Computer Communications 36 (2013) 1350–1359

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Inefficiency of MaxWeight scheduling in spatial wireless networks ☆

P.M. van de Ven [a,*], S.C. Borst [b,c], L. Ying [d]

[a] IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA
[b] Eindhoven University of Technology, Department of Mathematics and Computer Science, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[c] Bell Laboratories, Alcatel-Lucent, Murray Hill, NJ 07974, USA
[d] School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287, USA

ARTICLE INFO

Article history:
Available online 15 November 2012

Keywords:
Capacity region
Flow-level dynamics
MaxWeight scheduling
Stability
Wireless networks

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 and Ephremides [1] 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 and Wright [2] for instance. In subsequent work, Tassiulas and Ephremides [3] 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 for instance Andrews et al. [4], Eryilmaz et al. [5], Neely [6] and Neely et al. [7]. Extending the scope further, Eryilmaz and Srikant [8], Neely et al. [9] and Stolyar [10,11] 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. [12] 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 [16].

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 interference conditions for example. 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

☆ This work was supported in part by the NSF Grants CNS-1264012 and CNS-126232 and the DTRA Grants HDTRA1-08-1-0016 and HDTRA1-09-1-0055.

* Corresponding author.
E-mail addresses: pmvandev@us.ibm.com (P.M. van de Ven), s.c.borst@tue.nl (S.C. Borst), lei.ying.2@asu.edu (L. Ying).

0140-3664/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comcom.2012.11.001

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 [17], Eryilmaz et al. [5] and Chaporkar and Sarkar [18] 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 and Shroff [19], Sharma et al. [20,21] and Wu and Srikant [22].

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, [23] 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 [23] 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 [24,25] proposed more practical scheduling algorithms guaranteeing throughput optimality, and extended these to multi-channel scenarios. Sadiq and De Veciana [26] 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. [27].

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 [1] 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 [23].

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 [19–22]. It is further worth drawing a distinction with the work of Lin et al. [28] and Moallemi and Shah [29] 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 rather than at the level of individual flows.

While the fundamental cause for instability observed in this paper (flow-level dynamics) is the same as in [23], the way the presence of transient flows unhinges the MaxWeight scheduling algorithm is completely different. Consequently, the remedies for instability discussed in [24–26] 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.

This 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 distribution. Section 5 offers some concluding remarks.

A preliminary version of this paper has appeared as [30]. Compared to this earlier version, we have extended the results on the stability properties of region-based scheduling to discrete distributions (Proposition 2). In addition, we illustrate in Section 3 that the MaxWeight scheduling algorithm may become unstable for much

more general network topologies and traffic arrival patterns than previously discussed. Using simulations, we evaluate in Section 4 the impact of the level of aggregation on the delay and stability of region-based MaxWeight scheduling.

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 of that flow will arrive. Most of the results can be extended to a setting with gradual traffic, where traffic of a flow arrives over time.

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 $\|P_i - P_j\| > d$ 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.

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.,

$$\sum_{j \in J^*(t)} Q_j(t) = \max_{J \subseteq I(t), F(J)=1} \sum_{j \in J} Q_j(t).$$

The main reason for assuming unit transmission rates is to stress the fact that the instability phenomena demonstrated in the next section result from persistent spatial inefficiency rather than rate heterogeneity. Possible rate heterogeneity induces priorities among flows, which may exacerbate the spatial inefficiency and render the system even more prone to potential instability effects. Note for example that scaling both the transmission rates and sizes in proportion by class-dependent factors will induce priorities among flows, without fundamentally altering the actual resource requirements. Even in a single-node work-conserving system, where stability is not an issue, high-rate high-volume flows will receive preferential treatment, and low-rate low-volume flows will suffer poor performance. In non-work-conserving network scenarios, such priorities can degrade the spatial reuse and ultimately cause instability.

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 star topology with $N \geqslant 3$ regions, as is shown in Fig. 1a. Transmissions in the outer regions $1, \ldots, N-1$ all interfere with the transmission in the inner region N, but not with each other. Flows arrive in region i at a rate λ_i (per time slot) and have i.i.d. sizes B_i. The numbers of arriving flows in successive time slots is assumed to be i.i.d. Denote by $\rho_i = \lambda_i \mathbb{E}B_i$ the traffic intensity in region i. We assume that

$$\rho_i + \rho_N < 1, \quad i = 1, \ldots, N-1,$$

or equivalently,

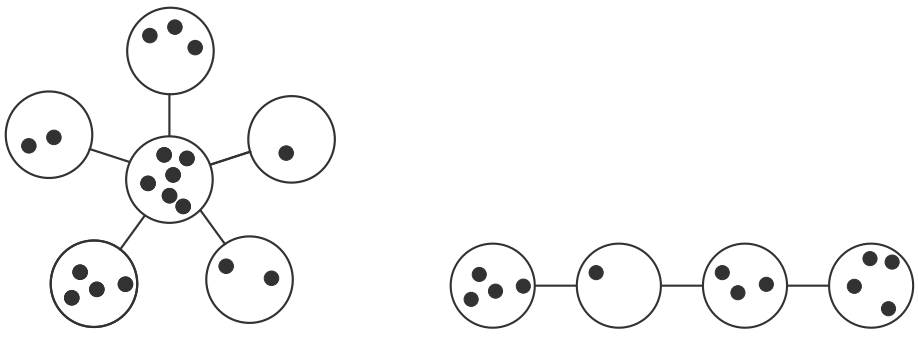
$$\rho_N < 1 - \max_{i=1,\ldots,N-1} \rho_i. \tag{1}$$

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 (if available) in region N with probability $\rho_N + \epsilon$, or schedule a flow (if available) in all outer regions with probability $\max_{i=1,\ldots,N-1}\rho_i + \epsilon$, where $\epsilon = \frac{1-\rho_N-\max_{i=1,\ldots,N-1}\rho_i}{2}$.

Now suppose $B_N \equiv 1$, 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 never schedules a flow in the inner region as long as a flow with a residual size of 2 or larger is present in any of the outer regions. Hence the time periods during which a flow of residual size 2 or larger gets scheduled at the various outer regions are independent. Also, the fraction of time that a flow of residual size 2 or larger gets scheduled at outer 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 the inner region N gets scheduled, is bounded from above by

$$\prod_{i=1}^{N-1}(1 - \rho_i + \lambda_i).$$

Thus a necessary condition for MaxWeight scheduling to achieve stability is



(a) Star topology (b) Linear topology

Fig. 1. Two examples of spatial wireless networks where MaxWeight scheduling is not throughput-optimal.

$$\rho_N \leqslant \prod_{i=1}^{N-1}(1-\rho_i+\lambda_i).$$

When the λ_i's are small and the $\mathbb{E}B_i$'s of the outer regions are large, the latter condition 'approaches' $\rho_N \leqslant \prod_{i=1}^{N-1}(1-\rho_i)$, which is a more stringent inequality than the sufficient condition (1).

In Example 1, the stabilizing strategy either schedules all outer regions, or schedules region N. The MaxWeight policy, however, tends to serve flows with large backlogs, so flows in the outer regions are served with priority when their residual sizes are greater than or equal to 2. Consequently, the MaxWeight policy only schedules the inner region if all outer regions are empty, which leads to inefficient spatial reuse.

In Example 2 we argue that the instability persists even if the size of arriving flows is the same for all regions.

Example 2. Consider again the star topology of Example 1, but now suppose $B_i \equiv 1$ for all $i = 1, \ldots, N$. We assume that

$$\lambda_i + \lambda_N < 1, \quad i = 1, \ldots, N-1,$$

or equivalently,

$$\lambda_N < 1 - \max_{i=1,\ldots,N-1} \lambda_i. \tag{2}$$

As before, this condition is necessary for stability to be achievable as well as sufficient under mild independence assumptions. With unit flow sizes, MaxWeight scheduling reduces to maximum-size scheduling, i.e., selecting an independent set of maximum size. Only when at most one of the outer regions has active flows, a flow in the inner region can get scheduled, depending on the tie-breaking rule. Let p be the probability that a tie is broken in favor of the inner region when only one of the outer regions has active flows. Proutière et al. [31] proved that for Bernoulli arrivals maximum-size scheduling fails to achieve stability in case

$$\lambda_N > \frac{\Pi + p\sum_{i=1}^{N-1} \frac{\lambda_i \Pi}{1-\lambda_i - p\Pi_i}}{1 + p\sum_{i=1}^{N-1} \frac{\lambda_i \Pi_i}{1-\lambda_i - p\Pi_i}}, \tag{3}$$

with $\Pi = \prod_{j=1}^{N-1}(1-\lambda_j)$ and $\Pi_i = \prod_{j\neq i,N}(1-\lambda_j) = \Pi/(1-\lambda_i)$. It may be checked that the condition (3) is again more stringent than the sufficient condition (2), except in the case $p = 1$ and $N = 3$. Similar instability issues for input-queued switches were demonstrated in [32,14].

In Example 2 we have demonstrated instability in a setting where all flows have initial size 1, and only require to be scheduled once. We next present an example where MaxWeight scheduling is unstable when all flows require multiple rounds of service.

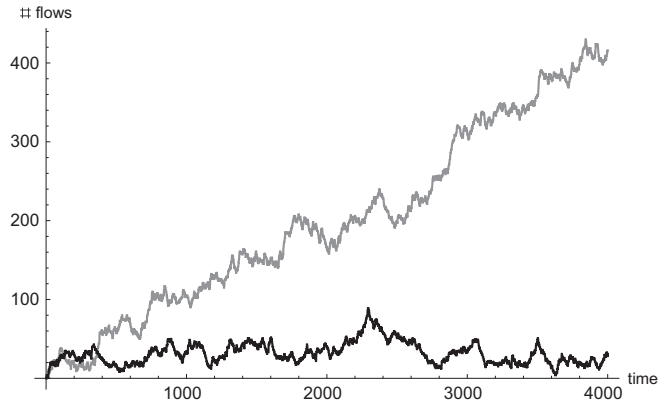


Fig. 2. Evolution of the total number of flows over time for flow-based MaxWeight scheduling (gray) and a region-based cyclic strategy (black), respectively.

Example 3. Consider the 'linear' topology with $N = 4$ regions shown in Fig. 1b. We have interference between regions 1 and 2, 2 and 3, and 3 and 4, so the maximal feasible schedules are $\{1,3\}$, $\{1,4\}$, and $\{2,4\}$. All flows are assumed to have size $B = 2$. As before, the MaxWeight scheduling strategy selects a subset of flows of maximum aggregate residual size, subject to the interference constraints.

Denote by $A_i(t)$ the number of flows arriving in region i in time slot t. We consider the following deterministic arrival pattern:

- at time slots $t = 9T$ and $t = 9T+4$ $(T = 0, 1, 2, \ldots)$, $A_1(t) = A_4(t) = 1$ and $A_2(t) = A_3(t) = 0$;
- at time slots $t = 9T+2$ and $t = 9T+7$ $(T = 0, 1, 2, \ldots)$, $A_2(t) = A_3(t) = 1$ and $A_1(t) = A_4(t) = 0$;
- at all other time slots, $A_1(t) = A_2(t) = A_3(t) = A_4(t) = 0$.

Lemma 1. *Consider MaxWeight scheduling and the deterministic arrival pattern described above. At the start of time slots $t = 9T$ and $t = 9T+4$ (before the arrival of new flows), all old flows have residual size 1, and regions 1 and 4 are empty, $T = 0, 1, \ldots$.*

The proof of Lemma 1 is provided in Section 6.1, and relies on the fact that the arrival pattern is such that in each slot, the maximal independent set is uniquely determined.

Lemma 1 implies that in time slots $t = 9T$ and $t = 9T+4$ schedule $\{1,4\}$ is used to serve the two newly arrived flows in regions 1 and 4. This consumes a fraction 2/9 of the time, and only leaves a fraction 7/9 of the time available to serve flows in regions 2 and 3. Now observe that $\rho_2 = \rho_3 = 4/9$ and that flows in regions 2 and 3 cannot be scheduled simultaneously. Hence, in order to stabilize regions 2 and 3, a fraction 8/9 of the time is required to serve flows in these two regions. It follows that that regions 2 and 3 cannot both be stable.

On the other hand, a strategy that alternatingly uses schedules $\{1,3\}$ and $\{2,4\}$, and thus allows each of the regions to be served half of the time, is easily seen to achieve stability.

The flow arrivals in the various regions in Example 3 are strongly correlated. The correlations are specifically constructed to yield tractable behavior and provably demonstrate instability. When the arrival processes are independent, the behavior is more complex, and stability is more difficult to determine. Hence we conducted a simulation experiment as presented below, suggesting that the instability issues also occur for independent arrival processes.

Example 4. We consider the network discussed in Example 3, and assume that the numbers of arriving flows at the various regions are independent geometrically distributed random variables with parameter $p = 0.81$, so $\lambda_i = \frac{1-p}{p} \approx 0.234$, and $\rho_i = \lambda_i \mathbb{E}B \approx 0.469 < 1/2$, $i = 1, 2, 3, 4$. Although we restrict ourselves to geometric arrivals here and in Example 6, the observations are expected to hold for any random arrival process.Fig. 2 shows the total number of flows present over time. We let the simulation run for 4000 slots. The gray line depicts a sample path for flow-based MaxWeight scheduling, and the black line corresponds to a cyclic strategy that alternates between schedules $\{1,3\}$ and $\{2,4\}$. Under flow-based scheduling, the total number of flows grows at a roughly linear rate, suggesting instability. In contrast, for the cyclic strategy, the number of flows fluctuates, but remains at fairly low levels, and indeed the system is clearly stable for the given load. Thus flow-based MaxWeight scheduling fails to achieve maximum stability.

Examples 1–4 illustrate the spatial inefficiency of MaxWeight scheduling by carefully constructing the regions where flows arrive. Next we present a further example where we consider a

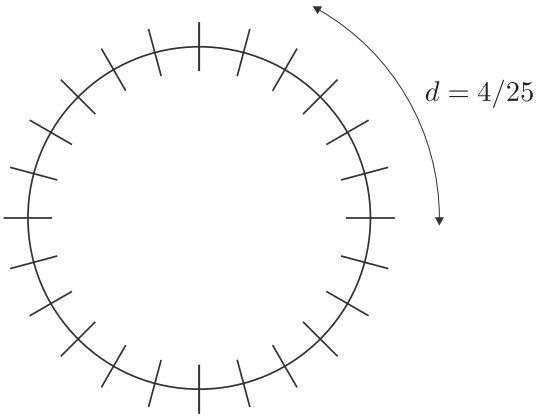


Fig. 3. A ring with unit circumference, reuse distance $d = 4/25$, partitioned into 25 intervals of equal size ($N = 1$).

one-dimensional space (a ring) with uniformly distributed arrival locations. We assume that all flows are of the same size, and show that even in this *uniform* traffic scenario, MaxWeight scheduling fails to achieve throughput optimality.

Example 5. Let $N \geqslant 1$ 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 Fig. 3. 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 $1 - a$.

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 $M(N) = \left\lfloor \frac{(2N+3)(3N+2)}{2(N+1)} \right\rfloor$, and the total traffic intensity equals $\rho = 2a(2N+3)$, so the necessary condition $\rho < M$ for stability takes the form

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

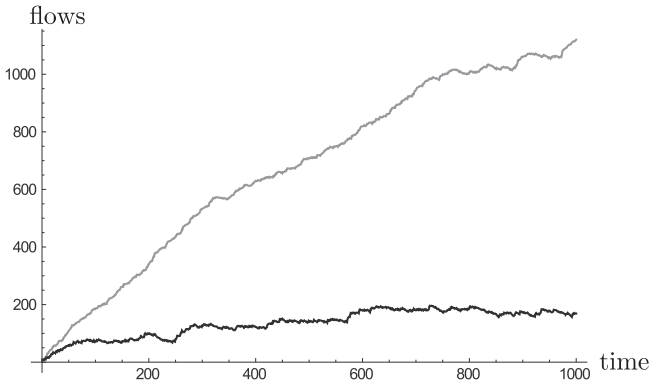


Fig. 4. Evolution of the total number of flows for MaxWeight scheduling (gray) and an interval-based randomized scheduler (black), respectively.

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 $a^{MW}(N) \leqslant a(N)$, with strict inequality for all $N \geqslant 2$, 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 subset that allowed for better spatial reuse. This persistent inefficiency then leads to instability. As we see below, this behavior occurs for more general traffic patterns as well.

The locations of arriving flows in Example 5 are uniformly distributed, but highly dependent. When the flow locations are independent, the behavior is more complex, and (in)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 6. Consider a ring network where the total number of arriving flows is independent between slots and is geometrically distributed with parameter $p = 0.45$ and mean $\alpha = \frac{1-p}{p} \approx 11/9$, so $\rho = \alpha \mathbb{E}B \approx 22/9$. We assume that the flow sizes are i.i.d., and 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 for 1000 slots, for both MaxWeight scheduling and the randomized strategy. Fig. 4 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 without bound, 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 rate [24,25] or according to the product of feasible transmission rate and the delay [26]. 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 distribution. 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. In the case that the spatial traffic distribution is absolutely continuous, the location of an arbitrary arriving flow is governed by some spatial arrival density function λ on $[0,1]^2$. We have $\lambda(x,y)=0$ 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 $\int_{(x,y)\in\mathcal{R}}\lambda(x,y)\,\mathrm{d}x\mathrm{d}y$. For discrete spatial traffic distributions we define m rates $\lambda_1,\ldots,\lambda_m$ and m locations $(x_1,y_1),\ldots,(x_m,y_m)\in\mathcal{S}$ such that flows arrive at (x_i,y_i) with rate $\lambda_i, i=1,\ldots,m$. We allow for $m=\infty$, and the expected number of arriving flows per time slot in region $\mathcal{R}\subseteq\mathcal{S}$ is $\sum_{(x_i,y_i)\in\mathcal{R}}\lambda_i$. Let the positive random variable B represent the size of an arbitrary flow.

4.1. A few special cases

The above general network setting includes the star network, the linear 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 maximally stable policies for these three special cases. For the star network, an alternative view of the network is to consider each region as a single node. The star network can then be seen as a classic N-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. The same reasoning can be applied to the linear network.

Now consider the ring network with uniform spatial traffic density α. Taking a similar approach as for the region-based networks, instead of scheduling flows, we divide the ring into intervals and schedule these intervals instead. As we 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 for any load $\rho=\alpha\mathbb{E}B<M$ there exists an interval-based scheduling strategy that achieves stability for any load.*

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 $\widehat{P}_i\in[P_i,P_i+\epsilon)$, the set $\widehat{\mathcal{P}}=\{\widehat{P}_1,\ldots,\widehat{P}_M\}$ is feasible as well. We choose integers $K_i,\ i=1,2,\ldots,M$ and K 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 5 and 6 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 can be readily verified that given the reuse distance d, the necessary condition for ρ to be supportable is $\rho\leqslant\lfloor 1/d\rfloor$ and that the scheduling algorithm presented in the proof can stabilize any $\rho<\lfloor 1/d\rfloor$.

4.2. General networks

The examples in Section 4.1 suggest that in the presence of flow-level dynamics, it is beneficial to 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 regions and then schedules a flow in each selected region, if available.

Naturally, such a partitioning would reduce the flexibility of the scheduler since the regions must be chosen such that the resulting schedule is always feasible, irrespective of the exact location of the flows inside the regions. Region-based scheduling is nevertheless interesting 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 stringent feasibility constraints of the partitioned system. Since our scheduling decision is based only on the aggregate backlog, which flow to select within a region does not affect the performance of a scheduler. We are interested in how the capacity region of the partitioned system compares to the capacity region under the original reuse constraints. As we will see, this depends on the granularity of the partitioning. Note that, in contrast to flow-based scheduling, 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 Fig. 5. 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 $\Omega(K,d)$ 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 similar qualitative results hold for other types of regions.

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 the set of points $\mathcal{P}=\{P_1,P_2,\ldots,P_n\}$ is feasible for any $P_i\in\mathcal{R}_{k_i,l_i},\ i=1,\ldots,n$. So under K-partition, scheduling is con-

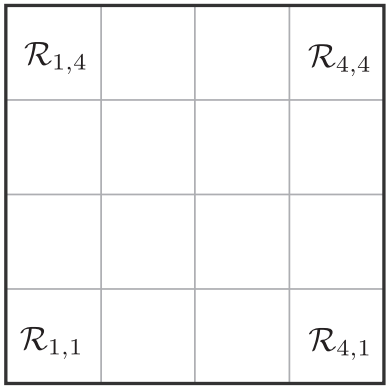


Fig. 5. The 4-partition, where the unit square is divided into 16 cells.

fined to subsets of flows that belong to a feasible subset of cells, which restricts beyond the original reuse constraint and guarantees feasibility. The capacity region for such a system is well-known:

$$\mathcal{C}(K,d) = \{\lambda : \ \lambda_{k,l}\mathbb{E}B \in \text{conv.hull}(\Omega(K,d))\}, \tag{4}$$

where $\lambda_{k,l}$ denotes the aggregate arrival rate of flows in the cell $\mathcal{R}_{k,l}$. Let $\mathcal{C}(d)$ denote the capacity region under the original reuse constraint, then $\mathcal{C}(K,d) \subseteq \mathcal{C}(d)$ for any $K \geqslant 1$. 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 Theorem 1, which states our main result, showing that for any arrival distribution $\lambda \in C(d)$ (under certain assumptions) there exists a K such that λ is contained in $\mathcal{C}(K,d)$.

Before stating Theorem 1, we first present the following lemmas.

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

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

The proof of Lemma 2 is presented in Section 6.2.

Let $\omega \in \Omega(K,d)$, $L \leqslant K$, 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 3. *Let $d > 0,\ K \in \mathbb{N}$ and set*

$$h = K\left\lfloor \frac{K(d - 2\sqrt{2}/K)}{d} \right\rfloor^{-1}. \tag{5}$$

Then $\omega \in \Omega(K, d - 2\sqrt{2}/K) \Rightarrow \omega^{(K/h)} \in \Omega(K/h, d)$.

Lemma 3 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 Section 6.3.

We are now in a position to state and prove Theorem 1. 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 $\lambda(x,y) \geqslant \kappa^{(0)}$ for all $(x,y) \in \mathcal{S}$;

–differentiable, with a uniformly upper bounded first-order partial derivative, i.e., there exists $\kappa^{(1)}, \kappa^{(2)} < \infty$ such that $\frac{\partial\lambda(x,y)}{\partial x} \leqslant \kappa^{(1)}$ and $\frac{\partial\lambda(x,y)}{\partial y} \leqslant \kappa^{(1)}$ for all $(x,y) \in \mathcal{S}$ and $\frac{\partial^2\lambda(x,y)}{\partial x\partial y} < \kappa^{(2)}$.

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 proof of Theorem 1 is presented in Section 6.4, and the idea behind the proof is as follows. By Lemma 2 we know that any given arrival density function within the capacity region $\mathcal{C}(d)$ can be stabilized by a region-based algorithm under K-partition and reduced reuse distance $d - 2\sqrt{2}/K$. 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 3 we know that our scheduler is now valid for reuse distance d. This is illustrated in Fig. 6 for the 8-partition. Certain cells in the scaled system are located outside 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.

Theorem 1 states that every smooth arrival process can be stabilized by K-partition for sufficiently fine granularity of the partitioning. The required value of K depends on the arrival density λ, but only through the parameters $\kappa^{(\cdot)}$. The smoothness is largely a technical condition, and in Proposition 2 we will show that K-partition also stabilizes the system for arbitrary discrete arrival distributions.

Proposition 2. *Let λ be a discrete arrival distribution 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 proof of Proposition 2 is presented in Section 6.5. Combining Theorem 1 and Proposition 2 suggests that K-partition can stabilize any arrival distribution function, although this remains to be shown formally.

While we have demonstrated in Theorem 1 and Proposition 2 that the capacity region $\mathcal{C}(K,d)$ of the partitioned system approaches $C(d)$ in a certain sense 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 function $\tilde{\lambda}$ such that $\tilde{\lambda} \in \mathcal{C}(d)$, but $(\frac{1}{2}+\epsilon)\tilde{\lambda} \notin \mathcal{C}(K,d)$ for any $\epsilon > 0$. In other words, *any given* K-partition may result in a 50% throughput loss for certain arrival distributions. 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 3. *Let $K \in \mathbb{N}$, then there exists a discrete probability mass function $\hat{\lambda}$ such that $\hat{\lambda} \in \mathcal{C}(d)$, but $(\frac{1}{2}+\epsilon)\hat{\lambda} \notin \mathcal{C}(K,d)$ 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 assume that flows arrive uniformly across these points, and that the expected number of arriving flows is given by $\Lambda = |\hat{\mathcal{P}}|$.

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 $|k-k'| + |l-l'| \leqslant 1$, which implies that $(\frac{1}{2}+\epsilon)\hat{\lambda} \notin \mathcal{C}(K,d)$ for any $\epsilon > 0$. □

To illustrate Proposition 3, consider the 9-partition shown in Fig. 7 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 not both. Now assume flows of size $B \equiv 1$ uniformly arrive at the nine loca-

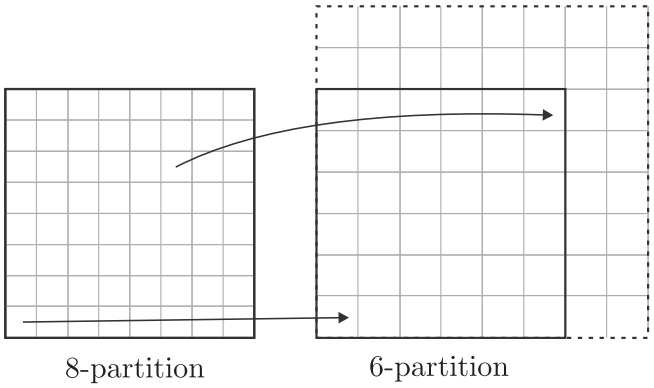


Fig. 6. Constructing a 6-partition from the original 8-partition.

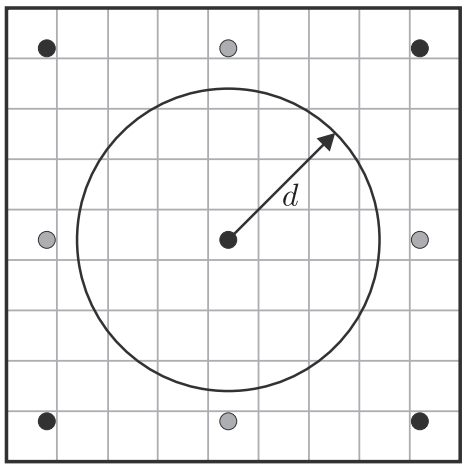


Fig. 7. The 9-partition with reuse distance $d = 0.35$.

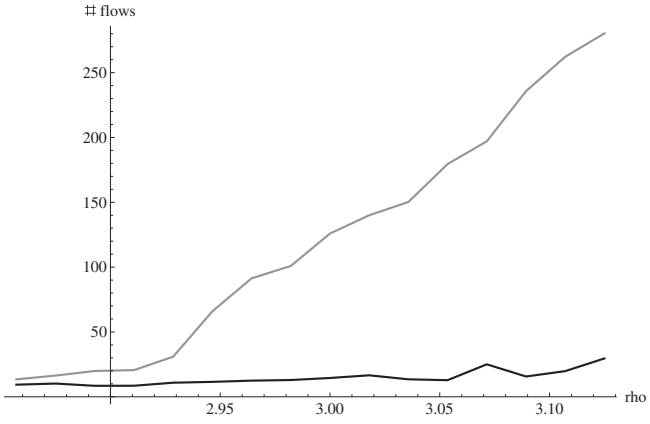


Fig. 8. The impact of K.

tions only at rate α (flows per location per time slot). A region-based scheduling strategy can at most support any $\alpha \leqslant 1/2$, while any $\alpha < 1$ is within the network capacity region, by scheduling all locations simultaneously.

Proposition 3 establishes a negative result, in that no given region-based strategy can be expected to perform well for arbitrary spatial arrival distributions. 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.

Both Theorem 1 and Proposition 2 state that their respective arrival functions can be stabilized by choosing a sufficiently large value of K; how to choose a stabilizing value of K is subject for further research. This is illustrated in Fig. 8, which shows the number of flows present after 5000 time slots for both $K = 10$ (gray) and $K = 40$ (black), in a ring network with $d = 0.3$. The graph for $K = 10$ explodes before $K = 40$, indicating that $K = 40$ provides better stability.

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. Loosely stated, MaxWeight policies tend to serve flows with large backlogs, even when the resulting spatial reuse is not particularly efficient, and fail to take advantage of 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 of a specific scheduler. The complexity of region-based scheduling does not depend on the number of flows, in contrast to direct implementations of the algorithms in [24–26].

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. In particular, the suitable level of aggregation depends on the specific 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.

6. Additional proofs

6.1. Proof of Lemma 1

Proof. Let t_- denote the start of time slot t, just before the arrival of new flows. We determine the states of the system at $t = 0, \ldots, 4$ as follows:

1. At $t = 0_-$: $N_i(0_-) = 0$ for all $i = 1, 2, 3, 4$, so the claim holds for $t = 0$.
2. At time slot $t = 0 : N_1(0) = N_4(0) = 1$ and $N_2(0) = N_3(0) = 0$, so the schedule $\{1, 4\}$ is used.
3. At time slot $t = 1$: no new flows arrive, so $N_2(1) = N_3(1) = 1$ and all old flows have residual size 1. Thus the schedule $\{1, 4\}$ is used again.
4. At time slot $t = 2_-$: $N_i(2_-) = 0$ for all $i = 1, 2, 3, 4$.
5. At time slot $t = 2 : N_2(2) = N_3(2) = 1$ and $N_1(2) = N_4(2) = 0$, so either node 2 or node 3 gets scheduled.
6. At time slot $t = 3$: If node 2 got scheduled in the previous time slot, then now node 3 gets scheduled; otherwise, node 2 gets scheduled.
7. At time slot $t = 4_-$: $N_2(4_-) = N_3(4_-) = 1$ and $N_1(4_-) = N_4(4_-) = 0$. All flows in the system are of size 1.

In conclusion, the claim holds for $K = 0$. Now assume the claim is valid for $K = k$ and consider $K = k + 1$.

1. At time slot $t = (9k + 4)_-$: $N_1((9k + 4)_-) = N_4((9k + 4)_-) = 0$ and all flows in the system are of size 1.
2. At time slot $t = 9k + 4$: The two new flows arriving at nodes 1 and 4 get scheduled.
3. At time slot $t = (9k + 5)_-$: $N_1((9k + 5)_-) = N_4((9k + 5)_-) = 1$, $N_2((9k + 5)_-) = N_3((9k + 5)_-) > 0$, and all flows in the system are of size 1. Then one of the schedules $\{1, 3\}, \{1, 4\}$ and $\{2, 4\}$ is used, so at least one of the nodes 1 and 4 gets scheduled.
4. At time slot $t = 9k + 6$: The selection of the schedule depends on the schedule used in the previous time slot. However, it can be easily checked that after time slot $t = 9k + 6$, we have $N_1((9k + 7)_-) = N_4((9k + 7)_-) = 0$.

5. At time slot $t = 9k+7$ and $t = 9k+8$: The two new flows arriving at time slot $t = 9k+7$ get scheduled once each.
6. At time slot $t = 9(k+1)_-$: $N_1(9(k+1)_-) = N_4(9(k+1)_-) = 0$, and all flows in the system are of size 1.
7. At time slot $t = 9(k+1)$: The two new flows arriving at nodes 1 and 4 get scheduled; $N_1(9(k+1)) = N_4(9(k+1)) = 1$.
8. At time slot $t = 9(k+1)+1$: One of the schedules $\{1,3\}$, $\{1,4\}$ and $\{2,4\}$ is used, and all flows in the system are of size 1; $N_1(9(k+1)) \leqslant 1$ and $N_4(9(k+1)) \leqslant 1$.
9. At time slot $t = 9(k+1)+2$ and $t = 9(k+1)+3$: The two schedules $\{1,3\}$ and $\{2,4\}$ are used.
10. At time slot $t = (9(k+1)+4)_-$: $N_1((9(k+1)+4)_-) = N_4((9(k+1)+4)_-) = 0$ and all flows in the system are of size 1.

Hence, the claim holds for $K = k+1$, and it follows by induction that the claim is valid for all K. □

6.2. Proof of Lemma 2

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

$$\{(k_1,l_1),(k_2,l_2),\ldots,(k_n,l_n)\} \in \Omega(K, d - 2\sqrt{2}/K). \tag{6}$$

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 the original reuse constraints are located in a feasible set of cells under K-partition and reuse distance $d - 2\sqrt{2}/K$. Therefore any feasible strategy is a legitimate region-based scheduling under K-partition with reuse distance $d - 2\sqrt{2}/K$, which implies $\mathcal{C}(d) \subseteq \mathcal{C}(K, d - 2\sqrt{2}/K)$.

We now prove (6). 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

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

As a result 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 $\Omega(K, d - 2\sqrt{2}/K)$. □

6.3. Proof of Lemma 3

Proof. Let $\omega \in \Omega(K, d - 2\sqrt{2}/K)$, and denote

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

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 $\|(x_1/h, y_1/h) - (x_2/h, y_2/h)\| \geqslant d - 2\sqrt{2}/K$, which implies $\|(x_1,y_1) - (x_2,y_2)\| \geqslant h(d - 2\sqrt{2}/K) \geqslant d$, completing the proof. □

6.4. 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 2 then implies that $(1+\epsilon)\lambda \in \mathcal{C}(K, d - 2\sqrt{2}/K)$ for any $K > 2\sqrt{2}/d$. By the characterization in (4) we know that there exist $\alpha(\omega) > 0$, $\omega \in \Omega(K, d - 2\sqrt{2}/K)$ with $\sum_{\omega \in \Omega(K, d-2\sqrt{2}/K)} \alpha(\omega) = 1$, such that

$$(1+\epsilon)\lambda_{k,l}\mathbb{E}B \leqslant \sigma_{k,l} = \sum_{\omega \in \Omega(K, d-2\sqrt{2}/K)} \alpha(\omega)\omega_{kl}, \quad k,l = 1,2,\ldots,K. \tag{7}$$

Now consider a randomized scheduling strategy which selects the schedule $\omega \in \Omega(K, d - 2\sqrt{2}/K)$ with probability $\alpha(\omega)$. Let h as in (5), and denote by $\tilde{\mathcal{R}}_{k,l}$ the cells under K/h partition. By Lemma 3 we know that for any schedule $\omega \in \Omega(K, d - 2\sqrt{2}/K)$, $\omega^{(K/h)}$ is feasible under K/h partition and reuse distance d. The region $\tilde{\mathcal{R}}_{k,l}$ is served a fraction of time $\sigma_{k,l}$ under the above randomized scheduler.

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

$$\begin{aligned}\lambda(hx,hy) &= \lambda(x,y) + \int_x^{hx}\int_y^{hy} \frac{\partial^2\lambda(x,y)}{\partial x \partial y}\,\mathrm{d}x\,\mathrm{d}y \\ &\leqslant \lambda(x,y) + \kappa^{(2)}\int_x^{hx}\int_y^{hy} \mathrm{d}x\,\mathrm{d}y \\ &= \lambda(x,y) + \kappa^{(2)}\|(hx,hy) - (x,y)\| \\ &\leqslant \lambda(x,y) + \kappa^{(2)}\sqrt{(hx-x)^2 + (hy-y)^2} \\ &= \lambda(x,y) + \kappa^{(2)}(h-1)\sqrt{x^2+y^2} \leqslant \lambda(x,y) + \sqrt{2}\kappa^{(2)}(h-1),\end{aligned}$$

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

$$\begin{aligned}\tilde{\lambda}_{k,l} &= \int_{\tilde{\mathcal{R}}_{k,l}} \lambda(x,y)\,\mathrm{d}x\,\mathrm{d}y = h^2\int_{\mathcal{R}_{k,l}} \lambda(hx,hy)\,\mathrm{d}x\,\mathrm{d}y \\ &\leqslant h^2\int_{\mathcal{R}_{k,l}} \left(\lambda(x,y) + 2\kappa^{(1)}(h-1)\right)\mathrm{d}x\,\mathrm{d}y.\end{aligned} \tag{8}$$

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

$$\begin{aligned}h^2\int_{\mathcal{R}_{k,l}} \left(\lambda(x,y) + 2\kappa^{(1)}(h-1)\right)\mathrm{d}x\,\mathrm{d}y &\leqslant (1+\epsilon)\int_{\mathcal{R}_{k,l}} \lambda(x,y)\,\mathrm{d}x\,\mathrm{d}y \\ &= (1+\epsilon)\lambda_{k,l}.\end{aligned} \tag{9}$$

Combining (7)–(9) yields $\tilde{\lambda}_{kl}\mathbb{E}\{B\} \leqslant \sigma_{k,l}$ for all $k,l = 1,\ldots,K/h$, i.e., $\lambda \in \mathcal{C}(K/h, d)$. □

6.5. Proof of Proposition 2

Proof. Since λ is a discrete arrival distribution, there exist locations $(x_i,y_i) \in [0,1]^2, \lambda_i' > 0$ such that $\sum_i \lambda_i' = 1$ and some constant $\Lambda < \infty$ such that $\lambda_i = \Lambda\lambda_i'$. We assume without loss of generality that the λ_i' are in non-decreasing order and let $N = \min\{n \in \mathbb{N} | \sum_{i=1}^n > 1 - \delta\}$ where $\delta = \epsilon/(\Lambda(1+\epsilon))$.

Since $\lambda(1+\epsilon) \in \mathcal{C}(d)$, there exists some policy Π that stabilizes λ while leaving a fraction of time slots $\epsilon/(1-\epsilon)$ unused. We consider a modified version of Π where a single arbitrary location (x_i,y_i), $i \geqslant N+1$, is scheduled in the slots unused by Π. It is readily seen that all traffic in locations (x_i,y_i), $i \geqslant N+1$ is stabilized, since

$$\Lambda\sum_{i=N+1}^{\infty}\lambda_i' < \Lambda\delta < \frac{\epsilon}{1-\epsilon},$$

as required. It remains to be shown that all schedules chosen by Π are feasible under K-partition, for sufficiently large K.

Let $d_0 = \min_{i,j\in\{1,2,\ldots,N\}}\|(x_i,y_i) - (x_j,y_j)\| > d$ and choose $K = K^* := \lceil 2\sqrt{2}/(d_0 - d)\rceil$. We will show that for any $\mathcal{P} = (P_1,\ldots,P_n) \in \Omega(d)$ (restricted to locations (x_i,y_i), $i = 1,\ldots,N$), it holds that

$$\{(k_1,l_1),\ldots,(k_n,l_n)\} \in \Omega(K^*, d), \tag{10}$$

with (k_i, l_i) such that $P_i \in \mathcal{R}_{k_i,l_i},\ i = 1, \ldots, n$. From this it is then clear that any feasible schedule under reuse distance d is also feasible under K^*-partition.

In order to prove (10), 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, analogous to the proof of Lemma 1,

$$\|Q_i - Q_j\| > d_0 - 2\sqrt{2}/K^* \geqslant d, \tag{11}$$

so the subset of cells $\{(k_1,l_1),\ldots,(k_n,l_n)\}$ belongs to $\Omega(K^*,d)$. □

References

[1] L. Tassiulas, A. Ephremides, Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks, IEEE Trans. Automat. Control 37 (12) (1992) 1936–1948.
[2] N. Kahale, P. Wright, Dynamic global packet routing in wireless networks, in: Proc. Infocom, Kobe, Japan, 1997, pp. 1416–1423.
[3] L. Tassiulas, A. Ephremides, Dynamic server allocation to parallel queues with randomly varying connectivity, IEEE Trans. Inform. Theory 39 (2) (1993) 466–478.
[4] D. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, P. Whiting, Scheduling in a queueing system with asynchronously varying service rates, Prob. Eng. Inform. Sci. 18 (2) (2004) 191–217.
[5] A. Eryilmaz, R. Srikant, J. Perkins, Stable scheduling policies for fading wireless channels, IEEE/ACM Trans. Netw. 13 (2) (2005) 411–424.
[6] M. Neely, Energy optimal control for time-varying wireless networks, IEEE Trans. Inform. Theory 52 (7) (2006) 2915–2934.
[7] M. Neely, E. Modiano, C. Rohrs, Dynamic power allocation and routing for time-varying wireless networks, IEEE J. Sel. Areas Commun. 23 (1) (2005) 89–103.
[8] A. Eryilmaz, R. Srikant, Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control, in: Proc. Infocom, Miami, FL, 2005, pp. 1794–1803.
[9] M. Neely, E. Modiano, C.-P. Li, Fairness and optimal stochastic control for heterogeneous networks, IEEE/ACM Trans. Netw. 16 (2) (2008) 396–409.
[10] A. Stolyar, Maximizing queueing network utility subject to stability: greedy primal dual algorithm, Queueing Syst. 50 (4) (2005) 401–457.
[11] A. Stolyar, Greedy primal-dual algorithm for dynamic resource allocation in complex networks, Queueing Syst. 54 (3) (2006) 203–220.
[12] L. Georgiadis, M. Neely, L. Tassiulas, Resource allocation and cross-layer control in wireless networks, Found. Trends Netw. 1 (1) (2006) 1–144.
[13] N. McKeown, V. Anantharam, J. Walrand, Achieving 100% throughput in an input-queued switch, in: Proc. Infocom, San Francisco, CA, 1996, pp. 296–302.
[14] N. McKeown, A. Mekkittikul, V. Anantharam, J. Walrand, Achieving 100% throughput in an input-queued switch, IEEE Trans. Commun. 47 (8) (1999) 1260–1267.
[15] S. Meyn, Control Techniques for Complex Networks, Cambridge University Press, 2007.
[16] A. Stolyar, MaxWeight scheduling in a generalized switch: state space collapse and workload minimization in heavy traffic, Ann. Appl. Probab. 14 (1) (2004) 1–53.
[17] L. Tassiulas, Linear complexity algorithms for maximum throughput in radio networks and input queued switches, in: Proc. Infocom, San Francisco, CA, 1998, pp. 533–539.
[18] P. Chaporkar, S. Sarkar, Stable scheduling policies for maximizing throughput in generalized constrained queueing systems, IEEE Trans. Autom. Control 53 (8) (2008) 1913–1931.
[19] X. Lin, N. Shroff, The impact of imperfect scheduling on cross-layer rate control in wireless networks, in: Proc. Infocom, Miami, FL, 2005, pp. 1804–1814.
[20] G. Sharma, R. Mazumdar, N. Shroff, On the complexity of scheduling in wireless networks, in: Proc. MobiCom, Los Angeles, CA, 2006, pp. 227–238.
[21] G. Sharma, N. Shroff, R. Mazumdar, Joint congestion control and distributed scheduling for throughput guarantees in wireless networks, in: Proc. Infocom, Anchorage, AK, 2007, pp. 2072–2080.
[22] X. Wu, R. Srikant, Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks, IEEE Trans. Mobile Comput. 6 (6) (2007) 595–605.
[23] P. van de Ven, S. Borst, V. Shneer, Instability of MaxWeight scheduling algorithms, in: Proc. Infocom, Rio de Janeiro, Brazil, 2009, pp. 1701–1709.
[24] S. Liu, L. Ying, R. Srikant, Throughput-optimal opportunistic scheduling in the presence of flow-level dynamics, IEEE/ACM Trans. Netw. 19 (4) (2011) 1057–1070.
[25] S. Liu, L. Ying, R. Srikant, Scheduling in multichannel wireless networks with flow-level dynamics, in: Proc. ACM SIGMETRICS, New York, NY, 2010, pp. 191–202.
[26] B. Sadiq, G. de Veciana, Throughput optimality of delay-driven maxweight scheduler for a wireless system with flow dynamics, in: Proc. 47th Annual Allerton Conf. Commun., Control, Comput., Monticello, IL., 2009, pp. 1097–1102.
[27] M. Markakis, E. Modiano, J. Tsitsiklis, Scheduling policies for single-hop networks with heavy-tailed traffic, in: Proc. 47th Annual Allerton Conf. Commun., Control, Comput., Monticello, IL, 2009, pp. 112–120.
[28] X. Lin, N. Shroff, R. Srikant, On the connection-level stability of congestion-controlled communication networks, IEEE Trans. Inform. Theory 54 (5) (2008) 2317–2338.
[29] C. Moallemi, D. Shah, On the flow-level dynamics of a packet-switched network, in: Proc. ACM SIGMETRICS, New York, NY, 2010, pp. 83–94.
[30] P. van de Ven, S. Borst, L. Ying, Spatial inefficiency of MaxWeight scheduling, in: Proc. WiOpt, Princeton, NJ, 2011, pp. 62–69.
[31] A. Proutière, Y. Yi, M. Chiang, Distributed algorithm and reversible network, in: Proc. CISS, Princeton, NJ, 2008, pp. 509–514.
[32] I. Keslassy, R. Zhang-Shen, N. McKeown, Maximum size matching is unstable for any packet switch, Tech. Rep. TR03-HPNG-03010, HPNG Technical Report (2003).