u_{1,2}}(1 - \mu_{u_{1,1}})$ for the second frequency. This virtual queue length reduction is similar to that used by [10], where it was shown to dramatically reduce delay in the case of perfect channel state knowledge.

Even using virtual queue length reduction, however, we may still schedule a queue on more frequencies than it has packets, wasting throughput. To address this, we introduce "duplicate scheduling protection." After selecting channels for sampling and transmission, and after performing the sampling, we modify the final channels chosen for transmission according to a simple heuristic: instead of transmitting a packet from the queue with the first successfully probed channel, transmit from the first queue with a successful probe that has not already been scheduled on a number of frequencies equal to its number of waiting packets. This simple improvement makes it impossible to schedule a queue on more frequencies than necessary, without affecting throughput optimality.

The complete probing and scheduling algorithm, including virtual queue length reduction and duplicate scheduling protection is presented as Algorithm 4.

This algorithm with modifications has the same throughput optimality guarantees as the original algorithm:

Theorem 7. *Algorithm 4 is throughput optimal and runs in* $O(mnk\log n)$ *time.*

We omit a full proof, but given the throughput optimality of

Algorithm 4: Algorithm with modifications and probing

Input: (q, μ, k)
$q' = \text{copy}(q)$
for $f = 1$ *to* m **do**
  $\mathbf{u}(f, \cdot) = SingleFrequency(q, \mu_f, k)$
  $r = 1$
  for $i = 1$ *to* k **do**
    $q(u(f,i)) = q(u(f,i)) - \mu(f, u(f,i)) \times r$
    $r = r \times (1 - \mu(f, u(f,i)))$
  $q(u(f,k+1)) = q(u(f,k+1)) - r$
for $f = 1$ *to* m **do**
  for $i = 1$ *to* k **do**
    $P(f,i) =$ State of channel i on frequency f
$A(f, \cdot) = 0$
for $f = 1$ *to* m **do**
  for $i = 1$ *to* $k+1$ **do**
    if $P(f,i) = 1, A(u(f,i)) < q'(u(f,i))$ **then**
      $T(u(f,i)) = i$
      $A(u(f,i)) = A(u(f,i)) + 1$
      Break
Return T

Algorithm 4, virtual queue length reduction affects the weights q by a bounded amount, which does not affect throughput optimality of MaxWeight, and duplicate scheduling protection provides a strict increase in throughput.

IV. Simulations

In this section, we conduct simulations to demonstrate the performance of our algorithm and evaluate it given various real-world adjustments and assumptions. In each case, we assume that there are 100 users and 100 frequency bands. The number of arrivals to each queue for each time step follows a Binomial(n, λ_i/n) random variable, where the values λ_i follow a power-law distribution with mean $0.985(m/n)$. The channel statistics μ_{ij} are independently uniformly chosen between 0 and 1, and the simulations are run for 5000 time steps. We have seeded the randomness so that each algorithm tested faces the same arrival and departure rates and the same arrival process.

A. Stability of Queues

First, we investigate the stability of the queues in the system by plotting the average queue length over time. In order to judge the effect of our algorithmic modifications, virtual queue length reduction (VR) and duplicate scheduling protection (DP), we ran our algorithm for $k = 5$ with and without each of these improvements to judge their effect. The average queue lengths are presented in Figure 1.

We note that virtual queue length reduction and duplicate scheduling protection are both effective independently at reducing the average queue length, but even more effective when combined. With the total arrival rate of $0.985m$, the system is very near the boundary of the capacity region, but the system still appears to be stabilized. The negative effect of estimating the departure rates μ_{ij} also appears to be quite brief and almost unnoticeable. Therefore, although we believe that knowledge of μ is required to achieve throughput optimality, simple estimates of μ should suffice.

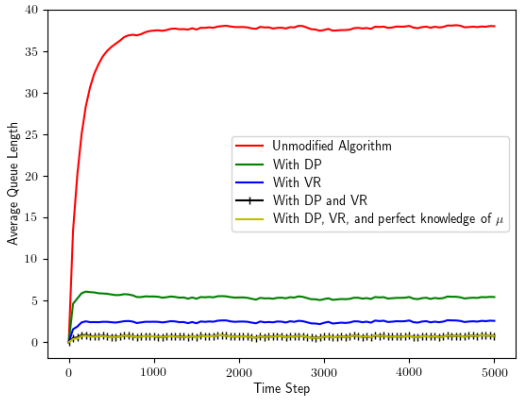

Fig. 1. Average queue length over time, showing the effects of virtual queue length reduction (VR), duplicate scheduling protection (DP), and estimating μ with the empirical average. Results are averaged over 50 trials. Note that VR+DP performs essentially the same with or without perfect knowledge of μ.

Several other algorithms were also tested, most of which failed to stabilize the queue length or did not noticeably impact performance. In particular, every algorithm we tried which did not use knowledge or estimates of the values of μ_{ij} resulted in an average queue length that consistently increased over time. This reinforces our earlier conjecture that knowledge of the μ_{ij} is required to achieve throughput optimality. No other method of estimating the μ_{ij} performed noticeably better than the empirical average, which is not surprising given the extremely small gap in performance between using an estimate of μ_{ij} and using exact values of μ_{ij}. For virtual queue length reduction, we also tried other methods, but reducing by the expected number of transmitted packets (as presented in this paper) was the most successful.

B. Delay performance

We also noted the delay performance, in terms of the empirical probability of packet delay reaching a threshold. We plot this delay violation for each value of the threshold, presenting the results in Figure IV-B. We have omitted the case where we use perfect knowledge of μ, as it is indistinguishable from the delay performance when estimating μ. Again, we find that our algorithmic improvements drastically reduce the delay violation probability.

C. Impact of k

One main parameter in our model is k, the number of channels to probe. The capacity region is a function of k, so increasing

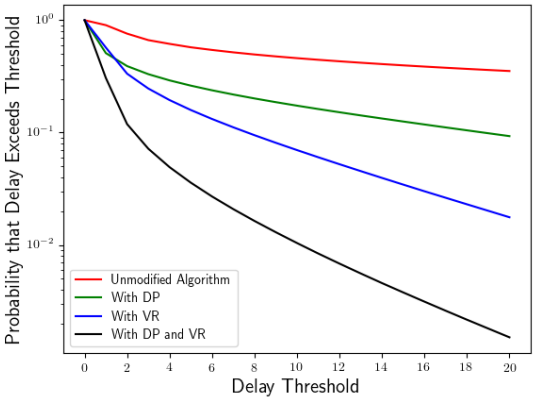

Fig. 2. Empirical probability of delay reaching a threshold, showing the effects of modifications to our algorithm for $k = 5$.

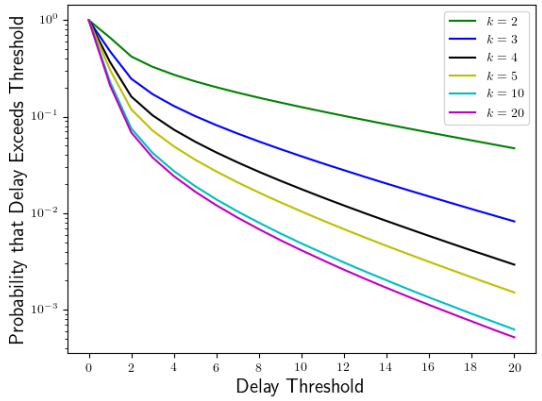

Fig. 3. Empirical probability of delay reaching a threshold, showing the effects of different values of k for our algorithm with VR and DP.

k will expand the set of arrival processes which can be stabilized. However, if the arrival process is fixed, increasing k should also decrease packet delay, as packets can be assigned more efficiently using information obtained from the additional polling. We observe this effect in Figure 3. The case where $k = 1$ is not shown, as the system was empirically determined to be unstable. Starting with $k = 2$, as k increases, the delay performance improves, albeit with diminishing returns. Most of the benefit is already obtained by $k = 10$, suggesting that in order to achieve very low packet delays, only a small number of channel probes are required.

V. Conclusion

In this paper, we considered the combinatorics of joint probing and scheduling algorithms for base stations in cellular networks. We designed a low-complexity algorithm, which we further modified to achieve small packet delays. Our simulations also studied the impact of the number of probes on packet delays and showed that a small number of probes is sufficient to achieve good delay performance.

VI. Acknowledgments

This research was supported by ARO Grant W911NF-16-1-0259 and DTRA Grant HDTRA1-15-1-0003, and by NSF Grants NeTS 1718203, ECCS 1739189, ECCS 16-09370, CMMI 1562276, ECCS-1609202, CNS-1824393, and CNS-1813392.

References

[1] 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*, vol. 37, no. 12, pp. 1936–1948, 1992.

[2] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 2. IEEE, 2004, pp. 1484–1489.

[3] A. L. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Systems*, vol. 50, no. 4, pp. 401–457, 2005.

[4] M. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proc. IEEE INFOCOM*, 2005.

[5] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," in *Proc. IEEE INFOCOM*, 2005.

[6] R. Srikant and L. Ying, *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge University Press, 2013.

[7] S. Kittipiyakul and T. Javidi, "Delay-optimal server allocation in multiqueue multiserver systems with time-varying connectivities," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2319–2333, 2009.

[8] S. Bodas, S. Shakkottai, L. Ying, and R. Srikant, "Low-complexity scheduling algorithms for multichannel downlink wireless networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 5, pp. 1608–1621, 2012.

[9] B. Ji, G. R. Gupta, X. Lin, and N. B. Shroff, "Performance of low-complexity greedy scheduling policies in multi-channel wireless networks: Optimal throughput and near-optimal delay," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 2589–2597.

[10] B. Ji, X. Lin, and N. B. Shroff, "Advances in multi-channel resource allocation: Throughput, delay, and complexity," *Synthesis Lectures on Communication Networks*, vol. 9, no. 1, pp. 1–130, 2016.

[11] M. J. Neely, S. T. Rager, and T. F. La Porta, "Max weight learning algorithms for scheduling in unknown environments," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1179–1191, 2012.

[12] B. Li, B. Ji, and J. Liu, "Efficient and low-overhead uplink scheduling for large-scale wireless internet-of-things," in *2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. IEEE, 2018, pp. 1–8.

[13] M. Ouyang and L. Ying, "Approaching throughput optimality with limited feedback in multichannel wireless downlink networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 6, pp. 1827–1838, 2013.