Spatial Inefficiency of MaxWeight Scheduling

Peter van de Ven[1,2], Sem Borst[1,3], Lei Ying[4]

Abstract

MaxWeight scheduling has gained enormous popularity as a powerful paradigm for achieving queue stability and maximum throughput in a wide variety of scenarios. The maximum-stability guarantees however rely on the fundamental premise that the system consists of a fixed set of flows with stationary ergodic traffic processes. In the present paper we examine networks where the population of active flows varies over time, as flows eventually end while new flows occasionally start. We show that MaxWeight policies may fail to provide maximum stability due to persistent inefficient spatial reuse. The intuitive explanation is that these policies tend to serve flows with large backlogs, even when the resulting spatial reuse is not particularly efficient, and fail to exploit maximum spatial reuse patterns involving flows with smaller backlogs. These results indicate that instability of MaxWeight scheduling can occur due to spatial inefficiency in networks with fixed transmission rates, which is fundamentally different from the inability to fully exploit time-varying rates shown in prior work. We discuss how the potential instability effects can be countered by spatial traffic aggregation, and describe some of the associated challenges and performance trade-offs.

1 Introduction

MaxWeight scheduling has gained immense popularity as a powerful concept for achieving maximum throughput and queue stability in a wide variety of scenarios. In a seminal paper, Tassiulas & Ephremides [28] presented a MaxWeight scheduling policy for throughput maximization in multi-hop wireless networks, where only certain subsets of the links may be activated simultaneously due to interference considerations, see also Kahale & Wright [6] for instance. In subsequent work, Tassiulas & Ephremides [29] described a MaxWeight policy for allocating a server among several parallel queues with time-varying connectivity.

Broadening the latter framework, MaxWeight-type policies were developed for power control and scheduling of wireless channels with rate variations, see

[1]Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[2]Eurandom, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[3]Bell Laboratories, Alcatel-Lucent, P.O. Box 636, Murray Hill, NJ 07974, USA
[4]Department of Electrical & Computer Engineering, Iowa State University, Ames, IA 50011, USA

for instance Andrews *et al.* [1], Eryilmaz *et al.* [3], Neely [17] and Neely *et al.* [19]. Extending the scope further, Eryilmaz & Srikant [4], Neely *et al.* [18] and Stolyar [25, 26] devised algorithms for joint congestion control, routing and scheduling based on MaxWeight principles. The powerful properties of MaxWeight-type policies have emerged as one of the central paradigms in the broader realm of cross-layer control and resource allocation in wireless networks, see Georgiadis *et al.* [5] for a comprehensive overview.

MaxWeight-type algorithms have also been proposed for throughput maximization in input-queued switches, where only certain subsets of input-output pairs (e.g. matchings) may be simultaneously connected because of compatibility constraints, see for instance McKeown *et al.* [13, 14]. The book of Meyn [15] contains extensive background material on MaxWeight policies. Crucial heavy-traffic results for MaxWeight algorithms were obtained by Stolyar [24].

The distinguishing characteristic of MaxWeight policies is that the subset of queues that are simultaneously served is selected so as to be of maximum "weight", hence the term "MaxWeight". The weight of a queue is usually defined as the current backlog or the product of the backlog and the feasible instantaneous service rate for that queue, if selected. The combinations of queues which can be scheduled simultaneously are subject to certain constraints, based on for example interference conditions. In a more general sense, MaxWeight policies can be interpreted as selecting a service vector from a (possibly time-varying) feasible region that maximizes the inner product with the backlog vector.

Under mild assumptions, MaxWeight-type algorithms have been shown to provide maximum throughput, i.e., achieve queue stability whenever feasible to do so at all. A particularly appealing feature is that MaxWeight policies only need information on the current backlogs and instantaneous service rates, and do not rely on any explicit knowledge of the rate distributions or the traffic parameters. On the downside, finding the maximum-weight subset is often a challenging problem and potentially NP-hard, which is exacerbated in a distributed setting, where message passing and exchange of backlog information create a substantial communication overhead in addition to the computational burden. This issue is especially pertinent as the maximum-weight problem generally needs to be solved at a very high pace, commensurate with the fast time scale on which scheduling algorithms tend to operate. In order to address this issue, Tassiulas [27], Eryilmaz *et al.* [3] and Chaporkar & Sarkar [2] showed that randomized policies involve less stringent requirements and yet suffice for achieving maximum stability. In addition, several authors have considered algorithms that solve the maximum weight problem in some approximate sense, and quantified the resulting penalty in guaranteed throughput, see for instance Lin & Shroff [8], Sharma *et al.* [22, 23] and Wu & Srikant [31].

Under mild assumptions, MaxWeight-type policies have been shown to achieve maximum stability. A fundamental premise however is that the system consists of a fixed set of queues with stationary ergodic traffic processes. In reality, the collection of active queues dynamically varies, as sessions eventually end, while new sessions occasionally start. In many situations the assumption of a fixed set of queues is still a reasonable modeling convention, since the scheduling actions

and packet-level queue dynamics tend to occur on a very fast time scale, on which the population of active sessions evolves only slowly. In other cases, however, sessions may be relatively short-lived, and the above time scale separation argument does not apply. The impact of flow-level dynamics over longer time scales is particularly relevant in assessing stability properties, as the notion of stability only has strict meaning over infinite time horizons.

Motivated by the above observations, [30] examined the stability properties of MaxWeight scheduling policies in the presence of flow-level dynamics. It was shown that these policies may fail to achieve maximum stability in wireless channels with time-varying transmission rates. The analysis in [30] also identified algorithms that do provide maximum stability, but these were geared to yield tractable behavior and required explicit knowledge of various system parameters. Subsequent work [10, 11] proposed more practical scheduling algorithms guaranteeing throughput optimality, and extended these to multi-channel scenarios. Sadiq & De Veciana [21] showed that a *delay*-driven (as opposed to *queue*-based) version of MaxWeight scheduling does guarantee maximum stability in the presence of flow-level dynamics and rate variations. A somewhat different manifestation of "queue instability" under MaxWeight policies for a fixed set of heavy-tailed traffic sources was studied by Markakis *et al.* [12].

It is crucial to observe that the rate variations play a critical role in the above-mentioned instability results. Intuitively speaking, MaxWeight policies tend to give preferential treatment to flows with large backlogs, even when their service rates are not particularly favorable, and thus fail to maximally exploit the rate variations of flows with smaller backlogs. This raises the question whether the rate variations are essential for the instability to occur. In the case of a shared downlink, where only a single flow can be scheduled at a time, the instability cannot occur in the absence of any rate variations, since this system is work-conserving, and any non-idling scheduling strategy will in fact achieve maximum stability.

The more challenging problem, however, arises in network settings as originally considered in [28] where certain subsets of the links can be activated simultaneously subject to interference constraints. In the present paper we will show that MaxWeight scheduling policies may fail to provide maximum stability in such scenarios as well, even in the absence of any rate variations. Loosely stated, MaxWeight policies tend to serve flows with large backlogs, even when the resulting spatial reuse is not particularly efficient, and neglect to take advantage of maximum spatial reuse patterns involving flows with smaller backlogs. These results indicate that the instability of MaxWeight scheduling can occur due to spatial inefficiency in networks with fixed transmission rates, which is fundamentally different from the inability to fully exploit time-varying transmission rates as in [30].

Note that the preferential treatment of flows with large backlogs in fact also applies in the absence of any flow-level dynamics. In that case the phenomenon cannot persist however since the flows with smaller backlogs will build larger queues and gradually start improving the spatial efficiency, creating a counteracting force. In contrast, in the presence of flow-level dynamics, MaxWeight

policies may constantly get diverted to arriving flows, while neglecting the opportunity to exploit higher spatial reuse patterns involving a persistently growing number of flows with relatively small remaining backlogs, so the opposing effect is never triggered.

It is worth observing that the possibly unbounded number of flow locations greatly exacerbates the computational complexity of solving the maximum-weight problem noted earlier. However, in the analysis we assume that the maximum-weight problem itself is solved to optimality in each time slot. Thus the instability of MaxWeight policies as discussed above is entirely disjoint from the throughput penalty which may result from solving the maximum-weight problem only approximately as considered for example in [8, 22, 23, 31]. It is further worth drawing a distinction with the work of Lin *et al.* [9] and Moallemi & Shah [16] showing the stability of joint scheduling and congestion control algorithms in the presence of flow-level dynamics without relying on the conventional simplifying time scale separation argument. The main difference with the present paper lies in the fact that in these studies the set of flow routes is fixed and that scheduling operates at a class level.

While the root cause for instability observed in this paper (flow-level dynamics) is the same as in [30], the way the presence of transient flows unhinges the MaxWeight scheduling algorithm is completely different. Consequently, the remedies for instability discussed in [10, 11, 21] cannot be directly applied in the current setting, and the spatial inefficiency identified in this paper calls for novel methods for stabilizing the system.

As we will show, the potential instability effects can be countered by implementing a region-based version of MaxWeight scheduling. However, the identification of adequate regions is quite challenging, since the degree of traffic aggregation involves a trade-off between scheduling complexity, spatial efficiency, and network capacity. In particular, the suitable level of aggregation depends on the spatial load profile, and seems difficult to determine without explicit knowledge of the traffic parameters, thus detracting from one of the most appealing features of MaxWeight scheduling mentioned earlier.

The remainder of the paper is organized as follows. In Section 2 we provide a detailed model description, and in Section 3 we demonstrate the potential instability of MaxWeight scheduling through several examples. In Section 4 we examine the performance of region-based scheduling in two-dimensional networks with an arbitrary spatial traffic density. Section 5 offers some concluding remarks.

2 Model description

We consider a time-slotted wireless system on some space $\mathcal{S}$. Traffic consists of finite-sized flows that enter the system at random, and leave once fully served. Each arriving flow is associated with a certain location in $\mathcal{S}$ and a finite size, as will be further described for specific model instances later. In each time slot, a centralized scheduler selects a subset of flows for transmission. In the

present paper we assume for simplicity that the total size of a flow is known upon arrival, and no further traffic will arrive into that flow. Most of the results can be extended to a setting with gradual traffic.

A subset of points in $\mathcal{S}$ is said to be feasible if flows in these locations can be scheduled simultaneously. The function $F(\cdot)$ indicates whether or not a subset of points is feasible, i.e., given n flows with distinct locations $P_1, \ldots, P_n \in \mathcal{S}$, $F(\{P_1, \ldots, P_n\})$ equals 1 if these flows can be scheduled simultaneously and is 0 otherwise. Flows in the same location can never be scheduled simultaneously. A prototypical scenario would be that $F(\{P_1, \ldots, P_n\}) = 1$ if and only if $\|P_i - P_j\| \geq d$ for all $i \neq j$, which corresponds to a so-called protocol model with reuse distance d. However, the feasibility function could also be based on SINR constraints for example.

In each time slot a certain subset of flows gets selected for service, as governed by the applicable scheduling strategy, subject to the feasibility constraints. Each time a flow gets scheduled, its residual size is reduced by 1, and a flow leaves the system once it has been served to completion i.e., its size reaches 0. The subset of flows selected by the scheduling strategy may depend on the locations $P_i(t)$ and residual sizes $Q_i(t)$, $i \in I(t)$, with $I(t)$ indexing the flows present in time slot t. In particular, the MaxWeight scheduling strategy selects a feasible subset of flows $J^*(t) \subseteq I(t)$, $F(J^*(t)) = 1$, of maximum aggregate residual size, i.e., $\sum_{j \in J^*(t)} Q_j(t) = \max_{J \subseteq I(t), F(J)=1} \sum_{j \in J} Q_j(t)$.

Besides notational convenience, the main reason for assuming unit transmission rates in the above model description is to stress the fact that the instability phenomena demonstrated in later sections result from persistent spatial inefficiency rather than rate heterogeneity. Possible rate heterogeneity will induce priorities among flows, which may exacerbate the spatial inefficiency and render the system even more prone to potential instability effects.

3 Instability of MaxWeight Scheduling

In this section we present several illustrative examples where the MaxWeight scheduling strategy fails to achieve maximum stability.

Example 1. *We first consider a network with three regions as shown in Figure 1. Transmissions in region 2 interfere with transmissions in both region 1 and region 3, and transmissions in regions 1 and 3 do not interfere with each other. Flows arrive at region i at a rate λ_i (per time slot) and have initial size B_i. Denote by $\rho_i = \lambda_i \mathbb{E}\{B_i\}$ the traffic intensity at region i. We assume that*

$$\rho_1 + \rho_2 < 1 \text{ and } \rho_3 + \rho_2 < 1,$$

or equivalently,

$$\rho_2 < 1 - \max\{\rho_1, \rho_3\}. \tag{1}$$

It is easily seen that the latter condition is necessary for stability to be achievable, and in fact also sufficient under mild independence assumptions. A scheduling

strategy that can stabilize the network when (1) holds is as follows. At each time slot, schedule a flow in region 2 with probability $\rho_2+\epsilon$*, or schedule a flow in both regions 1 and 3, with probability* $\max\{\rho_1,\rho_3\}+\epsilon$*, where* $\epsilon = \frac{1-\rho_2-\max\{\rho_1,\rho_3\}}{2}$.

Now suppose $B_2 = 1$*, and recall that the MaxWeight scheduling strategy as defined in the general network model of the previous section selects a set of* flows *with maximum aggregate residual size. Thus the MaxWeight strategy will never schedule a flow in region 2 as long as a flow with a residual size of 2 or larger is present in region 1 or region 3. Hence the scheduling of flows of residual size 2 or larger in region 1 and region 3 is independent from each other. Also, the fraction of time that a flow of residual size 2 or larger gets scheduled in region* i*, is* $\lambda_i(\mathbb{E}\{B_i\}-1) = \rho_i - \lambda_i$*. It follows that the fraction of time that a flow in region 2 gets scheduled, is bounded from above by*

$$(1-\rho_1+\lambda_1)(1-\rho_3+\lambda_3).$$

Thus a necessary condition for MaxWeight scheduling to achieve stability is $\rho_2 \leq (1-\rho_1+\lambda_1)(1-\rho_3+\lambda_3)$*. When the* λ_i*'s (*$i=1,3$*) are small and the* $\mathbb{E}\{B_i\}$*'s (*$i=1,3$*) are large, the latter condition 'approaches'* $\rho_2 \leq (1-\rho_1)(1-\rho_3)$*, which is a more stringent inequality than the sufficient condition* (1). □

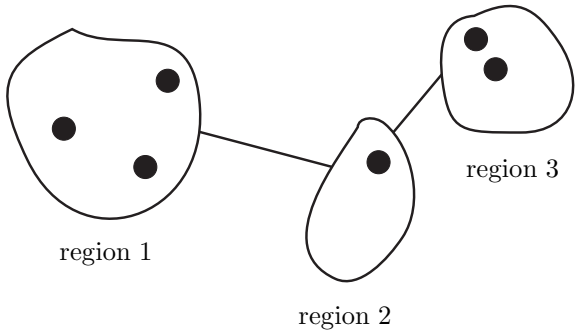

Figure 1: An example of a spatial ad hoc wireless network where MaxWeight scheduling is not throughput-optimal.

In the above example, the stabilizing strategy either schedules both region 1 and region 3, or schedules region 2. The MaxWeight policy, however, tends to serve flows with large backlogs, so flows in regions 1 and 3 are served with priority when their residual sizes are greater than or equal to 2. Consequently, the MaxWeight policy schedules a flow in region 1 (region 3) even when region 3 (region 1) is empty, which leads to inefficient spatial reuse. Thus, the MaxWeight policy fails to achieve maximum stability.

Example 1 illustrates the spatial inefficiency of MaxWeight scheduling by carefully constructing the regions where flows arrive. Next we present a less restrictive example where we consider a one-dimensional space (a ring) with

uniformly distributed arrival locations. We assume that all flows are of the same size, and we will show that even in this *uniform* traffic scenario, MaxWeight scheduling fails to achieve throughput optimality.

Example 2. *Let $N \geq 1$ and consider a ring with unit circumference and reuse distance $d = 2(N+1)/((2N+3)(3N+2))$, partitioned into $(2N+3)(3N+2)$ intervals of equal size, see Figure 2. In each time slot, either exactly $(2N+3)$ flows arrive with probability a, each of size $B = 2$, at locations uniformly distributed in the intervals $M + j(3N+2)$, $j = 1, 2, \ldots, (2N+3)$, where M is uniformly distributed on $1, 2, \ldots, 3N+2$, or no flows arrive at all with probability $1 - a$.*

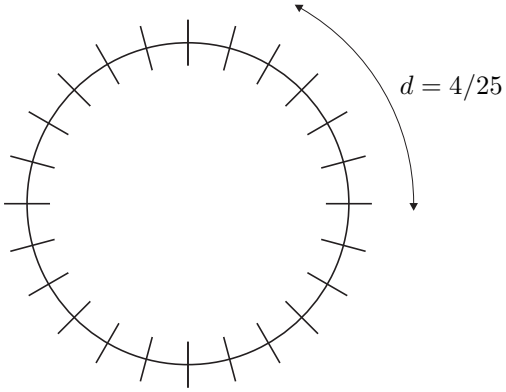

Figure 2: A ring with unit circumference, reuse distance $d = 4/25$, partitioned into 25 intervals of equal size ($N = 1$)

Consider a strategy that generates a random variable L uniformly distributed on $1, 2, \ldots, 2N+3$, and then selects an arbitrary flow for service from each of the intervals $L + i(2N+3)$, $i = 0, 1, \ldots, 3N+1$, if available. Note that the strategy respects the reuse distance, and achieves stability as long as the aggregate traffic intensity in each interval, $2a/(3N+2)$, is less than the fraction of time slots that each interval gets selected for service, $1/(2N+3)$, or equivalently, if $a < a(N) = (3N+2)/(4N+6)$. Note that $a(N) \to 3/4$ as $N \to \infty$. Also, the maximum size of a feasible subset of points is $M(N) = \lfloor \frac{(2N+3)(3N+2)}{2(N+1)} \rfloor$, and the total traffic intensity equals $\rho = 2a(2N+3)$, so the necessary condition $\rho < M$ for stability takes the form

$$a < b(N) = \frac{1}{2(2N+3)} \left\lfloor \frac{(2N+3)(3N+2)}{2(N+1)} \right\rfloor.$$

Observe that $b(N) \to 3/4$ as $N \to \infty$, and thus the above-described strategy in fact achieves maximum stability for large values of N.

It is easily verified that in each time slot with arriving flows, the MaxWeight strategy selects all $2N+3$ of them for service, while in a time slot without any

arrivals, it can serve at most $3N+2$ traffic units, so the expected total number of traffic units served per time slot is bounded from above by $a(2N+3)+(1-a)(3N+2)$. As a necessary condition in order for the MaxWeight strategy to be stable, the latter number must be larger than the total traffic intensity $2a(2N+3)$, which entails $a < a^{MW}(N) = (3N+2)/(5N+5)$. Note that $a^{MW}(N) \leq a(N)$, with strictly inequality for all $N \geq 2$, and that $a^{MW}(N) \to 3/5$ as $N \to \infty$.

We conclude that for $a \in (a^{MW}(N), a(N))$, the MaxWeight strategy fails to achieve stability, although there exists a strategy that does provide stability. For large values of N the MaxWeight strategy is only able to sustain at most a fraction 4/5 of the maximum throughput. □

In the above example MaxWeight scheduling always selected newly arrived flows for transmission, even when it could have chosen a schedule that allowed for better spatial reuse. This persistent inefficiency then leads to instability. As we will see below, this behavior occurs for more general traffic patterns as well.

The locations of arriving flows in Example 2 are uniformly distributed, but highly correlated. When the flow locations are independent, the behavior is more complex, and stability is more difficult to establish. We therefore proceed with a simulation experiment where we assume that the locations of arriving flows are independent. As we show in the next example, the MaxWeight strategy again fails to achieve throughput optimality.

Example 3. *Consider a ring network where the total number of arriving flows is geometrically distributed with parameter $p = 0.45$ and mean $\alpha = \frac{1-p}{p} \approx 1.22$, so $\rho = \alpha \mathbb{E}\{B\} \approx 2.44$. We assume that the locations of the flows are independent and uniformly distributed along the ring. The reuse distance is $d = 0.3$, and hence the maximal number of flows that can be scheduled simultaneously equals $M = 3$.*

We compare the performance of MaxWeight scheduling with that of a randomized interval-based scheduling strategy. We divide the ring into 42 intervals of length 1/42. We consider 42 schedules $\omega_k = \{k, k+14, k+28\}$ (modulo 42), and choose in each time slot one of these schedules uniformly at random.

We simulate the network 1000 slots, for both MaxWeight scheduling and the randomized strategy. Figure 3 shows the total number of flows present over time for MaxWeight scheduling (gray) and the randomized strategy (black). Under MaxWeight scheduling the number of flows grows unbounded, suggesting instability. In contrast, the number of flows settles around a relatively low level for the randomized strategy. □

4 Stability of region-based scheduling

In the previous section we demonstrated the spatial inefficiency of MaxWeight scheduling. This raises the question of finding scheduling algorithms that can be used to stabilize spatial networks with flow-level dynamics. For the single-channel case with flow-level dynamics it was recently shown that maximum stability can be achieved by scheduling according to the feasible transmission

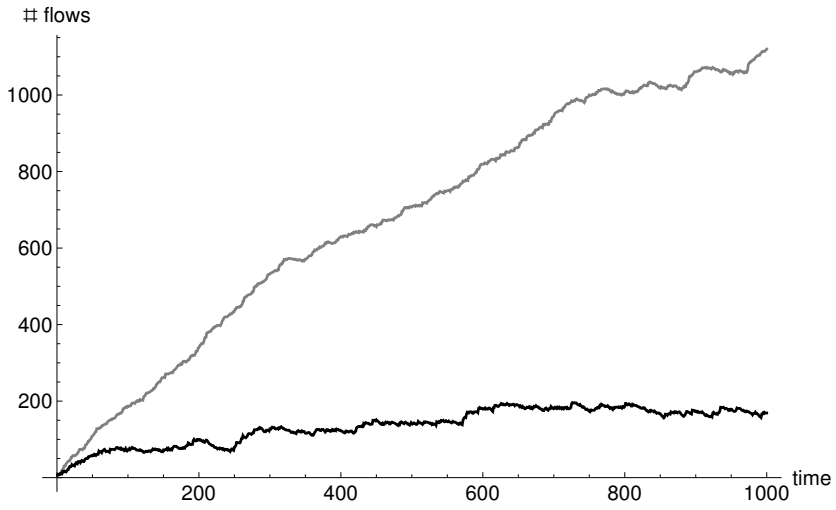

Figure 3: Evolution of the total number of flows for MaxWeight scheduling (gray) and an interval-based randomized scheduler (black), respectively.

rate [10, 11] or according to the product of feasible transmission rate and the delay [21]. It is not clear whether these policies are throughput-optimal in the spatial setting, or how they may need be modified to provide maximum stability. Both schedulers have to find a maximum (weighted) independent set over all flows in each time slot, and since the number of flows is unbounded, so is the scheduling complexity. In this section we present a class of policies that do have bounded complexity.

Consider a two-dimensional network with an arbitrary spatial traffic density. We assume that the space $\mathcal{S}$ is bounded, and without loss of generality we may suppose that location coordinates are scaled such that $\mathcal{S}$ is contained in the unit square $[0,1]^2$. The number of arriving flows, their locations, and their sizes are independent and identically distributed across time slots. The location of an arbitrary arriving flow is governed by some spatial measure λ on $[0,1]^2$, with $\lambda(x,y) = 0$ for all $(x,y) \notin \mathcal{S}$, i.e., the expected number of arriving flows per time slot in a region $\mathcal{R} \subseteq \mathcal{S}$ is $\int_{(x,y)\in\mathcal{R}} \lambda(x,y)\,\mathrm{d}x\mathrm{d}y$. Let the positive random variable B represent the size of an arbitrary flow.

4.1 Two special cases

The above general network setting includes the three-region network and the ring topology with uniform traffic density discussed in Section 3. Before presenting results on the stability of general spatial networks with flow-level dynamics, let us first consider maximal stable policies for these two special cases. For the three-region network, an alternative view of the network is to consider

each region as a single node. The three-region network can then be seen as a classic three-node network, where packets belonging to various flows are continuously injected into each node. It is easily verified that in this case the MaxWeight strategy that schedules according to the aggregate backlog at each node is throughput-optimal.

Now consider the ring network with uniform spatial traffic density α. Taking a similar approach as for the three-region network, instead of scheduling flows, we divide the ring into intervals and schedule these intervals instead. As we will show in the following proposition, for the right choice of intervals the ring network can be stabilized using such interval-based scheduling.

Proposition 1. *Consider a ring with unit circumference, uniform spatial traffic density, and a translation-invariant feasibility function, and denote by M the maximum number of flows that can be scheduled simultaneously. Then there exists an interval-based scheduling strategy that achieves stability for any load $\rho := \alpha\mathbb{E}\{B\} < M$.*

Proof. Let $\mathcal{P} = \{P_1, \ldots, P_M\} \subseteq [0, 1]$ denote a feasible maximum-size set and assume that there exist some $\epsilon > 0$ such that for any $\hat{P}_i \in [P_i, P_i + \epsilon)$, the set $\hat{\mathcal{P}} = \{\hat{P}_1, \ldots, \hat{P}_M\}$ is feasible as well. We choose K_i, $i = 1, 2, \ldots, M$ and K be integers such that each interval $[P_i, P_i + \epsilon)$ contains the points K_i/K and $(K_i + 1)/K$, $i = 1, 2, \ldots, M$. We partition the ring into K intervals, each of size $1/K$. Now consider a cyclic scheduling strategy which in time slot $tK + u$, $u = 1, \ldots, K$, $t = 0, 1, \ldots$, serves the intervals $[(K_i + u)/K, (K_i + 1 + u)/K]$, $i = 1, \ldots, M$ by selecting an arbitrary flow from each of these intervals, if available. Note that any set of flows thus selected is allowed since the feasibility function is translation-invariant. Also, each interval is allowed to be served a fraction of the time M/K and has aggregate traffic intensity ρ/K. Hence, the strategy achieves stability for any $\rho < M$. □

Note that Proposition 1 assumes a general interference model, so it includes the model with reuse distance discussed in Examples 2 and 3 as a special case. Consider the case where the reuse distance is d, and assume $1/d$ is not an integer. Then $M = \lfloor 1/d \rfloor$ and $\epsilon = 1/M - d$. It is easy to verify that given the reuse distance d, the necessary condition for ρ to be supportable is $\rho \leq \lfloor 1/d \rfloor$ and that the scheduling algorithm presented in the proof can stabilize any $\rho < \lfloor 1/d \rfloor$. In general, the required granularity of the partitioning depends on the feasibility function through ϵ, but not on the traffic intensity.

4.2 General networks

The examples in section 4.1 indicate that in the presence of flow-level dynamics we should aggregate over several nearby flows, rather than schedule based on individual flows. This suggests a *region-based* scheduling algorithm where the space is partitioned into a finite number of regions. In each time slot, the algorithm selects a subset of non-interfering regions and then schedules a flow in each selected region.

Naturally, such partitioning would reduce the flexibility of the scheduler since the region-based feasibility constraints are more stringent than the original constraints. Region-based scheduling is nevertheless useful because, in contrast to the partition-free system, throughput-optimal schedulers are available in this case. For example, since the partitioned system behaves as a network with a finite number of persistent queues, it is well-known that region-based MaxWeight scheduling (i.e., MaxWeight scheduling based on the aggregate backlog of all flows in a region) is throughput-optimal within the class of schedulers that satisfy the more rigid feasibility constraints of the partitioned system. We are interested in how the capacity region of the partitioned system relates to the capacity region under the original reuse constraints. As we will see, this depends on the granularity of the partitioning. Note that region-based MaxWeight scheduling limits the scheduling complexity, as the number of regions is fixed.

We continue to consider a specific form of region-based scheduling referred to as K-partition, where the area $[0,1]^2$ is partitioned into K^2 square cells of size $1/K^2$, for some $K \in \mathbb{N}$. The cells are denoted $\mathcal{R}_{k,l} = [(k-1)/K, k/K] \times [(l-1)/K, l/K]$, $k, l = 1, \ldots, K$. The 4-partition is illustrated in Figure 4.

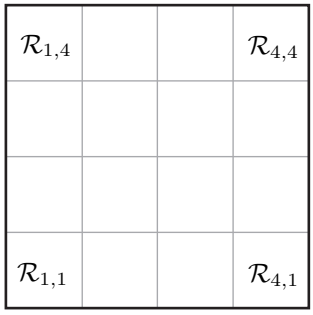

Figure 4: The 4-partition, where the unit square is divided into 16 cells.

Under K-partition, a set of cells $\mathcal{R}_{k_1,l_1}, \mathcal{R}_{k_2,l_2}, \ldots, \mathcal{R}_{k_n,l_n}$ is said to be feasible if for any $P_i \in \mathcal{R}_{k_i,l_i}$, $i = 1, \ldots, n$, the set of points $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ is feasible. For convenience, we henceforth assume that the feasibility function is governed by a protocol model with reuse distance d. Denote by $\Omega(d)$ all feasible sets of points, and let $\Omega(K,d) \subseteq \{0,1\}^{K^2}$ represent the collection of all feasible subsets of cells. So for $\omega \in \Omega(K,d)$ we have that $\omega_{k,l} = 1$ if $\mathcal{R}_{k,l}$ is contained in the schedule ω, and $\omega_{k,l} = 0$ otherwise. We focus on square regions for convenience, but we expect that for other types of regions the same qualitative results hold .

Under K-partition, scheduling is confined to subsets of flows that belong to a feasible subset of cells, which restricts beyond the original reuse constraint and guarantees feasibility. The aggregate arrival rate of flows into the cell $\mathcal{R}_{k,l}$ is given by $\lambda_{k,l} := \int_{(x,y)\in\mathcal{R}_{k,l}} \lambda(x,y)\,\mathrm{d}x\mathrm{d}y$. The capacity region for such a system

is well-known:

$$\mathcal{C}(K,d) = \big\{\lambda \ : \ \lambda_{k,l}\mathbb{E}\{B\} \in \text{conv.hull}(\Omega(K,d))\big\}.$$

Let $\mathcal{C}(d)$ denote the capacity region under the original reuse constraint, then for any $K \geq 1$ we have that $\mathcal{C}(K,d) \subseteq \mathcal{C}(d)$. As K increases, the granularity of the partitioning becomes finer, and it is intuitive that $C(K,d)$ converges to $C(d)$ in a certain sense. This is formalized in the following theorem, where we show that for any arrival density function $\lambda \in C(d)$ (under certain assumptions) there exists a K such that λ is contained in $\mathcal{C}(K,d)$.

Before stating our main result on the stability of region-based scheduling, we first present the following lemmas.

Lemma 1. *Let $d > 0$ and $K \in \mathbb{N}$, $K > 2\sqrt{2}/d$, then*

$$\mathcal{C}(d) \subseteq \mathcal{C}(K, d - 2\sqrt{2}/K).$$

The proof of Lemma 1 is presented in Appendix A.1.

Let $\omega \in \Omega(K,d)$, $L \leq K$ and denote by $\omega^{(L)}$ the vector ω restricted to the entries $\omega_{k,l}$, $k,l = 1,2,\ldots,L$. Then the following lemma holds.

Lemma 2. *Let $d > 0$, $K \in \mathbb{N}$ and set*

$$h = K\Big\lfloor \frac{K(d - 2\sqrt{2}/K)}{d} \Big\rfloor^{-1}. \tag{2}$$

Then $\omega \in \Omega(K,d) \Rightarrow \omega^{(K/h)} \in \Omega(k/h, d)$.

Lemma 2 states that if ω is a feasible set of cells under K-partition and reuse distance $d - 2\sqrt{2}/K$, then it is a feasible set of cells under (K/h)-partition and reuse distance d as well. The proof of the lemma is presented in Appendix A.2.

We are now in position the prove our main result. An arrival density function λ is said to be smooth if it is

- uniformly lower bounded, i.e., there exists a $\kappa^{(0)} > 0$ such that $\lambda(x,y) \geq \kappa^{(0)}$ for all $(x,y) \in \mathcal{S}$;
- differentiable, with a uniformly upper bounded first-order partial derivative, i.e., there exists a $\kappa^{(1)} < \infty$ such that $\frac{\partial\lambda(x,y)}{\partial x} \leq \kappa^{(1)}$ and $\frac{\partial\lambda(x,y)}{\partial y} \leq \kappa^{(1)}$ for all $(x,y) \in \mathcal{S}$.

Theorem 1. *Let λ be a smooth arrival density function such that $(1+\epsilon)\lambda \in \mathcal{C}(d)$ for some $\epsilon > 0$. Then there exists a $K = K(\lambda)$ such that $\lambda \in \mathcal{C}(K,d)$.*

The idea behind the proof of Theorem 1 is as follows. By Lemma 1 we know that for any given arrival density function within the capacity region $\mathcal{C}(d)$, the system can be stabilized by a randomized region-based algorithm under K-partition and reduced reuse distance $d - 2\sqrt{2}/K$ that selects schedule $\omega \in \Omega(K, d - 2\sqrt{2}/K)$ with a certain probability $\pi(\omega)$. In order to turn this

mechanism into a scheduler that is feasible for reuse distance d, we scale the entire system by a factor h^{-1}, and by Lemma 2 we know that our randomized scheduler is now valid for reuse distance d. This is illustrated in Figure 5 for the 8-partition. Certain cells in the scaled system are located outside of the unit square, and scheduling them does not result in flows being served. However, by choosing K sufficiently large we can make this throughput loss arbitrarily small, thus stabilizing the system.

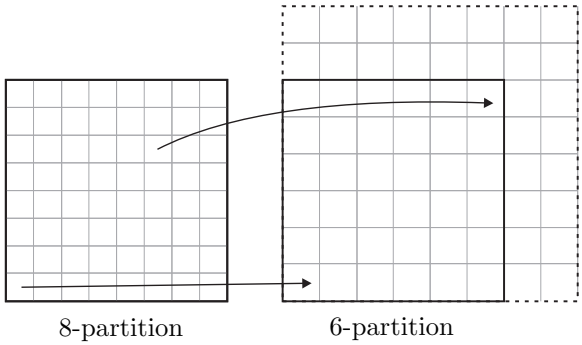

Figure 5: Constructing a 6-partition from the original 8-partition.

The proof of Theorem 1 is presented in Appendix A.3.

Theorem 1 states that every smooth arrival process can be stabilized by K-partition, by choosing the granularity of the partitioning sufficiently fine. The required value of K depends on the arrival density λ, but only through the parameters $\kappa^{(\cdot)}$. How to choose a stabilizing value of K given the $\kappa^{(\cdot)}$, and whether Theorem 1 can be extended to more general arrival densities is subject for further research.

While we have demonstrated in Theorem 1 that the capacity region $\mathcal{C}(K,d)$ of the partitioned system approaches $C(d)$ as K increases, it can be shown that they never coincide. In particular, we next establish a negative result which states that for any given K-partition, one can construct an arrival density function $\tilde{\lambda}$ such that $\tilde{\lambda} \in \mathcal{C}(d)$, but $\left(\frac{1}{2}+\epsilon\right)\tilde{\lambda} \notin \mathcal{C}(K,d)$ for any $\epsilon > 0$. In other words, *any given* K-partition may result in a 50% throughput loss for certain arrival density functions. Given a fixed K-partition, the idea behind this result is to find a set of points that can be scheduled simultaneously, but the related cells can not.

Proposition 2. *Let $K \in \mathbb{N}$, then there exists an arrival density function $\hat{\lambda}$ such that $\hat{\lambda} \in \mathcal{C}(d)$, but $\left(\frac{1}{2}+\epsilon\right)\hat{\lambda} \notin \mathcal{C}(K,d)$ for any $\epsilon > 0$.*

Proof. Assume that Kd is not an integer, and consider the set of points

$$\hat{\mathcal{P}} = \{(kG, lG) \ : \ k,l = 1,2,\ldots,\lfloor 1/G \rfloor\},$$

with $G = (d + \lceil Kd \rceil / K)/2$. We further define an arrival density function

$$\hat{\lambda}(x, y) = \sum_{(\hat{x},\hat{y}) \in \hat{\mathcal{P}}} \delta\left((x - \hat{x}, y - \hat{y})\right).$$

It is readily seen that $\hat{\mathcal{P}}$ is a feasible set of points under the original reuse constraint d, so $\hat{\lambda} \in \mathcal{C}(d)$. On the other hand, under K-partition, the point (kG, lG) belongs to cell $\mathcal{R}_{k\lceil Kd \rceil, l\lceil Kd \rceil}$. Thus the cell containing the point (kG, lG) interferes with the cell containing the point $(k'G, l'G)$ if $|k - k'| + |l - l'| \leq 1$, which implies that $\left(\frac{1}{2} + \epsilon\right) \hat{\lambda} \notin \mathcal{C}(K, d)$ for any $\epsilon > 0$. □

To illustrate Proposition 2, consider the 9-partition shown in Figure 6 and assume the reuse distance is $d = 0.35$. It is easy to verify that the set of all points shown in the figure is a feasible subset according to the original reuse constraints, but the related cells are not interference-free. For example, cell $\mathcal{R}_{1,1}$ interferes with cell $\mathcal{R}_{5,1}$. Consequently, under region-based scheduling either all black points or all gray points can be scheduled at any time, but bot both. Now assume flows of size $B \equiv 1$ uniformly arrive at the nine locations only at rate α (flows per location per time slot). A region-based scheduling strategy can only support any $\alpha \leq 1/2$, while any $\alpha < 1$ is within the network throughput region.

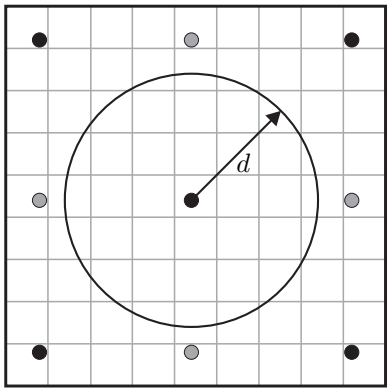

Figure 6: The 9-partition with reuse distance $d = 0.35$.

Proposition 2 establishes a negative result, in that no given region-based strategy can be expected to perform well for arbitrary spatial arrival densities. Note that the traffic pattern used in this counterexample is a discrete distribution which only injects traffic into the system at a finite number of locations. The question whether a universally stabilizing partitioning does exist when we restrict ourselves to a certain class of continuous arrival densities (e.g., smooth density functions) remains open.

5 Conclusion

We have demonstrated that MaxWeight policies may fail to provide maximum stability in the presence of flow-level dynamics due to persistent spatial inefficiency. The intuitive explanation is that these policies tend to serve flows with large backlogs, even when the resulting spatial reuse is not particularly efficient, and neglect to select maximum spatial reuse patterns involving flows with smaller backlogs.

We showed that the potential instability issues can be countered by traffic aggregation with sufficiently fine spatial granularity and adopting a region-based version of MaxWeight scheduling. A surprising fact is that the region-based approach involves a discretization with arrivals at a finite set of queues, which closely 'approximates' the arrivals in a continuum of locations as the spatial granularity increases, and yet the stability condition is markedly different. Even more remarkably, the set of admissible scheduling decisions is limited by the discretization, but the stability region for the MaxWeight strategy can be larger, i.e., constraining the set of feasible scheduling options can in fact expand the stability region. The complexity of region-based scheduling does not depend on the number of flows, in contrast to direct implementations of the algorithms in [10, 11, 21].

Finding the right granularity of the regions is non-trivial since the degree of traffic aggregation involves a trade-off between scheduling complexity, spatial efficiency, and network capacity. Determining the suitable level of aggregation without explicit knowledge of the traffic parameters remains as a challenging problem for further research.

A Remaining proofs

A.1 Proof of Lemma 1

Proof. Let $\lambda \in \mathcal{C}(d)$, $\mathcal{P} = \{P_1, P_2, \ldots, P_n\} \in \Omega(d)$ and $K \geq 2\sqrt{2}/d$. We will show that, with k_i, l_i such that $P_i \in \mathcal{R}_{k_i,l_i}$, $i = 1, \ldots, n$ it holds that

$$\{(k_1, l_1), (k_2, l_2), \ldots, (k_n, l_n)\} \in \Omega(K, d - 2\sqrt{2}/K). \tag{3}$$

That is, the points in $\mathcal{P}$ belong to a feasible set of cells under K-partition with a reduced reuse distance $d-2\sqrt{2}/K$. Consequently, any set of flows simultaneously scheduled under this strategy are located in a feasible set of cells under K-partition and reuse distance $d - 2\sqrt{2}/K$. Therefore this strategy is a legitimate region-based scheduling under K-partition with reuse distance $d-2\sqrt{2}/K$, which means $\lambda \in \mathcal{C}(K, d - 2\sqrt{2}/K)$.

We now prove (3). Let $i, j \in \{1, 2, \ldots, n\}$, $i \neq j$, and consider any two points

$Q_i \in \mathcal{R}_{k_i,l_i}$ and $Q_j \in \mathcal{R}_{k_j,l_j}$. Then

$$\begin{aligned}\|Q_i - Q_j\| =& \|Q_i - P_i + P_i - P_j + P_j - Q_j\| \\ \geq& \|P_i - P_j\| - \|P_i - Q_i\| - \|P_j - Q_j\| \\ \geq& d - 2\sqrt{2}/K.\end{aligned}$$

So no two points $Q_i \in \mathcal{R}_{k_i,l_i}$ and $Q_j \in \mathcal{R}_{k_j,l_j}$ are within distance $d - 2\sqrt{2}/K$, $i \neq j$, and thus the subset of cells $\{(k_1, l_1), (k_2, l_2), \ldots, (k_n, l_n)\}$ belongs to $\Omega(K, d - 2\sqrt{2}/K)$.

□

A.2 Proof of Lemma 2

Proof. Let $\omega \in \Omega(K, d - 2\sqrt{2}/K)$, and denote for $k, l = 1, 2, \ldots, K/h$,

$$\tilde{\mathcal{R}}_{k,l} = \left\{(x, y) \in [0, 1]^2 : (x/h, y/h) \in \mathcal{R}_{k,l}\right\},$$

the cells of size $(h/K)^2$ under (K/h)-partition. Consider two points $(x_1, y_1) \in \tilde{\mathcal{R}}_{k_1,l_1}$ and $(x_2, y_2) \in \tilde{\mathcal{R}}_{k_2,l_2}$, with $(k_1, l_1), (k_2, l_2) \in \omega$. It follows from the definition of $\Omega(K, d - 2\sqrt{K})$ that $\|(x_1/h, y_1/h) - (x_2/h, y_2/h)\| \geq d - 2\sqrt{2}/K$, which implies $\|(x_1, y_1) - (x_2, y_2)\| \geq h(d - 2\sqrt{2}/K) \geq d$, completing the proof. □

A.3 Proof of Theorem 1

Proof. Let λ be a smooth arrival density function such that $(1 + \epsilon)\lambda \in \mathcal{C}(d)$ for some $\epsilon > 0$. Lemma 1 then implies that for any $K > 2\sqrt{2}/d$, $(1 + \epsilon)\lambda \in \mathcal{C}(K, d - 2\sqrt{2}/K)$, i.e., there exists $\pi(\omega) > 0$, $\omega \in \Omega(K, d - 2\sqrt{2}/K)$ with $\sum_{\omega \in \Omega(K, d-2\sqrt{2}/K)} \pi(\omega) = 1$, such that

$$(1 + \epsilon)\lambda_{k,l}\mathbb{E}\{B\} \leq \sigma_{k,l} := \sum_{\omega \in \Omega(K, d-2\sqrt{2}/K)} \pi(\omega)\omega_{kl} \tag{4}$$

for all $k, l = 1, 2, \ldots, K$.

Now consider a randomized scheduling strategy which serves the set of cells in $\omega \in \Omega(K, d - 2\sqrt{2}/K)$ with probability $\pi(\omega)$. Let h as in (2), and denote by $\tilde{\mathcal{R}}_{k,l}$ the cells under K/h partition. By Lemma 2 we know that any set $\omega \in \Omega(K, d - 2\sqrt{2}/K)$ is valid under K/h partition and reuse distance d, and $\tilde{\mathcal{R}}_{k,l}$ is served a fraction of time $\sigma_{k,l}$.

Since the arrival density function λ is smooth, $\lambda(hx, hy)$ should be close to $\lambda(x, y)$ when h is close to 1. Specifically, it may be shown by the mean value theorem that

$$\lambda(hx, hy) \leq \lambda(x, y) + 2\kappa^{(1)}(h - 1),$$

The arrival intensity $\tilde{\lambda}_{k,l}$ of cell $\tilde{\mathcal{R}}_{k,l}$ can be bounded as

$$\begin{aligned}\tilde{\lambda}_{k,l} &= \int_{\tilde{\mathcal{R}}_{k,l}} \lambda(x,y)\,\mathrm{d}x\mathrm{d}y = h^2 \int_{\mathcal{R}_{k,l}} \lambda(hx,hy)\,\mathrm{d}x\mathrm{d}y. \\ &\leq h^2 \int_{\mathcal{R}_{k,l}} \big(\lambda(x,y) + 2\kappa^{(1)}(h-1)\big)\,\mathrm{d}x\mathrm{d}y. \end{aligned} \tag{5}$$

Now choose K large enough (and hence h small enough) such that $2\kappa^{(1)}(h-1) \leq \epsilon\kappa^{(0)}/2$ and $h^2 \leq \frac{1+\epsilon}{1+\epsilon/2}$, then

$$\begin{aligned} & h^2 \int_{\mathcal{R}_{k,l}} \big(\lambda(x,y) + 2\kappa^{(1)}(h-1)\big)\,\mathrm{d}x\mathrm{d}y \\ & \leq (1+\epsilon) \int_{\mathcal{R}_{k,l}} \lambda(x,y)\,\mathrm{d}x\mathrm{d}y = (1+\epsilon)\lambda_{k,l}. \end{aligned} \tag{6}$$

Combining (4)-(6) yields $\tilde{\lambda}_{kl}\mathbb{E}\{B\} \leq \sigma_{k,l}$ for all $k,l = 1,\ldots,K/h$, i.e., $\lambda \in \mathcal{C}(K/h,d)$. □

References

[1] D.M. Andrews, K. Kumaran, K. Ramanan, A.L. Stolyar, R. Vijayakumar, P.A. Whiting (2004). Scheduling in a queueing system with asynchronously varying service rates. *Prob. Eng. Inf. Sc.* **18**, 191–217.

[2] P. Chaporkar, S. Sarkar (2006). Stable scheduling policies for maximizing throughput in generalized constrained queueing. In: *Proc. Infocom 2006.*

[3] A. Eryilmaz, R. Srikant, J.R. Perkins (2005). Stable scheduling policies for fading wireless channels. *IEEE/ACM Trans. Netw.* **13 (2)**, 411–424.

[4] A. Eryilmaz, R. Srikant (2005). Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. In: *Proc. Infocom 2005*, 1794–1803.

[5] L. Georgiadis, M.J. Neely, L. Tassiulas (2006). Resource allocation and cross-layer control in wireless networks. *Found. Trends Netw.* **1**, 1–144.

[6] N. Kahale, P.E. Wright (1997). Dynamic global packet routing in wireless networks. In: *Proc. Infocom '97*, 1416–1423.

[7] I. Keslassy, R. Zhang-Shen, N. McKeown (2003). Maximum size matching is unstable for any packet switch. HPNG Technical Report TR03-HPNG-03010.

[8] X. Lin, N.B. Shroff (2005). The impact of imperfect scheduling on cross-layer rate control in wireless networks. In: *Proc. Infocom 2005*, 1804–1814.

[9] X. Lin, N.B. Shroff, R. Srikant (2008). On the connection-level stability of congestion-controlled communication networks. *IEEE Trans. Inf. Theory* **54**, 2317–2338.

[10] S. Liu, L. Ying, R. Srikant (2010). Throughput-optimal opportunistic scheduling in the presence of flow-level dynamics. In: *Proc. Infocom 2010*,

[11] S. Liu, L. Ying, R. Srikant (2010). Scheduling in multichannel wireless networks with flow-level dynamics. In: *Proc. ACM SIGMETRICS 2010*, 191–202.

[12] M.G. Markakis, E.H. Modiano, J.N. Tsitsiklis (2009). Scheduling policies for single-hop networks with heavy-tailed traffic. In: *Proc. 47th Annual Allerton Conf. Commun., Control, Comp.*, Monticello IL.

[13] N. McKeown, V. Anantharam, J.C. Walrand (1996). Achieving 100% throughput in an input-queued switch. In: *Proc. Infocom '96*, 296–302.

[14] N. McKeown, A. Mekkittikul, V. Anantharam, J.C. Walrand (1999). Achieving 100% throughput in an input-queued switch. *IEEE Trans. Commun.* **47 (8)**, 1260–1267.

[15] S.P. Meyn (2007). *Control Techniques for Complex Networks*, Cambridge University Press.

[16] C. Moallemi, D. Shah (2010). On the flow-level dynamics of a packet-switched network. In: *Proc. ACM SIGMETRICS 2010*, 83–94.

[17] M.J. Neely (2005). Energy optimal control for time-varying wireless networks. In: *Proc. Infocom 2005*, 572–583.

[18] M.J. Neely, E. Modiano, C.-P. Li (2005). Fairness and optimal stochastic control for heterogeneous networks. In: *Proc. Infocom 2005*, 396–409.

[19] M.J. Neely, E. Modiano, C.E. Rohrs (2005). Dynamic power allocation and routing for time-varying wireless networks. *IEEE J. Sel. Areas Commun.* **23**, 89–103.

[20] A. Proutière, Y. Yi, M. Chiang (2008). Throughput of random access without message passing. In: *Proc. CISS 2008*

[21] B. Sadiq, G. de Veciana (2009). Throughput optimality of delay-driven MaxWeight scheduler for a wireless system with flow dynamics. In: *Proc. 47th Annual Allerton Conf. Commun., Control, Comp.*.

[22] G. Sharma, R.R. Mazumdar, N.B. Shroff (2006). On the complexity of scheduling in wireless networks. In: *Proc. MobiCom '06*, 227–238.

[23] G. Sharma, N.B. Shroff, R.R. Mazumdar (2007). Joint congestion control and distributed scheduling for throughput guarantees in wireless networks. In: *Proc. Infocom 2007*, 2072–2080.

[24] A.L. Stolyar (2004). MaxWeight scheduling in a generalized switch: state space collapse and workload minimization in heavy traffic. *Ann. Appl. Prob.* **14**, 1–53.

[25] A.L. Stolyar (2005). Maximizing queueing network utility subject to stability: greedy primal dual algorithm. *Queueing Systems* **50**, 401–457.

[26] A.L. Stolyar (2006). Greedy primal-dual algorithm for dynamic resource allocation in complex networks. *Queueing Systems* **54**, 203–220.

[27] L. Tassiulas (1998). Linear complexity algorithms for maximum throughput in radio networks and input queued switches. In: *Proc. Infocom '98*, 533–539.

[28] L. Tassiulas, A. Ephremides (1992). Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. Aut. Contr.* **37**, 1936–1948.

[29] L. Tassiulas, A. Ephremides (1993). Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Trans. Inf. Theory* **39**, 466–478.

[30] P.M. van de Ven, S.C. Borst, V. Shneer (2009). Instability of MaxWeight scheduling algorithms. In: *Proc. Infocom 2009*, 1701–1709.

[31] X. Wu, R. Srikant (2006). Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks. In: *Proc. Infocom 2006*.