Dynamic Control of Tunable Sub-optimal Algorithms for Scheduling of Time-varying Wireless Networks

Mahdi Lotfinezhad, Ben Liang, Elvino S. Sousa

***Abstract*—It is well known that the Generalized Max-Weight Matching (GMWM) scheduling policy, and in general throughput-optimal scheduling policies, often require the solution of a complex optimization problem, making their implementation prohibitively difficult in practice. This has motivated many researchers to develop distributed sub-optimal algorithms that approximate the GMWM policy. One major assumption commonly shared in this context is that the time required to find an appropriate schedule vector is negligible compared to the length of a timeslot. This assumption may not be accurate as the time to find schedule vectors usually increases polynomially with the network size. On the other hand, we intuitively expect that for many sub-optimal algorithms, the schedule vector found becomes a better estimate of the one returned by the GMWM policy as more time is given to the algorithm. We thus, in this paper, consider the problem of scheduling from a new perspective through which we carefully incorporate channel variations and time-efficiency of sub-optimal algorithms into the scheduler design. Specifically, we propose a Dynamic Control Policy (DCP) that works on top of a given sub-optimal algorithm, and dynamically but in a large time-scale adjusts the time given to the algorithm according to queue backlog and channel correlations. This policy does not require the knowledge of the structure of the given sub-optimal algorithm, and with low-overhead can be implemented in a distributed manner. Using a novel Lyapunov analysis, we characterize the stability region induced by DCP, and show that our characterization can be tight. We also show that the stability region of DCP is at least as large as the one for any other static policy. Finally, we provide two case studies to gain further intuition into the performance of DCP.**

I. Introduction

The problem of scheduling of wireless networks has been extensively investigated in the literature. A milestone in this context is the seminal work by Tassiulas and Ephremides [1], where the authors characterized the *capacity region* of constrained queueing systems, including wireless networks, and designed a *throughput-optimal* scheduling policy, commonly referred to as GMWM scheduling. Capacity region by definition is the largest region that can be stably supported using any policy, including those with the knowledge of future arrivals and channel states. A throughout-optimal policy is a policy that stabilizes the network for any input rate that is within the capacity region. In general [2][3], GMWM scheduling should maximize the sum of backlog-rate product at each timeslot given channel states, which can be considered as a GMWM problem explaining the name of the policy. This problem has been shown to be, in general, complex and NP-Complete [4][3][5]. Even in those cases where the optimization problem can be solved polynomially, distributed implementation becomes a major obstacle. These issues, naturally, motivated researchers to study and develop suboptimal centralized or distributed algorithms that provide solutions that are a constant factor away from optimality [6][4][5][7][8].

One implicit but major assumption in this context is that the time required to find an appropriate scheduling vector, *search-time*, is negligible compared to the length of a timeslot, or otherwise, during this search-time, channel states remain effectively unchanged. Since most algorithms take polynomial time with the number of users to output a solution [4][5][8], we see that this assumption may not hold for networks with large number of users. In particular, it is possible that once an optimal solution corresponding to a particular channel state is found, due to channel variations, it becomes outdated to the point of being intolerably far away from optimality.

Intuitively, for many suboptimal algorithms, the solution found becomes a better and more *efficient* estimate of the optimal solution as the number of iterations increases or more *time* is given to the algorithm, e.g., see PTAS in [5]. This inspires us to consider this *time-efficiency* correspondence as a classifying tool for sub-optimal algorithms. As mentioned earlier, however, the solution found might become outdated due to channel variations. This poses a challenging problem as how the search-time given to sub-optimal algorithms should be adjusted to ensure an efficient scheduling when channels states are time-varying.

Our work in this paper addresses the above challenge by joint consideration of channel correlation and time-efficiency of sub-optimal algorithms. In particular, we propose a dynamic control policy (DCP) that operates on top of a given sub-optimal algorithm A, where the algorithm is assumed to provide an approximate solution to the GMWM problem. Our proposed policy dynamically tunes the length of scheduling frames as the search-time given to the algorithm A so as to maximize the time average of backlog-rate product, improving the stability region. This policy does not require the knowledge of input rates or the structure of the algorithm A, works with a general class of sub-optimal algorithms, and with low-overhead can be implemented in a distributed manner. We analyze the performance of DCP in terms of its associated stability region, and prove that this policy enables the network to support all input rates that are within θ_∞-scaled version of the capacity region. The scaling factor θ_∞ is a function of the interference model, algorithm A, and channel correlation, and we prove that in general this factor can be tight. We also show that the stability region of DCP is at least as large as the one for any other static scheme that uses a fixed frame-length, or search-time, for scheduling.

As far as we are aware, our study is the first that jointly incorporates the time-efficiency of sub-optimal algorithms and

The authors are with the Department of Electrical and Computer Engineering, University of Toronto. E-mail: {mlotfinezhad, liang}@comm.utoronto.ca, es.sousa@utoronto.ca

channel variations into the scheduler design and stability region analysis. One distinguishing feature of our work, apart form its practical implications, is the use of a Lyapunov drift analysis that is based on a *random* number of steps. Therefore, to establish stability results, we use a method recently developed for Markov chains [9], and modify it such that it is also applicable to our network model.

The rest of this paper is organized as follows. We review the related work in the next section. Network model including details of arrival and channel processes is presented in Section III. Structures of the sub-optimal algorithms and DCP policy are discussed in Section IV. We then provide performance analysis and the related discussion in Section V, followed by two case studies in Section VI. Finally, we conclude the paper in Section VII.

II. Related Work

Previous work on throughput-optimal scheduling includes the studies in [1][10][2]. In particular, in [1], Tassiulas and Ephremides characterized the throughput capacity region for multi-hop wireless networks, and developed GMWM scheduling as a throughput-optimal scheduling policy. This result has been further extended to general network models with ergodic channel and arrival processes [2]. Due to its applicability to general multi-hop networks, GMWM scheduling has been employed as a key component in many cross-layer designs. Examples include rate control [11], energy optimal design [12][13], and congestion control [14][15].

GMWM scheduling despite its optimality, in every timeslot, requires the solution of the GMWM problem, which can be, in general, NP-Complete and Non-Approximable [5]. Thus, many studies has focused on developing sub-optimal constant factor approximations to GMWM scheduling. One interesting study addressing the complexity issue is the work in [16], where sub-optimal algorithms are modeled as randomized algorithms, and it is shown that throughput-optimality can be achieved with linear complexity. This work, however, assumes non-time-varying channels. More recent studies in [3][17] generalize the approach in [16] to time-varying networks, and prove its throughput-optimality. This optimality, as expected, comes at the price of requiring excessively large amount of other valuable resources in the network, which in this case is memory storage. Specifically, the memory requirement in [3][17] increases exponentially with the number of users, making the generalized approach hardly amenable to practical implementation in large networks.

Another example of sub-optimal approximation is the work in [4], where the authors assume that the controller can use only an *imperfect* scheduling component, and as an example they use maximal matching to design a distributed scheduling that is within a constant factor of optimality. This scheduling algorithm under the name of *maximal scheduling* (MM) has been widely studied in the literature [6][5][18][8][19]. In [6][4], it is shown that under simple interference models, MM scheduling can achieve a throughput (or stability region) that is at least half of the throughput achievable by a throughput-optimal algorithm (or the capacity region). Extended versions of these results for more general interference models are presented in [5][8], where in [8] randomized distributed algorithms are proposed for implementing MM scheduling, being a constant factor away from the optimality. All of the mentioned proposals so far either do not consider channel variations, or assume the search-time is relatively small compared to the length of a timeslot.

The closest work to ours in this paper is [7], where based on the linear-complexity algorithm in [16], the impact of channel memory on the stability region of a general class of sub-optimal algorithms is studied. Despite its consideration for channel variations, this work still does not model the search-time, and implicitly assumes it is negligible.

In this paper, we consider the problem of scheduling from a new perspective. We assume a sub-optimal algorithm A is given that can approximate the solution of the GMWM problem, and whose efficiency naturally improves as the search-time increases. We then devise a dynamic control policy which tunes the search-time, as the length of scheduling frames, according to queue backlog levels in the network, and also based on channel correlations. As far as we are aware, our study is the first that explicitly models the time-efficiency of sub-optimal approaches, and uses this concept along with channel correlation in the scheduler design.

III. Network Model

We consider a wireless network with N one-hop source-destination pairs, where each pair represents a data flow[1]. Associated with each data flow, we consider a separate queue, maintained at the source of the flow, that holds packets to be transmitted over a wireless link. Examples of this type of network include downlink or uplink of a cellular or a mesh network.

A. Queueing

We assume the system is time-slotted, and channels hold their state during a timeslot but may change from one timeslot to another. Let the state of the i_{th} link (channel) at time t be $s_i(t)$, and $\mathbf{s}(t)$ be vector of channel states, i.e., $\mathbf{s}(t) = (s_1(t), \ldots, s_N(t))$. Throughout the paper, we use bold face to denote vectors. Let $\mathcal{S}$ represent the set of all possible channel state vectors with finite cardinality $|\mathcal{S}|$. Let $D_i(t)$ denote the rate over the i_{th} link at time t, and $\mathbf{D}(t)$ be the corresponding vector of rates, i.e., $\mathbf{D}(t) = (D_1(t), \ldots, D_N(t))$. In addition, let $I_i(t)$ represent the amount of resource used by the i_{th} link at time t, and $\mathbf{I}(t)$ be the corresponding vector, i.e., $\mathbf{I}(t) = (I_1(t), \cdots, I_N(t))$. The vector $\mathbf{I}(t)$ contains both scheduling and resource usage information, and hereafter, we refer to it simply as the schedule vector. Let $\mathcal{I}$ denote the set containing all possible schedule vectors, with finite cardinality $|\mathcal{I}|$.

Note that the exact specification of the scheduling vector $\mathbf{I}(t)$ is system dependent. For instance, in CDMA systems, it may represent the vector of power levels associated with wireless links; in OFDMA systems, it may represent the number

[1]Extension to multi-hop flows are possible using the methods in [1][2].

of sub-channels allocated to each physical link; in OFDM-CDMA systems, it can be a combination of the two; finally, when interference is modeled as K-hop interference model [5], the vector can be an activation vector representing a sub-graph in the network. Since transmission rates are completely characterized given channel states, the schedule vector, and the interference model, we have

$$\mathbf{D}(t) = \mathbf{D}(\mathbf{s}(t), \mathbf{I}(t)).$$

We emphasize that the exact dependence of $\mathbf{D}(t)$ on $\mathbf{s}(t)$ and $\mathbf{I}(t)$ is determined by the interference model. We assume that transmission rates are bounded, i.e., for all $\mathbf{s} \in \mathcal{S}$ and $\mathbf{I} \in \mathcal{I}$,

$$D_i(\mathbf{s}, \mathbf{I}) < D_{max},\ 1 \leq i \leq N,$$

for some large $D_{max} > 0$.

Let $A_i(t)$ be the number of packets arriving in timeslot t associated with the i_{th} link (or data flow), and $\mathbf{A}(t)$ be the vector of arrivals, i.e., $\mathbf{A}(t) = (A_1(t), \cdots, A_N(t))$. We assume arrivals are i.i.d.[2] with mean vector

$$\mathbb{E}[\mathbf{A}(t)] = \mathbf{a} = (a_1, \ldots, a_N),$$

and bounded above:

$$A_i(t) < A_{max},\ 1 \leq i \leq N,$$

for some large A_{max}.

Finally, let $\mathbf{X(t)} = (X_1(t), \ldots, X_N(t))$ be the vector of queue lengths, where $X_i(t)$ is the queue length associated with the i_{th} link (or data flow). Using the preceding definitions, we see that $\mathbf{X(t)}$ evolves according to the following equation

$$\mathbf{X}(t+1) = \mathbf{X}(t) + \mathbf{A}(t) - \mathbf{D}(t) + \mathbf{U}(t),$$

where $\mathbf{U(t)}$ represents the wasted service vector with non-negative elements; the service is wasted when in a queue the number of packets waiting for transmission is less than the number that can be transmitted, i.e., when $X_i(t) < D_i(t)$.

B. Channel State Process

We assume the channel state process is stationary and ergodic. In particular, for all $\mathbf{s} \in \mathcal{S}$, as $k \to \infty$, we have

$$\frac{1}{k}\sum_{i=0}^{k-1} \mathbf{1}_{\mathbf{s}(t+i)=\mathbf{s}} \to \pi(\mathbf{s}), \quad a.s.,$$

where $\mathbf{1}_{(\cdot)}$ denotes the indicator function associated with a given event, and $\pi(\mathbf{s})$ is the steady state probability of state $\mathbf{s}$. Let $\mathcal{P}_t$ represent the past history of the channel process and be defined by $\mathcal{P}_t = \{\mathbf{s}(i); 0 \leq i \leq t\}$. The above almost surely convergence implies that for any $\epsilon > 0$ and $\zeta > 0$, we can find a sufficiently large $K_{\epsilon,\zeta,t} > 0$ such that [20]

$$P\Big(\sup_{k > K_{\epsilon,\zeta,t}} \big|\frac{1}{k}\sum_{i=0}^{k-1} \mathbf{1}_{\mathbf{s}(t+i)=\mathbf{s}} - \pi(\mathbf{s})\big| > \epsilon \ \ \big|\mathcal{P}_t\Big) < \zeta. \quad (1)$$

We assume that the almost surely convergence is *unform* in the past history and t in the sense that regardless of $\mathcal{P}_t$ and t, there exists a $K_{\epsilon,\zeta}$ such that (1) holds with $K_{\epsilon,\zeta,t} = K_{\epsilon,\zeta}$[3].

C. Capacity Region

In our context, capacity region, denoted by Γ, is defined as the closure of the set of all input rates that can be *stably* supported by the network using any scheduling policy including those that use the knowledge of future arrivals and channel states. In [1][21] and recently under general conditions in [2], it has been shown that the capacity region Γ is given by

$$\Gamma = \sum_{\mathbf{s}\in\mathcal{S}} \pi(\mathbf{s})\ \text{Convex-Hull}\{\mathbf{D}(\mathbf{s}, \mathbf{I}) | \mathbf{I} \in \mathcal{I}\}.$$

IV. Dynamic Control Policy

As mentioned in the introduction, DCP controls and tunes the search-time given to a sub-optimal algorithm to improve the stability region. The considered sub-optimal algorithms are assumed to provide a sub-optimal solution to the GMWM problem. In the following, we first elaborate on the structure of the sub-optimal algorithms, and then, describe the operation of DCP.

A. Sub-optimal Algorithms Approximating GMWM Problem

It is well known that GMWM scheduling is throughput-optimal in that it stabilizes the network for all input rates interior to capacity region Γ. This policy in each timeslot uses the schedule vector $\mathbf{I}^*(t)$ that is argmax to the following GMWM problem:

$$\max \sum_{l=1}^{N} X_l(t) D_l(\mathbf{s}(t), \mathbf{I}), \quad \text{subject to } \mathbf{I} \in \mathcal{I}. \quad (2)$$

However, as mentioned in Section I, this optimization problem can be in general NP-Complete. We therefore assume that there exists an algorithm A that can provide suboptimal solutions to the max-weight problem given in (2). To characterize the structure of algorithm A, let $\mathbf{I}^*(\mathbf{X}, \mathbf{s})$ be the argmax to (2) by setting $\mathbf{X}(t) = \mathbf{X}$ and $\mathbf{s}(t) = \mathbf{s}$. Thus,

$$\mathbf{I}^*(\mathbf{X}, \mathbf{s}) = \underset{\mathbf{I}\in\mathcal{I}}{\operatorname{argmax}}\, \mathbf{XD}(\mathbf{s}, \mathbf{I}),$$

where $\mathbf{XD}(\mathbf{s}, \mathbf{I})$ is the scalar product of the two vectors, and for ease of notation, we have dropped the transpose symbol required for $\mathbf{D}(\mathbf{s}, \mathbf{I})$. In the rest of this paper, we use the same method to show the scalar products. Associated with $\mathbf{I}^*(\mathbf{X}, \mathbf{s})$, let $\mathbf{D}^*(\mathbf{X}, \mathbf{s})$ be defined as

$$\mathbf{D}^*(\mathbf{X}, \mathbf{s}) = \mathbf{D}(\mathbf{s}, \mathbf{I}^*(\mathbf{X}, \mathbf{s})). \quad (3)$$

Thus, $\mathbf{D}^*(\mathbf{X}, \mathbf{s})$ is the optimal rate, in the sense of (2), when the backlog vector is $\mathbf{X}$ and the channel state vector is $\mathbf{s}$.

Let $\mathbf{I}^{(n)}$ be the *output* schedule vector of algorithm A when it is given an amount of time equal to n timeslots, $\mathbf{X}(t) = \mathbf{X}$, and $\mathbf{s}(t) = \mathbf{s}$. We therefore assume that the time given to algorithm A can be programmed or tuned as desired, or simply, the algorithm can continue or iterate towards finding better solutions over time. We assume that $\mathbf{I}^{(n)}$ is in general a random vector with distribution $\mu^{(n)}_{\mathbf{X},\mathbf{s}}$. Since the objective function in (2) is a continuous function of $\mathbf{X}(t)$, we naturally assume that algorithm A characterized by the distribution of $\mathbf{I}^{(n)}$, for all $n \geq 1$, and all values of $\mathbf{X}$ and $\mathbf{s}$, has the following property:

Assumption 1: For all $\mathbf{I} \in \mathcal{I}$, $\mathbf{s} \in \mathcal{S}$, and n, we have that

$$|\mu^{(n)}_{\mathbf{X}_1,\mathbf{s}}(\mathbf{I}^{(n)} = \mathbf{I}) - \mu^{(n)}_{\mathbf{X}_2,\mathbf{s}}(\mathbf{I}^{(n)} = \mathbf{I})| \to 0,$$

as $\mathbf{X}_1 \to \mathbf{X}_2$. In addition, assuming and keeping $\|\mathbf{X}_1 - \mathbf{X}_2\| < C$ for a given $C > 0$, the above convergence also holds when

[2]This assumption is not essential for our results to hold, and is made only to simplify the analysis.

[3]Examples of this channel model include but are not limited to Markov chains.

$\|\mathbf{X}_1\| \to \infty$. Moreover, the convergence becomes equality if $\mathbf{X}_1 = \beta \mathbf{X}_2$, for some $\beta > 0$.

In the following, we discuss concrete models that provide further details on the structure of algorithm A. Note that these models serve only as examples, and our results do not depend on any of these models; what required is only Assumption 1.

The first model arises from the intuition that the distribution $\mu^{(n)}_{\mathbf{X},\mathbf{s}}$ should improve as n increases. More precisely, we can define the sequence $\{\mu^{(n)}_{\mathbf{X},\mathbf{s}},\ n = 1, 2, 3, \cdots\}$ to be an *improving* sequence if for all $n > 1$,

$$\mathbb{E}[\mathbf{XD}(\mathbf{s}, \mathbf{I}^{(n)})] \geq \mathbb{E}[\mathbf{XD}(\mathbf{s}, \mathbf{I}^{(n-1)})] \geq \cdots \geq \mathbb{E}[\mathbf{XD}(\mathbf{s}, \mathbf{I}^{(1)})].$$

The first model uses the above and defines a *natural* algorithm to be the one for which the above inequalities hold for all values of $\mathbf{X}$ and $\mathbf{s}$.

As for the second model, we may have that $\mathbf{I}^{(n)}$ is such that for all $\mathbf{X}$'s and $\mathbf{s}$'s

$$\mathbf{XD}(\mathbf{s}, \mathbf{I}^{(n)}) \geq g(n)\mathbf{XD}(\mathbf{s}, \mathbf{I}^{*}(\mathbf{X}, \mathbf{s})), \tag{4}$$

where the function $g(n)$ is a non-decreasing function of n, and less than or equal to one. For instance, if the optimization problem can be approximated to a convex problem [22], then $g(n) = \xi(1 - \zeta^n)$, where $0 < \xi \leq 1$ and $0 \leq \zeta < 1$. Another possible form for $g(n)$ is

$$\Big(1 - \beta \frac{\ln N}{\ln n}\Big),$$

where β is a positive constant. This form of $g(n)$ may stem from cases where the optimization problem associated with (2) admits Polynomial-Time Approximation Scheme (PTAS) [5].

The last model that we consider is a generalization of the previous model, where we assume that (4) holds with probability $h(n)$ as a non-decreasing function of n. This specification can model algorithms that use randomized methods to solve (2), and without its consideration for the improvement over n, is similar to the ones developed in [16][7].

B. Dynamic Control Policy and Scheduling

The dynamic control policy in this paper interacts with scheduling component, and through some measures, which will be defined later, dynamically tunes the time spent by the scheduler, or more precisely algorithm A, to find a schedule vector. In what follows, we describe the joint operation of DCP and the scheduler.

As DCP operates, the time axis becomes partitioned to a sequence of *scheduling rounds*, where each round might consist of a different number of timeslots. An illustrative example is provided in Fig. 1. Let $\hat{t}_k$ denote the start time of the k_{th} round. Each round begins with a *test interval* followed by an *update interval*. In the beginning of the test interval of each round, a *candidate* value for the number of timeslots given to the algorithm A to solve (2) is selected by DCP. Let $N^r_1(\hat{t}_k)$ denote this candidate value for the k_{th} round, and assume $N^r_1(\hat{t}_k) \in \mathcal{N}_1$, where $\mathcal{N}_1$ has a finite cardinality. In the rest, we use N^r_1 instead of $N^r_1(\hat{t}_k)$ where appropriate. The algorithm that chooses the candidate value might be in general a randomized algorithm. Thus, we use the superscript r to make this point clear. We assume N^r_1 takes an optimal value with probability $\delta > 0$, where optimality will be defined later by (7) and its following discussion.

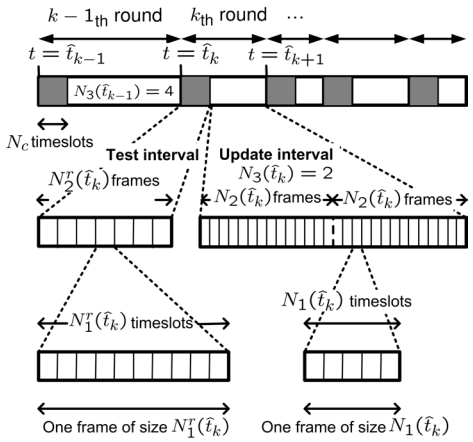

Fig. 1. Illustration of scheduling rounds, test intervals, update intervals, and frames.

We set the length of the test interval to be

$$N^r_1 N^r_2 = N_c = \text{const.},$$

a multiple of N^r_1, where N^r_2 is adjusted accordingly so that the test interval has a fixed length N_c. Therefore, given N^r_1, the test interval becomes partitioned into N^r_2 consecutive frames of N^r_1 timeslots. In the beginning of each frame, e.g., at time t, the current backlog vector $\mathbf{X}(t)$ and channel state vector $\mathbf{s}(t)$ are provided to the algorithm A. The algorithm then spends N^r_1 timeslots to find a schedule vector that is used throughout the next frame of N^r_1 timeslots starting at time $t + N^r_1$. Thus, the schedule vector used in any frame is obtained by using backlog and state vector information at the beginning of its previous frame. Note that depending on the properties of a particular instance of algorithm A, other methods can be considered to use the found vector for the update of scheduling decisions [23], and our results extend to these cases as long as Property 1 and Property 2 in Section V-B hold.

Given N^r_1, DCP evaluates scheduling performance resulting from the chosen value for N^r_1. The performance criterion is the normalized time-average of the backlog-rate (scalar) product. To define the criterion precisely, let $\varphi(\cdot, \cdot, \cdot)$ be defined as

$$\varphi(t, n_1, n_2) = \sum_{j=0}^{n_2-1} \sum_{i=0}^{n_1-1} \frac{\mathbf{X}_{t+jn_1+i}\mathbf{D}_{t+jn_1+i}}{n_1 n_2 \|\mathbf{X}_t\|}.$$

If $\|\mathbf{X}_t\| = 0$, we set $\varphi(t, n_1, n_2) = 0$. Based on the above definition, the criterion associated with the test interval of the k_{th} scheduling round, which is computed by DCP, is denoted by $\varphi^r(\hat{t}_k)$, where

$$\varphi^r(\hat{t}_k) = \varphi(\hat{t}_k, N^r_1(\hat{t}_k), N^r_2(\hat{t}_k)).$$

This quantity is then used to determine the length of frames in the update interval of the k_{th} round.

Update intervals are similar to the test intervals in that they are consisted of a multiple number of fixed-length frames. More precisely, we assume that the update interval in the k_{th} round becomes partitioned into $N_2(\hat{t}_k)N_3(\hat{t}_k)$ consecutive frames of

$N_1(\hat{t}_k)$ timeslots. Integers $N_1(\hat{t}_k)$ and $N_2(\hat{t}_k)$ are such that

$$N_1(\hat{t}_k)N_2(\hat{t}_k) = N_c. \tag{5}$$

Therefore, the length of the k_{th} update interval is $N_3(\hat{t}_k)$ times the length of a test interval. Moreover, we see that $N_1(\hat{t}_k)$ in the k_{th} scheduling round takes the role of $N_1^r(\hat{t}_k)$ in the k_{th} test interval. Assuming the same method is applied to all test and update intervals to use the output of algorithm A, we can properly define $\varphi(\hat{t}_k)$ as

$$\varphi(\hat{t}_k) = \varphi(\hat{t}_k + N_c, N_1(\hat{t}_k), N_2(\hat{t}_k)N_3(\hat{t}_k)).$$

The quantity $\varphi(\hat{t}_k)$ is similar to $\varphi^r(\hat{t}_k)$, and measures the normalized time-average of backlog-rate product in the k_{th} update interval.

DCP , on top of algorithm A, uses $\varphi(\hat{t}_{k-1})$ and $\varphi^r(\hat{t}_k)$ to dynamically control the value of $N_1(\hat{t}_k)$ and $N_3(\hat{t}_k)$ over time. Specifically, in the k_{th} round, at the *end* of its corresponding test interval, the policy chooses either the N_1 used in the previous update interval, $N_1(\hat{t}_{k-1})$, or the newly chosen value of N_1 in the current test interval, $N_1^r(\hat{t}_k)$, according to the following update rule:

$$N_1(\hat{t}_k) = \begin{cases} N_1^r(\hat{t}_k) & \text{if } \varphi^r(\hat{t}_k) > \varphi(\hat{t}_{k-1}) + \alpha \\ N_1(\hat{t}_{k-1}) & \text{otherwise} \end{cases},$$

where α is a suitably small but otherwise an *arbitrary* positive constant. At the same time, the value of $N_3(\hat{t}_k)$, is updated according to the following:

$$N_3(\hat{t}_k) = \begin{cases} \max(1, \frac{N_3(\hat{t}_{k-1})}{2}) & \text{if } \varphi^r(\hat{t}_k) > \varphi(\hat{t}_{k-1}) + \alpha \\ \min(L_1, 2N_3(\hat{t}_{k-1})) & \text{otherwise,} \end{cases}$$

where L_1 is a suitably large but otherwise an *arbitrary* positive constant. Note that $N_2(\hat{t}_k)$ becomes updated such that (5) holds. Once the values of N_1, N_2, and N_3 are updated, in the rest of the scheduling round, which by definition is the update interval, the policy proceeds with computing the time average $\varphi(\hat{t}_k)$. When the k_{th} round finishes, the $k + 1_{\text{th}}$ round starts with a test interval, and DCP proceeds with selecting $N^r(\hat{t}_{k+1})$, and applying the update rule at the end of the $k + 1_{\text{th}}$ test interval. This completes the description of joint operation of DCP and the scheduling component.

Considering the above description, we see that DCP keeps trying new values for N_1. Once a good candidate is found for N_1, the update rule with high probability uses this value for longer periods of time by doubling the length of update intervals. In case the performance in terms of the backlog-rate product degrades, the length of update intervals are halved to expedite trying new values for N_1. Note that α can be arbitrarily small, but should be a positive number. This avoids fluctuations between different values of N_1 performing closely, thus preventing short update intervals. In addition, it limits incorrect favoring towards new values of N_1 in the test intervals, where due to atypical channel conditions, the normalized backlog-rate product deviates from and goes beyond its expected value. Finally, note that L_1 can be arbitrarily large, but should be a finite integer. This assumption is mainly analysis-inspired but is also motivated by the fact that a larger L_1 can lead to a larger delay. Delay analysis is an interesting topic and is left for future research.

V. Performance Analysis

In this section, we evaluate the performance of DCP in terms of its associated stability region. We first introduce several key definitions and functions, and then state the main theorem of the paper.

A. Definitions

Since the backlog vector is non-Markovian, we consider the following definition for the stability of a process.

1) Stability: Suppose there are a bounded closed region $\mathcal{C}$ around the origin, and a real-valued function $F(\cdot) \geq 0$ such the following holds: For any t, and $\sigma_{\mathcal{C}}$ defined by

$$\sigma_{\mathcal{C}} = \inf\{i \geq 0 : \mathbf{X}_{t+i} \in \mathcal{C}\},$$

we have

$$\mathbb{E}[\sigma_{\mathcal{C}}] \leq F(\mathbf{X}(t))\mathbf{1}_{\mathbf{X}(t)\notin\mathcal{C}}.$$

Then, the system is said to be stable.

This definition implies that when $\mathbf{X}(t) \notin \mathcal{C}$, i.e, when $\|\mathbf{X}(t)\|$ is larger than a threshold, the conditional expectation of the time required to return to $\mathcal{C}$, i.e., $\|\mathbf{X}(t)\|$ becomes less than or equal to the threshold, is bounded by a function of only $\mathbf{X}(t)$, uniformly in the past history and t. This definition further implies that if the sequence $\mathbf{X}(t)$ is stable, then [24]

$$\lim_{k\to\infty} \sup_t P(|\mathbf{X}(t)| > k) = 0.$$

2) θ-scaled Region and Maximal Stability: Suppose $0 \leq \theta \leq 1$. A region is called θ-scaled of the region Γ, and denoted by $\theta\Gamma$, if it contains all rates that are θ-scaled of the rates in Γ, i.e.,

$$\theta\Gamma = \{\mathbf{a}_1 : \ \mathbf{a}_1 = \theta\mathbf{a}_2, \ \text{ for some } \mathbf{a}_2 \in \Gamma\}.$$

Further, the θ-scaled region is called *maximally* stable if for all arrival rate vectors interior to $\theta\Gamma$, the system can be stabilized, and for all $\epsilon > 0$ there exists at least one rate vector interior to $(\theta + \epsilon)\Gamma$ that makes the system unstable, both under the same given policy. Thus, maximal stability determines the largest *scaled* version of Γ that can be stably supported under a given policy.

B. Auxiliary Functions and Their Properties

To define the first function, hypothetically suppose for all t, $\mathbf{X}(t) = \mathbf{X}$ for a given $\mathbf{X}$, $\mathbf{X} \neq \mathbf{0}$, and thus, $\mathbf{X}(t)$ does not get updated. In addition, assume that N_1 has a fixed value over time. Considering these assumptions and an update interval of infinite number of frames[4], each consisting of N_1 timeslots, we can see that in the *steady state* the expected normalized backlog-rate product, averaged over one frame, is equal to

$$\phi(\mathbf{X}, N_1) = \mathbb{E}_{\mathbf{s},A}\frac{(\sum_{i=1}^{N_1}\mathbf{X}\mathbf{D}_i)}{N_1\|\mathbf{X}\|}, \tag{6}$$

where $\mathbf{D}_i$ is the rate vector in the i_{th} timeslot of a given frame in the steady state. This expectation is over the steady state distribution of channel process, and possibly over the randomness introduced by the algorithm A.

[4]Here, we assume the channel evolves, and that the algorithm A is used in the same manner as it is used in an ordinary update interval with a finite N_c, as discussed in Section IV-B.

Intuitively, $\phi(\mathbf{X}, N_1)$ states how well a particular choice for N_1 performs, in terms of backlog-rate product, when queue-length changes are ignored. This is exactly what we need to study since the stability region often depends on the behavior of scheduling at large queue-lengths, where in a finite window of time the queue-lengths do not change significantly.

To simplify notation, where appropriate, we use t as the first argument of $\phi(\cdot,\cdot)$; by that we mean[5]

$$\phi(t, N_1) = \phi(\mathbf{X}(t), N_1).$$

Having defined $\phi(\mathbf{X}, N_1)$, we define $\tilde{N}_1(\mathbf{X})$ and $\tilde{\phi}(t)$ by

$$\tilde{N}_1(\mathbf{X}) = \underset{N_1 \in \mathcal{N}_1}{\operatorname{argmax}}\, \phi(\mathbf{X}, N_1), \tag{7}$$

and

$$\tilde{\phi}(t) = \tilde{\phi}(\mathbf{X}(t)) = \phi\Big(\mathbf{X}(t), \tilde{N}_1\big(\mathbf{X}(t)\big)\Big).$$

Finally, for a given $\mathbf{X}$ with $\|\mathbf{X}\| \neq 0$, we define

$$\chi(\mathbf{X}) = \mathbb{E}_{\mathbf{s}}\Big[\frac{\mathbf{X}\mathbf{D}^*(\mathbf{X}, \mathbf{s})}{\|\mathbf{X}\|}\Big],$$

where $\mathbf{D}^*(\mathbf{X}, \mathbf{s})$ is defined in (3), and the expectation is over the *steady state* distribution of the channel process.

According to the above definitions, we see that when variations in the backlog vector are ignored after time t, and N_1 is confined to have a fixed value, $\tilde{N}_1(\mathbf{X}(t))$ becomes the optimal value for N_1 in terms of the normalized backlog-rate product, and $\tilde{\phi}(t)$ represents the corresponding expected value. In particular, note that $\tilde{N}_1(\mathbf{X})$ is a function of $\mathbf{X}$ and may take different values for different $\mathbf{X}$'s. The quantity $\chi(\mathbf{X})$, on the other hand, is the expected normalized backlog-rate product if for all states we could find the optimal schedule vector. This quantity, therefore, can serve as a benchmark to measure performance of sub-optimal approaches.

Note that $\chi(\mathbf{X})$ is continuous function of $\mathbf{X}$ and does not depend on $\|\mathbf{X}\|$. Similarly, by Assumption 1, $\phi(\mathbf{X}, N_1)$ does not depend on $\|\mathbf{X}\|$, and is expected to have the following property.

Property 1: Suppose $\|\mathbf{X}_1 - \mathbf{X}_2\| < C$ for a given $C > 0$. For any given $\epsilon > 0$, there exists a sufficiently large $M > 0$ such that if $\|\mathbf{X}_1\| > M$, then for all $N_1 \in \mathcal{N}_1$

$$|\phi(\mathbf{X}_1, N_1) - \phi(\mathbf{X}_2, N_1)| < \epsilon.$$

This property holds since by Assumption 1, the algorithm A statistically finds similar schedule vectors when two backlog vectors are close and large.

Recall that $\varphi^r(\hat{t}_k)$ is the normalized time-average of backlog-rate product over the k_{th} test interval. If we assume that the backlog vector is kept fixed at $\mathbf{X}(\hat{t}_k)$, by ergodicity of the channel process as explained in Section III-B, we expect $\varphi^r(\hat{t}_k)$ to converge to $\phi(\hat{t}_k, N_1^r(\hat{t}_k))$. Hence, when the number of frames is large, which is the case when N_c is large, $\varphi^r(\hat{t}_k)$ should be close to $\phi(\hat{t}_k, N_1^r(\hat{t}_k))$ with high probability. However, the backlog vector is not fixed and changes over time. But by Assumption 1, algorithm A statistically responds similarly to different backlog vectors if they are close and sufficiently large. This can be exactly our case since arrivals and departures are limited, and thus, for a fixed N_c, the changes in the norm of backlog vector are bounded over one test interval. Therefore, by Assumption 1, if $\|\mathbf{X}(\hat{t}_k)\|$ is sufficiently large, the changes in the backlog have little impact on the distribution of $\varphi^r(\hat{t}_k)$. Applying a similar discussion to $\varphi(\hat{t}_k)$ while noting that the length of update intervals is bounded by $L_1 N_c$, we expect the following property[6].

Property 2: There exist $\varrho_\varphi > 0$ and $\theta_\varphi > 0$ such that for any given $\epsilon > 0$, there exists $M > 0$ such that if $\|\mathbf{X}_{\hat{t}_k}\| > M$, then regardless of k and the past history, up to and including time $\hat{t}_k$, with probability $(1 - \varrho_\varphi)$

$$\big|\varphi^r(\hat{t}_k) - \phi(\hat{t}_k, N_1^r(\hat{t}_k))\big| < \theta_\varphi + \epsilon_1,$$

where ϵ_1 is some positive number less than ϵ. Similarly, regardless of k and the past history, up to and including time $\hat{t}_k + N_c$, with probability $(1 - \varrho_\varphi)$

$$\big|\varphi(\hat{t}_k) - \phi(\hat{t}_k + N_c, N_1(\hat{t}_k))\big| < \theta_\varphi + \epsilon_2,$$

where ϵ_2 is some positive number less than ϵ. Moreover, ϱ_φ and θ_φ approach zero as N_c approaches infinity.

According to the preceding discussion, we can see that θ_φ and ϱ_φ mainly measure how fast the time-averages converge to their expected value, and ϵ models the error due to variations in the backlog vector $\mathbf{X}_{\hat{t}_k+i}$, $1 \leq i \leq \hat{t}_{k+1} - \hat{t}_k - 1$. Thus, as stated above, ϱ_φ and θ_φ can be made arbitrarily small by assuming a sufficiently large value for N_c. In a practical implementation, however, N_c is a limited integer, and therefore, $\theta_\varphi > 0$ and $\varrho_\varphi > 0$.

As the final step towards the main theorem, we define several random variables that are indirectly used in the theorem statement. Specifically, let i_δ be a geometric random variable with success probability $\delta^{'}$, where

$$\delta^{'} = (1 - \varrho_\varphi)^2 \delta,$$

where δ is defined in Section IV-B. In addition, let i_φ be a r.v. with the following distribution.

$$P(i_\varphi = 0) = \varrho_\varphi,$$

and

$$P(i_\varphi = k) = (1 - \varrho_\varphi)^{2k-1}(1 - (1 - \varrho_\varphi)^2),\ k \geq 1.$$

We also define the random sequence $\{N_3^{'}(i),\ i \geq 1\}$ as[7]

$$N_3^{'}(i) = \begin{cases} L_1 & (1 \leq i \leq i_\delta) \vee \\ & \quad (i = i_\delta + i_\varphi + 1) \\ 1 & (i = i_\delta + 1) \wedge (i_\varphi = 1) \\ 2 & (i = i_\delta + 1) \wedge (i_\varphi > 1) \\ \min(\frac{2^i}{2^{i_\delta+2}}, L_1) & (i_\delta + 2 \leq i \leq i_\delta + i_\varphi) \wedge \\ & \quad (i_\varphi > 1) \\ 0 & i > i_\delta + i_\varphi + 1 \end{cases}.$$

Using the above sequence, we define R_∞ as

$$R_\infty = \frac{\mathbb{E}\big[\sum_{i=i_\delta+1}^{i_\delta+i_\varphi} N_3^{'}(i)\big]}{\mathbb{E}\big[\sum_{i=1}^{i_\delta+i_\varphi+1}(1 + N_3^{'}(i))\big]}, \tag{8}$$

which plays a key role in theorem statement and its proof. Note that for a fixed $\delta > 0$, we have

$$\lim_{\varrho_\varphi \to 0} R_\infty = \frac{L_1}{1 + L_1}.$$

[5]By definition of $\phi(\cdot,\cdot)$, here we hypothetically assume the backlog vector $\mathbf{X}(t_1)$ for all times t_1 is equal to $\mathbf{X}(t)$.

[6]We can prove that this property holds as a result of the above discussion, uniform convergence of the channel process, and finiteness of $|\mathcal{I}|$.

[7]Here, $\wedge$ and $\vee$ are the *and* and *or* operators, respectively.

As mentioned earlier, we can make ϱ_φ and θ_φ arbitrarily small by choosing a sufficiently large value for N_c. We are now ready to state the theorem.

C. Main Theorem on Stability of DCP

We have the following theorem:

Theorem 1: Consider a network as described in Section III. For this network, let θ be a constant defined by

$$\theta = R_\infty \inf_{\|\mathbf{X}\|=1} \frac{(\tilde{\phi}(\mathbf{X}) - \alpha - 3\theta_\varphi)}{\chi(\mathbf{X})}.$$

In addition, let θ_∞ be

$$\theta_\infty = \inf_{\|\mathbf{X}\|=1} \frac{\tilde{\phi}(\mathbf{X})}{\chi(\mathbf{X})}.$$

(a) If $6\theta_\varphi < \alpha$ and $2\alpha \leq \inf_{\|\mathbf{X}\|=1} \tilde{\phi}(\mathbf{X})$, then the network is stable under DCP if the mean arrival rate vector, $\mathbf{a}$, lies strictly inside the region $\theta\Gamma$.

(b) For any input rate strictly inside $\theta_\infty\Gamma$, there exist a sufficiently small value for α, and sufficiently large values for L_1 and N_c such that the network becomes stabilized under DCP. In other words, we can expand the sufficient stability region $\theta\Gamma$ arbitrarily close to $\theta_\infty\Gamma$ by choosing appropriate values for for α, L_1, and N_c.

(c) There exist instances of networks, as described in Section III, for which their associated region $\theta_\infty\Gamma$ is maximally stable under DCP.

Proof: The proof is provided in the Appendix. ■

D. Discussion

1) Intuitive Explanation of θ: Theorem 1 states that all input rates interior to $\theta\Gamma$ can be stably supported under DCP. In particular, it implicitly quantifies θ as a function of the sub-optimality of algorithm A and channel state correlation. Clearly, the value of θ is not fixed, and can vary from a particular network setup to another. As expected, for a fixed $\mathbf{X}$, as algorithm A finds better schedule vectors in shorter times, and as the channel states become more correlated, $\tilde{\phi}(\mathbf{X})$ gets closer to $\chi(\mathbf{X})$, and θ gets closer to one, expanding the region $\theta\Gamma$ to the capacity region Γ.

In addition, Theorem 1 shows how the stability region is directly affected by the choices for α and L_1, and the values for θ_φ and ϱ_φ. The impact of α on θ could be predicted by noting that the update rule uses N_1^r in an update interval only when the normalized average backlog-rate product increases at least by α. Thus, we expect to see a decrease of the type $\frac{\alpha}{\chi(\mathbf{X})}$ in the stability region scaling. The effect of θ_φ and ϱ_φ is less obvious, but can be roughly explained as follows. Suppose at the k_{th} round the optimal N_1 is selected, i.e., $N_1^r(\hat{t}_k) = \tilde{N}_1(\hat{t}_k)$. In this case, to have a proper comparison, $\varphi^r(\hat{t}_k)$ and $\varphi(\hat{t}_{k-1})$ should satisfy their corresponding inequalities in Property 2. Moreover, to make sure that $N_1^r(\hat{t}_k)$ or a near optimal N_1 is used in the l_{th} round after the k_{th}, we at least require $\varphi^r(\hat{t}_l)$ satisfy its corresponding inequality in Property 2. Therefore, there are at least three inequalities of the form in Property 2 that should be satisfied, which results in the term $3\theta_\varphi$ in the expression for θ.

The factor R_∞ in a sense measures a lower-bound for the fraction of time where *near* optimal values for N_1 are used by DCP. To better understand R_∞, suppose ϱ_φ is small, and the backlog vector is large. Once the optimal value for N_1 is found in a round, as long as the inequalities in Property 2 hold for the subsequent rounds, N_1 gets updated for only a few times. By the update rule, this means that N_3 gets doubled in most of the rounds, and is likely equal to L_1. Thus, the update intervals constitute $\frac{L_1}{1+L_1}$ fraction of time. At the same time, in these intervals, near optimal values for N_1 are being used. Thus, we expect to see $\frac{L_1}{1+L_1}$ as a multiplicative factor in θ.

The above discussion and Theorem 1 also state that DCP successfully adapts N_1 in order to keep $\varphi(\hat{t}_k + N_c, N_1(\hat{t}_k))$ close to $\tilde{\phi}(\mathbf{X}(\hat{t}_k + N_c))$[8]. Note that for a given $\mathbf{X}$ finding $\tilde{N}_1(\mathbf{X})$, or equivalently, $\tilde{\phi}(\mathbf{X})$, in general, is a difficult problem. Specifically, it requires the exact knowledge of channel state and arrival process statistics, and the structure of algorithm A. Even when this knowledge is available, as the number of users increases, finding $\tilde{N}_1(\mathbf{X})$ demands computation over a larger number of dimensions, which becomes exponentially complex. Hence, we see that DCP dynamically solves a difficult optimization problem, without requiring the knowledge of input rates or the structure of algorithm A[9].

2) Comparison with Static Policies, Minmax v.s. Maxmin: Part (b) of the theorem gives the region $\theta_\infty\Gamma$ as the fundamental lower-bound on the limiting performance of DCP. It also implicitly states that this lower-bound depends on the solution to a minmax problem. To see this, recall that by definition $\tilde{\phi}(\mathbf{X})$ is the maximum of $\phi(\mathbf{X}, N_1)$ over all choices for N_1. Thus, we have that

$$\theta_\infty = \inf_{\|\mathbf{X}\|=1} \max_{N_1 \in \mathcal{N}_1} \frac{\phi(\mathbf{X}, N_1)}{\chi(\mathbf{X})}.$$

Now, consider a *static* policy that assumes a fixed value for N_1. This policy partitions the time axis into a set of frames each consisting of N_1 timeslots, with the i_{th} frame starting at time $(i-1)N_1$. The static policy, in the beginning of each frame, e.g., the i_{th} frame, provides algorithm A with vectors $\mathbf{X}((i-1)N_1)$ and $\mathbf{s}((i-1)N_1)$. Algorithm A uses these vectors as input, and after spending N_1 timeslots, returns a schedule vector as the output. This output vector is then used to schedule users in the next following frame.

It is not difficult to show that the above static policy stabilizes the network for all rates interior to $\theta^s_{N_1}\Gamma$, where

$$\theta^s_{N_1} = \inf_{\|\mathbf{X}\|=1} \frac{\phi(\mathbf{X}, N_1)}{\chi(\mathbf{X})}.$$

Thus, the best static policy, in terms of the region $\theta^s_{N_1}\Gamma$, is the one that maximizes $\theta^s_{N_1}$. Let θ^s_o be the maximum value. We have that

$$\theta^s_o = \max_{N_1 \in \mathcal{N}_1} \inf_{\|\mathbf{X}\|=1} \frac{\phi(\mathbf{X}, N_1)}{\chi(\mathbf{X})}.$$

Therefore, the best static policy corresponds to a maxmin problem. Considering the definition of θ_∞ and θ^s_o, and that

[8]This statement is in fact a direct result of Lemma 4 in [23], which is omitted due to page limitation.

[9]DCP also does not require the exact knowledge of channel state statistics. However, a practical implementation of DCP requires N_c to be related to the convergence-rate of channel process to its steady state.

the minmax of a function is always larger than or equal to the maxmin, we have that $\theta_o^s\Gamma \subseteq \theta_\infty\Gamma$. More generally, using the above definitions and a simple drift analysis, we can show that the stability region of static policies is not larger than the limiting stability region of DCP.

3) Tightness of θ_∞ and θ_o^s: Note that parts (a) and (b) of the theorem do not exclude the possibility of networks being stable under DCP for rates outside of $\theta\Gamma$ or $\theta_\infty\Gamma$. Part (c) of the theorem, on the other hand, compliments parts (a) and (b), and shows that for some networks the region $\theta_\infty\Gamma$ is indeed the largest *scaled* version of Γ that can be stably supported under DCP. This for instance may happen when the channel state vector is statistically symmetric with respect to users [23]. The same discussion also applies to $\theta_{N_1}^s$ and the stability region of static policies. We therefore have θ_∞ and θ_o^s both as tight measures, stating that for some networks, including the ones in the next section, DCP can increase throughput efficiency of static policies by a factor of $\frac{\theta_\infty - \theta_o^s}{\theta_o^s}$. Analytical quantification of this gain is an interesting problem and is left for future research.

4) Delay and Distributed Implementation: Note that getting close to the boundary of $\theta_\infty\Gamma$ increases delay. This follows from part (b) of the theorem stating that for input rates close to the boundary, L_1 and N_c should be large. These choices, as expected, increase the length of test and update intervals, which can potentially be large intervals of sub-optimal transmissions in terms of the value used for N_1. This in turn makes data wait in queues before transmission, thus increasing the delay.

Assuming algorithm A is decentralized [6][4][5][8], DCP can be implemented in a distributed manner with low overhead. This is possible since consistent implementation of DCP in all nodes requires updates of only queue backlog and nodes' time-average of backlog-rate product, and such updates are needed only over *long* time intervals. Further discussion of this topic is provided in [23].

VI. Case Studies

In this section, we present two examples that provide further insight into our analytical results and the performance of DCP. To be able to compare the simulation results with analytical ones, we consider a small network consisting of two data flows in the downlink of a wireless LAN or a cellular network. In this case, $\mathbf{s}(t)$ is the vector of channel gains, and we assume the schedule vector is the power allocation vector, i.e., $\mathbf{I} = \mathbf{P} = (p_1, p_2)$, with constraint

$$p_1 + p_2 = P_t,$$

where P_t is total power budget. Assuming super-position coding is used in the downlink, if $s_1(t) < s_2(t)$, then [25]

$$D_1(\mathbf{s}(t), \mathbf{P}) = \log\Big(1 + \frac{p_1|s_1|^2}{p_2|s_1|^2 + n_0}\Big),$$

and

$$D_2(\mathbf{s}(t), \mathbf{P}) = \log\Big(1 + \frac{p_2|s_2|^2}{n_0}\Big).$$

If $s_1(t) \geq s_2(t)$, we obtain similar expressions for user rates by swapping the role of one user for another.

For illustration purposes, we assume that algorithm A in every step, i.e., during each timeslot, reduces the gap to the optimal backlog-rate product. Specifically, if the initial gap corresponding to the initial power vector $\mathbf{I}^{(0)}$, assumed to be chosen randomly, is Δ_0, then after i steps the gap is decreased to Δ_i, where

$$\begin{aligned}\Delta_i &= \mathbf{XD}^*(\mathbf{X}, \mathbf{s}) - \mathbf{XD}(\mathbf{s}, \mathbf{I}^{(n)}) \\ &= \frac{1}{\beta^i}\big(\mathbf{XD}^*(\mathbf{X}, \mathbf{s}) - \mathbf{XD}(\mathbf{s}, \mathbf{I}^{(0)})\big) = \frac{\Delta_0}{\beta^i},\end{aligned}$$

where $\beta > 1$. This case corresponds to $g(n) = (1 - \zeta^i)$ with $\zeta = \frac{1}{\beta}$, where $g(n)$ is introduced in Section IV-A.

Having specified rates and algorithm A, as the first example, we assume that the channel state is Markovian with two possible state vectors, namely, $\mathbf{s}_1 = (1, 5)$ and $\mathbf{s}_2 = (5, 1)$, where the channel in each transition takes a different state with probability $p_t = 0.3$. For this case, we set $\alpha = 0.06$, $N_c = 12000$, $L_1 = 32$, $\beta = 1.7$, $\mathcal{N}_1 = \{N_1 : 1 \leq N_1 \leq 6\}$, $n_0 = 10$, and $p_t = 50$. To study the stability region, we consider the rate vector $\mathbf{a} = (2.4181, 2.4181)$ which belongs to the boundary of Γ corresponding to this example. We then assume the arrival vector is $\gamma\mathbf{a}$, where γ is the load factor, and varies from 0.84 to 0.92. Fig. 2 depicts the resulting average queue sizes. For loads larger than 0.93, the queue sizes increase with time implying network instability. The range selected for γ is motivated by noting that $\theta_\infty = 0.9447$, which is computed numerically. Considering the growth of average queue sizes in Fig. 2, we therefore see that for this example θ_∞ is indeed an upper bound for capacity region scaling. In fact, part (c) of Theorem 1 applies to this example, and any rate of the form $(\theta_\infty + \epsilon)\mathbf{a}$, $\epsilon > 0$, makes the network unstable.

As for the second example, we increase the number of states

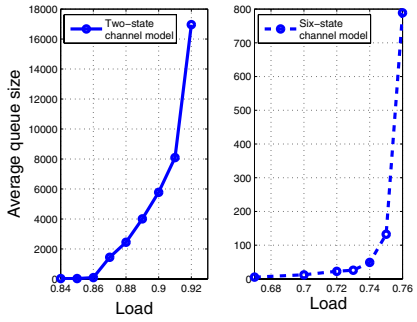

Fig. 2. Average queue size as a function of load factor.

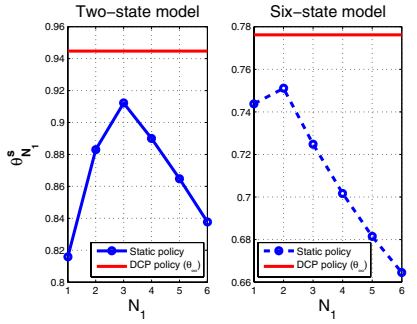

Fig. 3. Comparison of capacity region scaling for DCP and static policies.

to six corresponding to the following state vectors:

$$\mathbf{s}_1 = (1,5), \quad \mathbf{s}_2 = (5,1),$$
$$\mathbf{s}_3 = (1,2), \quad \mathbf{s}_4 = (2,1),$$
$$\mathbf{s}_5 = (2,5), \quad \mathbf{s}_6 = (5,2),$$

and having the following symmetric transition matrix:

$$\mathbf{T_m} = \begin{pmatrix} 0.3 & 0.1 & 0.2 & 0.1 & 0.2 & 0.1 \\ 0.1 & 0.3 & 0.1 & 0.2 & 0.1 & 0.2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ 0.1 & 0.2 & 0.1 & 0.2 & 0.1 & 0.3 \end{pmatrix}. \tag{9}$$

For this case, we keep the same N_c, L_1, and $\mathcal{N}_1$, but assume $\alpha = 0.02$, $\beta = 1.5$, $n_0 = 50$, and $p_t = 10$. Similar to the previous example, to vary arrival rate vector, we consider the rate vector $\mathbf{a} = (0.6952, 0.6952)$ which belongs to the boundary of Γ associated with this example. Then, the arrival vector is assumed to be $\gamma\mathbf{a}$, where the load factor γ varies from 0.67 to 0.76. The resulting average queue sizes are also shown in Fig. 2. In this case, for load factors larger than 0.76, the queue sizes increase with time, suggesting network instability. This result is consistent with our analytical results since the numerically computed value of θ_∞ is 0.7762. Note that part (c) of Theorem 1 also applies to this example, and any rate of the form $(\theta_\infty + \epsilon)\mathbf{a}$, $\epsilon > 0$, makes the network unstable.

Finally, in Fig. 3, for the two examples, we have shown $\theta^s_{N_1}$ [10] as a function of N_1, and also shown the value of θ_∞ for DCP. As expected and the figure suggests, since DCP adapts N_1 according to queue backlog, it outperforms the best static policy. We also see that the optimal stationary policy for the first example is the one with $N_1 = 3$ and $\theta^s_o = 0.9122$, and for the second example is the one with $N_1 = 2$ and $\theta^s_o = 0.7511$. Note that characterization of the best static policy requires computation of $\tilde{\phi}(\mathbf{X})$, which, as briefly discussed in Section V-D1, can be computationally intensive. From the figure, we also observe that the performance of a suboptimal static policy can be substantially less than DCP if the static policy does not assume a proper value for N_1.

VII. Conclusion

In this paper, we have considered the problem of scheduling in time-varying networks from a new perspective. Specifically, in contrast to previous research which assumes the search-time to find schedule vectors is negligible, we have considered this time, based on which we modeled the time-efficiency of sub-optimal algorithms. Inspired by this modeling, we have proposed a dynamic control policy that dynamically but in a large time-scale tunes the time given to an available sub-optimal algorithm according to queue backlog and channel correlation. Remarkably, this policy does not require knowledge of input rates or the structure of available sub-optimal algorithms, nor it requires exact statistics of channel process. We have shown that this policy can be implemented in a distributed manner with low overhead. In addition, we have analyzed the stability region of the proposed policy and showed that it performs at least as efficient as any other static policy. We believe that study and design of similar policies opens a new dimension in the design of scheduling policies, and in parallel to the efforts to improve the performance of sub-optimal algorithms, can help boost the throughput performance to the capacity limit.

[10]Note that for both examples, the region $\theta^s_{N_1}\Gamma$ is either maximally stable or very close to the maximal stable region of the form $(\theta^s_{N_1} + \epsilon)\Gamma$, where $\epsilon < 10^{-5}$. This can be easily verified through numerical values for $\tilde{\phi}(\mathbf{X})$, and the proof of part (c) of Theorem 1.

Appendix

Proof of Theorem 1: For the lack of space, we only provide the proof of part (a) of the theorem, with some intermediate steps omitted. The complete proof of the theorem is provided in [23].

The proof of part (a) consists of two main parts. First, using several lemmas, we obtain a negative drift with a random number of steps. In the second part, we use the negative drift analysis to show that the return time to a bounded region has a finite expected value, and conforms to the properties required for network stability, according to the definition given in Section V-A1.

We start by noting that $\theta \leq 1$, and since $\mathbf{a}$ is strictly inside $\theta\Gamma$, there must be some non-negative constants $\beta_{\mathbf{s},\mathbf{I}}$ with the property that for all $\mathbf{s} \in \mathcal{S}$

$$\sum_{\mathbf{I}\in\mathcal{I}} \beta_{\mathbf{s},\mathbf{I}} < \theta \leq 1, \tag{10}$$

such that

$$\mathbf{a} = \sum_{\mathbf{s}\in\mathcal{S}} \pi(\mathbf{s}) \sum_{\mathbf{I}\in\mathcal{I}} \beta_{\mathbf{s},\mathbf{I}} \mathbf{D}_{\mathbf{s},\mathbf{I}}. \tag{11}$$

Considering (10), we can define positive ξ' as

$$\xi' = \theta - \max_{\mathbf{s}\in\mathcal{S}} \sum_{\mathbf{I}\in\mathcal{I}} \beta_{\mathbf{s},\mathbf{I}}.$$

Since $\xi' > 0$, by the definition of θ, for $\|\mathbf{X}_t\| \neq 0$, we have that

$$\frac{R_\infty(\tilde{\phi}(t) - \alpha - 3\theta_\varphi)}{\chi(\mathbf{X}_t)} - \max_{\mathbf{s}\in\mathcal{S}} \sum_{\mathbf{I}\in\mathcal{I}} \beta_{\mathbf{s},\mathbf{I}} \ > \xi' > 0. \tag{12}$$

To proceed with the proof, associated with a given time t, we define a sequence of random variables $\{\tau_i\}_{i=-1}^{\infty}$, where τ_{-1} and τ_0 denote the number of timeslots to the last timeslot of the previous and the current scheduling round, respectively, and τ_i, $i \geq 1$, is the number of timeslots to the last timeslot of the i_{th} subsequent scheduling round. Let $\mathcal{H}_t$ denote the past history of the system up to and including time t. Thus, given $\mathcal{H}_t$, the value of $\mathbf{X}_t$ is known. Let $f(\cdot)$ be defined as

$$f(\mathbf{X}) = \|\mathbf{X}\|^2,$$

Considering a τ_K+1-step drift with function $f(\cdot)$, we can write

$$\begin{aligned}\Delta(\tau_K+1) &= \mathbb{E}[f(\mathbf{X}_{t+\tau_K+1}) - f(\mathbf{X}_t)|\mathcal{H}_t] \\ &= \mathbb{E}[\sum_{k=0}^{\tau_K} f(\mathbf{X}_{t+k+1}) - f(\mathbf{X}_{t+k})|\mathcal{H}_t] \\ &= \mathbb{E}[\sum_{k=0}^{\tau_K} (\mathbf{X}_{t+k+1} + \mathbf{X}_{t+k})(\mathbf{X}_{t+k+1} - \mathbf{X}_{t+k})|\mathcal{H}_t].\end{aligned}$$

Using the fact that arrivals and departures are bounded, after

performing some preliminary steps, we can show that

$$\Delta(\tau_K+1) \leq \mathbb{E}\Big[(\tau_K+1)C_1+(\tau_K+1)^2C_2 + 2\sum_{k=0}^{\tau_K}(\mathbf{X}_t\mathbf{A}_{t+k}-\mathbf{X}_{t+k}\mathbf{D}_{t+k})\ \big|\mathcal{H}_t\Big],$$

for appropriate constants C_1 and C_2. Since $\mathbf{X}_{t+k}\mathbf{D}_{t+k}\geq 0$, we have

$$\begin{aligned}\Delta(\tau_K+1) \leq \mathbb{E}\Big[&(\tau_K+1)C_1+(\tau_K+1)^2C_2 \\ &+2\sum_{k=0}^{\tau_K}(\mathbf{X}_t\mathbf{A}_{t+k}-\mathbf{X}_t\mathbf{a}) \\ &+2\sum_{k=0}^{\tau_K}(\mathbf{X}_t\mathbf{a}-\mathbf{X}_{t+k}\mathbf{D}^*_{t+k}) \\ &+2\sum_{k=0}^{\tau_K}\mathbf{X}_{t+k}\mathbf{D}^*_{t+k} \\ &-2\sum_{k=\tau_0+1}^{\tau_K}\mathbf{X}_{t+k}\mathbf{D}_{t+k}\ \big|\mathcal{H}_t\Big],\end{aligned}$$

where $\mathbf{D}^*_{t+k}=\mathbf{D}^*(\mathbf{X}(t+k),\mathbf{s}(t+k))$. In the following, we derive an upper bound for $\Delta(\tau_K+1)$.

As mentioned in Section III-A, arrivals are i.i.d with mean vector $\mathbf{a}$. We can therefore apply the same method used to prove Lemma 1 in [23] to obtain

$$\mathbb{E}\big[\|\sum_{k=0}^{\tau_K}\mathbf{A}_{t+k}-(\tau_K+1)\mathbf{a}\|\ \big|\mathcal{H}_t\big]\leq\epsilon\mathbb{E}\big[(\tau_K+1)|\mathcal{H}_t\big],$$

where $\epsilon>0$, and can be made arbitrarily small by choosing a sufficiently large K.

Using the above inequality, Lemma 2, Lemma 3, and Lemma 4, as provided in [23], all with the same choice for ϵ, we can show that

$$\Delta(\tau_K+1)\leq\mathbb{E}\Big[(\tau_K+1)\|\mathbf{X}_t\|\chi(\mathbf{X}_t) \Big(\epsilon_1+2\big(\max_{\mathbf{s}\in\mathcal{S}}\sum_{\mathbf{I}\in\mathcal{I}}\beta_{\mathbf{s},\mathbf{I}}-\frac{R_\infty(\tilde{\phi}(t)-\alpha-3\theta_\varphi)}{\chi(\mathbf{X}_t)}\ \big)\Big)\big|\mathcal{H}_t\Big], \quad (13)$$

where

$$\epsilon_1=\frac{1}{\chi(\mathbf{X}_t)}\Big(\frac{C_1}{\|\mathbf{X}_t\|}+\frac{C_2(\tau_K+1)}{\|\mathbf{X}_t\|}+8\epsilon\Big). \quad (14)$$

Note that according to the lemmas, ϵ can take any given positive real number if K and $\|\mathbf{X}_t\|$ are sufficiently large.

We can show that $\epsilon_1<\xi'$ if K is chosen sufficiently large, and $\|\mathbf{X}_t\|>M_K$, for an appropriately large M_K [23]. Using these assumptions, (13), and (12), we have that

$$\Delta(\tau_K+1)<-\mathbb{E}\Big[\xi'\|\mathbf{X}_t\|(\tau_K+1)\chi(\mathbf{X}_t)\ \big|\mathcal{H}_t\Big].$$

This inequality and that for $\|\mathbf{X}\|\neq 0$, $\chi(\mathbf{X})\geq\frac{\upsilon}{\sqrt{N}}$, for some positive $\upsilon>0$ [23], further imply that

$$\Delta(\tau_K+1)<-\mathbb{E}\Big[\xi(\tau_K+1)\|\mathbf{X}_t\|\big|\mathcal{H}_t\Big], \quad (15)$$

where $\xi=\frac{\upsilon}{\sqrt{N}}\xi'>0$. We, therefore, have obtained the negative drift expression, completing the first part of the proof.

Note that in above τ_K is a random variable, and in fact, is a stopping time with respect to the filtration $\mathcal{H}=\{\mathcal{H}_t\}_{t=0}^{\infty}$. This means that we have obtained a drift expression that is based on a random number of steps. Proofs of stability in the literature, however, are often based on a negative drift with a fixed number of steps. This contrast has motivated us to adopt an interesting method recently developed in [9]. This method is general since it can be applied in both cases, and also leads to an intuitive notation of stability. However, it has been originally developed for Markov chains. Therefore, as well as using less technical notations, in what follows, we apply minor modifications to the method so that it is appropriate in our context.

We now, in the second part of the proof, use the negative drift, and prove that the expected value of the return time to some bounded region is finite in a manner that renders network stable. Let $\mathcal{C}$ denote the bounded region, and be defined as

$$\mathcal{C}=\{X\in\mathbb{R}^N,\|X\|\leq M_K\}.$$

Associated with $\mathcal{C}$, we define $\sigma_{\mathcal{C}}$ to be the number timeslots after which the process $\{\mathbf{X}_{t+i}\}_{i=0}^{\infty}$ *enters* $\mathcal{C}$, i.e.,

$$\sigma_{\mathcal{C}}=\inf\{i\geq 0:\mathbf{X}_{t+i}\in\mathcal{C}\}.$$

Similarly, we let $\tau_{\mathcal{C}}$ be

$$\tau_{\mathcal{C}}=\inf\{i\geq 1:\mathbf{X}_{t+i}\in\mathcal{C}\}.$$

Therefore, $\tau_{\mathcal{C}}$, in contrast $\sigma_{\mathcal{C}}$, characterizes the first time that the process $\{\mathbf{X}_{t+i}\}_{i=1}^{\infty}$ *returns* to $\mathcal{C}$.

Back to the drift expression in (15), let η be a random variable defined by

$$\eta=\xi(\tau_K+1)\|\mathbf{X}_t\|.$$

We obtain, for K sufficiently large,

$$\mathbb{E}[f(\mathbf{X}_{t+\tau_K+1})+\eta|\mathcal{H}_t]\leq f(\mathbf{X}_t), \quad (16)$$

provided that $\|\mathbf{X}_t\|>M_K$. Let $\eta_0=\eta$, and $\tau_{K,0}=\tau_K$, where η and τ_K are random variables defined by considering time t. We now consider time $t_K^{(1)}=t+\tau_{K,0}+1$. For this particular time, we can define another pair $\tau_{K,1}$ and η_1 such that if $\|\mathbf{X}_{t_K^{(1)}}\|>M_K$, then

$$\mathbb{E}[f(\mathbf{X}_{t_K^{(1)}+\tau_{K,1}+1})+\eta_1|\mathcal{H}_{t_K^{(1)}}]\leq f(\mathbf{X}_{t_K^{(1)}}),$$

where $\tau_{K,1}$ is the number of timeslots from time $t_K^{(1)}$ to the last timeslot of the K_{th} subsequent scheduling round, and

$$\eta_1=\xi(\tau_{K,1}+1)\|\mathbf{X}_{t_K^{(1)}}\|.$$

Note that the definition of $\tau_{K,1}$ and η_1 is independent of whether the previous inequality holds.

We can continue this process by considering the drift criteria for time $t_K^{(i)}=t_K^{(i-1)}+\tau_{K,i-1}+1$, and defining random variables $\tau_{K,i}$ and η_i. The random variables $\tau_{K,i}$ and η_i have a similar definition as $\tau_{K,1}$ and η_1, respectively, except that they are associated with time $t_K^{(i)}$. Using these definitions, we can define $t_K^{(i)}$ more precisely by

$$t_K^{(0)}=t,$$

$$t_K^{(i)}=t_K^{(i-1)}+(\tau_{K,i-1}+1)=t+\sum_{j=0}^{i-1}(\tau_{K,j}+1).$$

Note that $t_K^{(i)}$ is a stopping time with respect to $\mathcal{H}$. Using $t_K^{(i)}$,

we set

$$\bar{\mathbf{X}}_i = \mathbf{X}_{t_K^{(i)}},\ i \geq 0, \tag{17}$$

and define $\mathcal{H}^\tau$ as the filtration given by $\mathcal{H}^\tau = \{\mathcal{H}_{t_K^{(i)}}\}_{i=0}^\infty$. In addition, associated with η_i, which is given by

$$\eta_i = \xi(\tau_{K,i}+1)\|\mathbf{X}_{t_K^{(i)}}\|,$$

we define $\eta^{(i)}$ as

$$\eta^{(0)} = 0, \quad \eta^{(i)} = \sum_{j=0}^{i-1} \eta_j. \tag{18}$$

We also define ν as

$$\nu = \inf\{i \geq 0 : t_K^{(i)} \geq t + \sigma_C\}, \tag{19}$$

which is a stopping time with respect to $\mathcal{H}^\tau$. Intuitively, $i = \nu$ marks the first time $t_K^{(i)}$ at or before which the process $\{\mathbf{X}_{t+i}\}_{i=0}^\infty$ enters $\mathcal{C}$. We finish the chain of definitions by introducing the sequence $\{Z_i\}_{i=0}^\infty$, where

$$Z_i = f(\bar{\mathbf{X}}_i) + \eta^{(i)}. \tag{20}$$

For $i < \nu$, using (18), we have

$$\begin{aligned}\mathbb{E}[Z_{i+1}|\mathcal{H}_{t_K^{(i)}}] &= \mathbb{E}\big[f(\bar{\mathbf{X}}_{i+1}) + \eta_i|\mathcal{H}_{t_K^{(i)}}\big] + \eta^{(i)} \\ &\leq f(\bar{\mathbf{X}}_i) + \eta^{(i)} = Z_n,\end{aligned} \tag{21}$$

where the first equality follows from the fact that $\eta^{(i)}$ is completely determined given $\mathcal{H}_{t_K^{(i)}}$, and the inequality is simply an immediate result of (16) and the assumption $i < \nu$. To simplify the notation, let $\nu \wedge i$ denote

$$\nu \wedge i = \min(\nu, i).$$

It now follows directly from (21) that the sequence $\{Z_{\nu\wedge i}\}_{i=0}^\infty$ is an $\mathcal{H}^\tau$-supermartingale. Since $f(\cdot)$ is non-negative, we have

$$\mathbb{E}[\eta^{(\nu\wedge i)}|\mathcal{H}_t] \leq \mathbb{E}[Z_{\nu\wedge i}|\mathcal{H}_t].$$

But $\mathcal{H}_t = \mathcal{H}_{t_K^{(0)}}$, and $\{Z_{\nu\wedge i}\}_{i=0}^\infty$ is a supermartingale. Hence,

$$\begin{aligned}\mathbb{E}[Z_{\nu\wedge i}|\mathcal{H}_t] &= \mathbb{E}[Z_{\nu\wedge i}|\mathcal{H}_{t_K^{(0)}}] \\ &\leq Z_0 = f(\mathbf{X}_t).\end{aligned}$$

Considering the last two inequalities, we obtain

$$\mathbb{E}[\eta^{(\nu\wedge i)}|\mathcal{H}_t] \leq f(\mathbf{X}_t). \tag{22}$$

In addition, using the definition of $\eta^{(i)}$ and η_j while assuming $M_K > 1$, it is easy to see that

$$\begin{aligned}\eta^{(\nu\wedge i)} = \sum_{j=0}^{i-1} \eta_j \mathbf{1}_{(j<\nu)} &\geq \xi \sum_{j=0}^{i-1} (\tau_{K,j}+1)\mathbf{1}_{(j<\nu)} \\ &= \xi(t_K^{(\nu\wedge i)} - t).\end{aligned} \tag{23}$$

Applying the monotone convergence theorem [20], we can take the limit in (22) and (23) as $i \to \infty$ yielding

$$\mathbb{E}[t_K^{(\nu)} - t|\mathcal{H}_t] \leq \xi^{-1} f(\mathbf{X}_t).$$

But by definition in (19), $\sigma_\mathcal{C} \leq t_K^{(\nu)} - t$. Thus, for $\mathbf{X}_t \notin \mathcal{C}$

$$\mathbb{E}[\sigma_\mathcal{C}|\mathcal{H}_t] \leq \xi^{-1} f(\mathbf{X}_t).$$

If $\mathbf{X}_t \in \mathcal{C}$, we have $\sigma_\mathcal{C} = 0$. Hence, we have that

$$\mathbb{E}[\sigma_\mathcal{C}|\mathcal{H}_t] \leq \xi^{-1} f(\mathbf{X}_t)\mathbf{1}_{\mathbf{X}_t \notin \mathcal{C}},$$

showing that the expected $\sigma_\mathcal{C}$ is bounded by a function of $\mathbf{X}_t$ uniformly in the past history and t, as required. ■

REFERENCES

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[2] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.

[3] A. Eryilmaz, R. Srikant, and J. Perkins, "Stable scheduling policies for fading wireless channels," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 411–424, Apr. 2005.

[4] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer rate control in wireless networks," *in Proc. IEEE INFOCOM'05*, vol. 3, pp. 1804–1814, Mar. 2005.

[5] G. Sharma, R. R. Mazumdar, and N. B. Shroff, "On the complexity of scheduling in wireless networks," in *Proc. of the 12th annual international conference on Mobile computing and networking (MobiCom'06)*, 2006, pp. 227–238.

[6] X. Wu and R. Srikant, "Regulated maximal matching: A distributed scheduling algorithm for multi-hop wireless networks with node-exclusive spectrum sharing," in *44th IEEE conference on decision and control, and European control conference CDC-ECC*, dec 2005.

[7] M. Lotfinezhad, B. Liang, and E. Sousa, "On the stability region of linear-memory scheduling for time varying channels," in *Fifteenth IEEE International Workshop on Quality of Service (IWQoS'07)*, jun 2007.

[8] G. Sharma, N. B. Shroff, and R. R. Mazumdar, "Joint congestion control and distributed scheduling for throughput guarantees in wireless networks," in *Proc. IEEE INFOCOM'07*, may 2007.

[9] B. H. Fralix, "Foster-type criteria for Markov chains on general spaces," *Journal of Applied Probability*, vol. 43, no. 4, pp. 1194–1200, 2006.

[10] S. Shakkottai and A. Stolyar, "Scheduling for multiple flows sharing a time-varying channel: The exponential rule," *Translations of AMS, A Volume in Memory of F. Karpelevich*, vol. 207, 2002.

[11] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *in Proc. IEEE INFOCOM'05*, vol. 3, pp. 1723–1734, Mar. 2005.

[12] M. J. Neely, "Energy optimal control for time varying wireless networks," *in Proc. IEEE INFOCOM'05*, vol. 1, pp. 572–583 vol. 1, Mar. 2005.

[13] C. Li and M. J. Neely, "Energy-optimal scheduling with dynamic channel acquisition in wireless downlinks," in *Proc. of 46th IEEE Conf. on Decision and Control (invited paper)*, dec 2007.

[14] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *in Proc. IEEE INFOCOM'05*, vol. 3, pp. 1794–1803, Mar. 2005.

[15] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle;, "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," in *Proc. IEEE INFOCOM'06*, apr 2006.

[16] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," *in Proc. IEEE INFOCOM'98*, vol. 2, 1998.

[17] P. Chaporkar and S. Sarkar, "Stable scheduling policies for maximizing throughput in generalized constrained queueing," in *Proc. IEEE INFOCOM'06*, apr 2006.

[18] X. Wu and R. Srikant, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," in *Proc. IEEE INFOCOM'06*, apr.

[19] M. J. Neely, "Delay analysis for maximal scheduling in wireless networks with bursty traffic," in *Proc. IEEE INFOCOM'08*, apr 2008.

[20] A. A. Borovkov, *Probability Theory*. Gordon and Breach, 1998.

[21] L. Tassiulas, "Scheduling and performance limits of networks with constantly changing topology," *IEEE Trans. Inf. Theory*, vol. 43, no. 3, pp. 1067–1073, May 1997.

[22] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[23] M. Lotfinezhad, B. Liang, and E. Sousa, "Dynamic control of tunable sub-optimal algorithms for scheduling of time-varying wireless networks," University of Toronto, Tech. Rep., 2008. [Online]. Available: http://www.comm.utoronto.ca/∼liang/publications/preprints/DCP.pdf

[24] R. Buche and H. J. Kushner, "Control of mobile communication systems with time-varying channels via stability methods," *IEEE Trans. Autom. Control*, vol. 49, no. 11, pp. 1954–1962, Nov. 2004.

[25] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.