Optimal Slicing and Scheduling with Service Guarantees in Multi-Hop Wireless Networks

Nicholas Jones
jonesn@mit.edu
Massachusetts Institute of Technology
Cambridge, MA, USA

Eytan Modiano
modiano@mit.edu
Massachusetts Institute of Technology
Cambridge, MA, USA

ABSTRACT

We analyze the problem of scheduling in wireless networks to meet end-to-end service guarantees. Using network slicing to decouple the queueing dynamics between flows, we show that the network's ability to meet hard throughput and deadline requirements is largely influenced by the scheduling policy. We characterize the feasible throughput/deadline region for a flow under a fixed route and set of slices, and find throughput- and deadline-optimal policies for a solitary flow. We formulate the feasibility problem for multiple flows in a general topology, and show its equivalence to finding a bounded-cost cycle on an exponentially large graph, which is unsolvable in polynomial time by the best-known algorithm. Using a novel concept called delay deficit, we develop a sufficient condition for meeting deadlines as a function of inter-scheduling times, and show that regular schedules are optimal for satisfying this condition. Motivated by this, we design a polynomial-time algorithm that returns an (almost) regular schedule, optimized to meet service guarantees for all flows.

CCS CONCEPTS

• **Networks** → **Network performance modeling**; **Network performance analysis**; **Mobile ad hoc networks**; **Network control algorithms**.

KEYWORDS

Wireless networks, Scheduling, Service guarantees, Network slicing

ACM Reference Format:
Nicholas Jones and Eytan Modiano. 2024. Optimal Slicing and Scheduling with Service Guarantees in Multi-Hop Wireless Networks. In *The Twenty-fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MOBIHOC '24), October 14–17, 2024, Athens, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641512.3686385

1 INTRODUCTION

Future wireless networks must provide stringent throughput and delay guarantees to support new technologies, including real-time control and virtual reality systems. These guarantees take the form of service agreements, and due to their critical nature, network providers must move beyond best-effort service to ensure the agreements are met. As a result, work has begun on new methods of handling such traffic using network slicing [2]. The 5G standard contains support for Quality of Service (QoS) guarantees at the flow level [1], including guarantees on maximum latency seen by any packet up to a given throughput. A largely open question, however, is how to make these service guarantees in wireless networks with limited resources, unreliable links, and interference constraints.

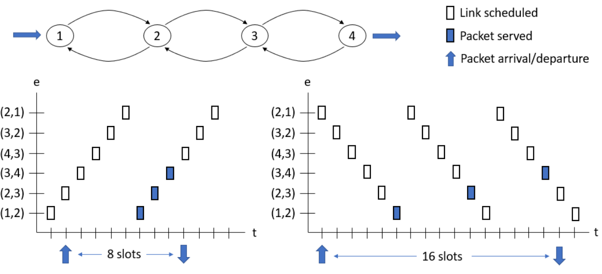

Figure 1: Effect of Scheduling Order on Packet Delay

In this work we follow a similar approach, using network slicing to decouple the queueing dynamics between traffic flows. This is not only practical for making service guarantees at the flow level, but useful for highlighting how wireless interference affects a network's ability to meet these guarantees. By understanding the impact of interference, we can enable networks to meet service requirements with fewer resources and to support more flows.

To illustrate the effect of interference on delay, consider a line network with a flow traveling from node 1 to node 4 as in Figure 1, and for ease of exposition assume an interference model where only one link can transmit at a time. The bottom of the figure shows two round-robin scheduling, policies and the worst-case end-to-end delay that a given packet sees in each case, ignoring any queueing delay. The packet sees very different delays between the two policies, and the reason is clear. On the left it is served in three consecutive slots, while on the right it must wait five slots each time it is served before being served again.

This example shows the impact of *scheduling order* on packet delays. The activation rates and the packet's route remain the same in both examples, but ordering the schedule in the direction of the flow has a significant impact on worst-case delay. As the size of the network grows, more complex interference models are considered, and more flows are introduced with different routes, this impact becomes even more pronounced, while at the same time harder to analyze. Motivated by this, we study the impact of interference and scheduling order on packet delays. Then, coupled with network slicing, we seek to characterize feasible service guarantees and to find supporting policies in general wireless networks.

There is a large body of work on wireless scheduling for maximizing throughput [18, 22, 24, 32]. Most notably, Hajek and Sasaki [18] designed an algorithm to find a minimum schedule length which meets a set of link demands in polynomial time, and Kodialam et al. [24] used Shannon's algorithm for coloring a multigraph to efficiently find schedules that are guaranteed to achieve at least 2/3 of the maximum throughput.

There has also been considerable work done on making QoS guarantees in networks. One of the first approaches was Cruz's network calculus [11, 12], which uses a traffic shaping envelope and the convolution of service processes to bound the delay each packet experiences over multiple hops in a wired network. Several works have extended this to the wireless setting using a variant called *stochastic* network calculus [3, 6, 17, 25], which bounds the tails of arrival and service processes to obtain a high-probability bound on end-to-end delay.

A novel QoS framework for single-hop wireless networks was developed by Hou and Kumar [20]. They assume strict deadlines by which each packet must be delivered over an unreliable channel, and design a policy to ensure the "delivery ratio," or time average fraction of packets which are delivered by their deadline, meets a reliability requirement. They extend this framework in [21] to solve utility maximization and in [19] to support Markov arrival processes. Several works have extended a version of this framework to multi-hop. In [27], the authors analyze a multi-hop network with end-to-end deadline constraints and develop policies to meet delivery ratio requirements over wired links. In [28], the authors design a spatio-temporal architecture with virtual links to solve a similar problem. In [30] and [31], the authors analyze a multi-hop wireless network with unreliable links. They develop policies for maximizing throughput with hard deadlines, using relaxed link capacity constraints and assuming no interference.

The closest to our work is [14] and [15], which consider a multi-hop wireless network with interference constraints and find transmission schedules to meet traffic deadlines under constant arrivals. They show that when the direction of traffic is uniform, the optimal ordering can be found in polynomial time, and they develop heuristics for when the traffic direction is not uniform. The efficiency of these schedules was improved in [10] by optimizing slot re-use for non-interfering links. The authors of [7, 8] generalize the arrival processes to calculus-style envelopes and consider the case of sink-tree networks, while [9] incorporates routing.

Each paper in this line of work considers policies where links are scheduled in one contiguous block of time slots per scheduling period, which reduces the problem complexity but leads to end-to-end packet delays that grow with the schedule length. In this paper, we generalize and extend this line of work by designing scheduling policies without this restriction, deriving feasibility conditions on service guarantees, and incorporating network slicing. Using a novel concept called *delay deficit*, we are able to meet much tighter deadlines without sacrificing throughput or adding complexity. Our main contributions can be summarized as follows.

- In Section 3, we develop necessary conditions on feasibililty for throughput and deadline guarantees, and both throughput- and deadline-optimal policies for a solitary flow under general interference.
- In Section 4, we derive exact solutions to these optimal policies for primary interference, and show that packet delays can grow linearly with the schedule length if the schedule order is not carefully designed.
- In Section 5, we show that when deadlines are non-trivial, solving the feasibility problem for service guarantees in general networks is intractable, and we prove an alternative upper bound on packet delays based on schedule regularity.
- Finally, in Section 6, we develop a polynomial-time algorithm to construct (almost) regular schedules, which are guaranteed to satisfy throughput and delay requirements.

2 PRELIMINARIES

2.1 System Model

We consider a wireless network with fixed topology modeled as a directed graph $G = (V, E)$. Each link $e \in E$ has a fixed capacity c_e and can transmit up to this number of packets in each time slot it is activated. Because links are wireless and share a wireless channel, they are subject to interference, which restricts the sets of links that can be scheduled at the same time. Time is slotted, with the duration of one slot equal to the transmission time over each link, so in each time slot a controller chooses a non-interfering set of links to be active.

We consider a set of interference models Φ, which take the form of ϕ-hop interference constraints. Specifically, for any ϕ, no two links can be activated at the same time if they are separated by fewer than ϕ hops. We will also refer to the interference model itself as $\phi \in \Phi$. The main focus of this work is primary interference, where $\phi = 1$, but our framework can be generalized to any interference model in Φ, and we present general results where possible. Denote M^{ϕ} as the set of feasible link activations, i.e., non-interfering links which can be activated simultaneously, under the model ϕ.

Traffic arrives at the network in the form of flows, which can represent a single customer or an aggregation of customers. Denote flow i as f_i and the set of flows as $\mathcal{F}$. Arrivals are deterministic with λ_i packets from f_i arriving at the beginning of each slot[1]. Each packet belonging to f_i has a deadline τ_i, and we say that a packet meets its deadline if it is delivered to its destination within τ_i slots of when it arrives, otherwise it expires. Flow f_i is assigned a fixed pre-determined route $T^{(i)}$ from source to destination, and we denote $T_j^{(i)}$ as the j-th hop in the route. The set of flows is fixed over a finite horizon T, which is independent of packet deadlines and assumed to be substantially larger. Assume that packet arrivals stop at time T, but packets remaining in the network are given time to be served by their respective deadlines.

Each link $e \in T^{(i)}$ reserves capacity for f_i in the form of a network slice, with capacity, or *slice width*, equal to $w_{i,e}$. We will also refer to the slice itself as $w_{i,e}$ when there is no risk of confusion. Because link capacities are fixed, the sum of all slice widths allocated on link e must be bounded by c_e. Each slice has its own first-come-first-served queue, which decouples the queueing dynamics between flows. Let $Q_{i,e}(t)$ be the size of the queue belonging

[1]This can be generalized to a network calculus-style envelope, and for simplicity of exposition we assume traffic shaping occurs before packets arrive at the source.

to slice $w_{i,e}$ at the beginning of time slot t, after packets have arrived but before any packets are served. We assume the network is empty before $t = 0$, so $Q_{i,e}(t) = 0$ for all i and e, and $t < 0$.

2.2 Policy Structure

Define Π as the set of admissible scheduling policies, which are work-conserving and satisfy the interference constraints ϕ. Let $\mu^{\pi}(t) \in M^{\phi}$ be the set of links activated at time t under π, and let $\mu_e^{\pi}(t) = 1$ if $e \in \mu^{\pi}(t)$, and 0 otherwise. The work-conserving property ensures that each link $e \in \mu^{\pi}(t)$ serves the smaller of its queue size and its slice capacity each time it is activated.

We are interested in policies which meet the following criteria.

Definition 1. *A policy $\pi \in \Pi$ supports a set of flows $\mathcal{F}$ if and only if it allows no packets to expire.*

Specifically, we would like to understand how wireless interference affects scheduling delay, what sets of flows can be supported in general, and how to design policies which support a set of flows while minimizing slice widths, thus leaving as much capacity as possible for best-effort traffic.

Without wireless interference, this problem would be trivial because there is no scheduling component. Under our model of constant arrivals with fixed routes and slicing, each link can forward a slice width of packets at every time step. Then the network need only guarantee that slice widths satisfy both throughput and link capacity constraints, and that the number of hops in each flow's route is less than its deadline. In the wireless setting, this problem becomes nontrivial and interesting. The scheduling policy, which is a function of the interference, dictates how often and in what order links are scheduled. The less frequently a link is served, the larger its slice width must be for a given throughput, and the more delay a packet sees at that link in the worst case.

It can be shown that if a policy exists that supports $\mathcal{F}$, then there must exist a cyclic policy π that supports $\mathcal{F}$, has a period of length K^{π}, and repeats itself once every period. Therefore, without loss of optimality, we consider only cyclic policies in the remainder of this work, and denote this subclass of policies as $\Pi_c \subseteq \Pi$. Note that proofs of all results are omitted for brevity and can be found in the technical report [23].

Under a policy $\pi \in \Pi_c$, define the time average activation frequency of link e as

$$\bar{\mu}_e^{\pi} \triangleq \frac{1}{K^{\pi}} \sum_{t=0}^{K^{\pi}} \mu_e^{\pi}(t), \tag{1}$$

and the number of activations per scheduling period as $\eta_e^{\pi} \triangleq \bar{\mu}_e^{\pi} K^{\pi}$. Finally, let the time average service rate of slice $w_{i,e}$ be

$$\bar{w}_{i,e}^{\pi} \triangleq \bar{\mu}_e^{\pi} w_{i,e}. \tag{2}$$

We will make heavy use of these quantities in the analysis going forward.

3 FEASIBILITY

Having defined our policy class, we now turn to characterizing the feasibility region. Define the feasible region for a route $T^{(i)}$ with slices $\{w_{i,e},\ \forall\ e \in T^{(i)}\}$ as the set of all throughput/deadline pairs which a flow can achieve under ϕ and any scheduling policy in Π_c, and denote this region as Λ_i^{ϕ}. We define this region for a given route to show the feasible throughput/deadline guarantees that can be made to a flow on that route by optimizing the scheduling policy independently of other flows. In the general setting, with many flows on separate routes coupled through the scheduling policy, the jointly achievable region is defined as

$$\Lambda^{\phi} \subseteq \prod_{T^{(i)} \in \mathcal{T}} \Lambda_i^{\phi}, \tag{3}$$

where the scheduling policy must optimize over all flows jointly. Analyzing Λ_i^{ϕ} allows us to define optimal policies for a single flow, as well as bounds on feasibility in the general case. We begin by characterizing throughput optimality.

3.1 Throughput Optimality

Define $\lambda_i^*(\pi)$ as the maximum throughput a policy π can support on $T^{(i)}$, given by the following intuitive result.

Lemma 1. *For any admissible policy $\pi \in \Pi_c$,*

$$\lambda_i^*(\pi) = \min_{e \in T^{(i)}} \bar{w}_{i,e}^{\pi}. \tag{4}$$

Now define a throughput-optimal policy as follows.

Definition 2. *A policy $\pi \in \Pi_c$ is throughput-optimal for a route $T^{(i)}$ and slice widths $\{w_{i,e},\ \forall\ e \in T^{(i)}\}$ if $\lambda_i^*(\pi) \geq \lambda_i^*(\pi')$ for all $\pi' \in \Pi_c$.*

Because $\lambda_i^*(\pi)$ is the largest supported throughput for a given π, a throughput-optimal policy is one that maximizes this quantity over all $\pi \in \Pi_c$. In particular, it is any solution to

$$\begin{aligned} &\max_{\pi \in \Pi_c} \min_{e \in T^{(i)}} \bar{\mu}_e^{\pi} w_{i,e} \\ &\text{s.t. } \mu^{\pi}(t) \in M^{\phi},\ \forall\ 0 \leq t \leq K^{\pi}, \end{aligned} \tag{5}$$

and we denote this solution as λ_i^*. We will solve this problem exactly for the case of primary interference in the next section, but note that on average the optimization tries to drive $\bar{\mu}_e^{\pi} w_{i,e}$ to equality along all links, so in general links wth smaller slice widths tend to be activated more frequently.

3.2 Deadline Optimality

Define $\tau_i^*(\pi, \lambda_i)$ as the smallest deadline the network can guarantee to a flow on $T^{(i)}$ with a throughput of λ_i under policy π. Equivalently, we refer to τ_i^* as the largest end-to-end delay experienced by any packet in the flow. Likewise, let $\tau_i^*(\pi) = \lim_{\lambda_i \to 0} \tau_i^*(\pi, \lambda_i)$ be the smallest deadline that π can guarantee for any $\lambda_i > 0$. Note that because λ_i can be arbitarily close to zero, $\tau_i^*(\pi)$ is independent of slice widths. Because of the equivalence of maximum packet delays and minimum guaranteed deadlines, we will refer to τ_i^* interchangeably by either of these definitions.

To derive a lower bound on $\tau_i^*(\pi)$, we begin by introducing the concept of inter-scheduling times. Denote the set of time slots where link e is scheduled under a policy π as $\mathcal{T}_e^{\pi} \triangleq \{t \geq 0 \mid \mu_e^{\pi}(t) = 1\}$. Then define the minimum inter-scheduling time $\underline{k}_{e,e+1}^{\pi}$ of links e and $e+1$ to be the smallest time interval between consecutive scheduling events of links e and $e+1$, in that order, so that

$$\underline{k}_{e,e+1}^{\pi} \triangleq \min_{t_e \in \mathcal{T}_e^{\pi}} \min_{t_{e+1} > t_e : t_{e+1} \in \mathcal{T}_{e+1}^{\pi}} (t_{e+1} - t_e), \tag{6}$$

and the maximum inter-scheduling time $\overline{k}^{\pi}_{e,e+1}$ to be the largest such time, defined as

$$\overline{k}^{\pi}_{e,e+1} \triangleq \max_{t_e \in \mathcal{T}^{\pi}_e} \min_{t_{e+1} > t_e : t_{e+1} \in \mathcal{T}^{\pi}_{e+1}} (t_{e+1} - t_e). \tag{7}$$

We can also speak of the inter-scheduling times of a single link e as the times between consecutive scheduling events of that link, and denote this as $\underline{k}^{\pi}_e$ and $\overline{k}^{\pi}_e$ respectively for ease of notation. Using these quantities, we can lower bound minimum deadlines as follows.

Lemma 2. *For any admissible policy $\pi \in \Pi_c$,*

$$\tau^*_i(\pi) \geq \underline{k}^{\pi}_0 + \sum_{0 \leq j < |T^{(i)}|-1} \underline{k}^{\pi}_{j,j+1} + 1, \tag{8}$$

for all $f_i \in \mathcal{F}$, and where link $j = T^{(i)}_j$ for all j.

This bound begins to formalize the idea that schedule order plays an important role in minimizing delay. In particular, it motivates us to minimize inter-scheduling times between consecutive links on a route to keep packet delays small. We will show that this results in a deadline-optimal policy, per the following definitions.

Definition 3. *A policy $\pi \in \Pi_c$ is deadline-minimizing for a route $T^{(i)}$, slice widths $\{w_{i,e}, \forall\ e \in T^{(i)}\}$, and throughput λ_i if $\tau^*_i(\pi, \lambda_i) \leq \tau^*_i(\pi', \lambda_i)$ for all $\pi' \in \Pi_c$.*

Furthermore, a policy $\pi \in \Pi_c$ is deadline-optimal for a route $T^{(i)}$ if $\tau^*_i(\pi) \leq \tau^*_i(\pi')$ for all $\pi' \in \Pi_c$.

To avoid confusion, we distinguish between the terms *deadline-minimizing* when speaking in terms of a specific throughput, and *deadline-optimal* when speaking independently of throughput.

We will show that deadline optimality is achieved by a subclass of Π_c we call *ordered round-robin* (ORR) scheduling policies, which minimize the bound in (8) while showing that it is tight. Denote the ORR policy for $T^{(i)}$ as $ORR(i)$, and recall that under an interference model ϕ, links separated by fewer than ϕ hops cannot be scheduled simultaneously. Then define the ORR policy as follows. At each time t, activate link $T^{(i)}_j$, where $j = t \bmod \phi$, along with every $\phi + 1$ subsequent links. This ensures that, at each slot, links at equally spaced intervals of $\phi + 1$ hops are activated, beginning with hops $\{0, \phi + 1, 2\phi + 2, \dots\}$ at time $t = 0$, hops $\{1, \phi + 2, 2\phi + 3, \dots\}$ at $t = 1$, and so on. Note that this schedule has a period of $\phi + 1$.

Theorem 1. *The ORR policy is deadline-optimal under any interference model $\phi \in \Phi$, with a maximum packet delay*

$$\tau^*_i\big(ORR(i)\big) = |T^{(i)}| + \phi, \tag{9}$$

and activation rates

$$\bar{\mu}^{ORR(i)}_e = \frac{1}{\phi + 1}, \forall e \in T^{(i)}. \tag{10}$$

Note that the ORR policy meets the bound in (8) with equality, because the minimum inter-scheduling times are $\underline{k}_0 = \phi$ and $\underline{k}_{e,e+1} = 1$ for all e. From (4), the maximum throughput under the ORR policy is $\lambda^*_i(ORR(i)) = \min_{e \in T^{(i)}} \frac{w_{i,e}}{\phi+1}$, which is not throughput-optimal in general, and highlights the tradeoff between maximizing throughput and minimizing delay. When slice widths are equal, however, this tradeoff does not occur.

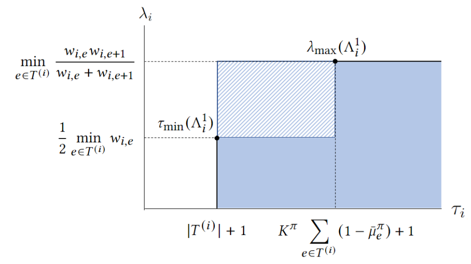

Figure 2: Feasible Region Λ^1_i Under Primary Interference

Corollary 1. *When slice widths are equal across $T^{(i)}$, the ORR(i) policy is both throughput-optimal and deadline-optimal.*

4 PRIMARY INTERFERENCE

Having characterized bounds on the feasibility region and the structure of throughput- and deadline-optimal policies under general interference, we develop exact results for primary interference, which is the main focus of this work.

The deadline-optimal point $\tau_{\min}$ is straightforward. From Theorem 1, recalling that $\phi = 1$ under primary interference, the minimum achievable deadline is $\tau^*_i = |T^{(i)}| + 1$ under an ORR(i) policy. Furthermore, from Lemma 1, this policy can support a maximum throughput of $\lambda^*_i(ORR(i)) = \frac{1}{2} \min_{e \in T^{(i)}} w_{i,e}$.

Obtaining the throughput-optimal point $\lambda_{\max}$ is less trivial. A throughput-optimal policy can be derived from (5) by relaxing the link activation constraint to be $\bar{\mu}_e + \bar{\mu}_{e+1} \leq 1$ for all adjacent link pairs $(e, e + 1)$. In our single route scenario under primary interference, only adjacent links interfere with one another, so this is a necessary and sufficient condition for feasibility. Then a throughput-optimal set of activations is any solution to

$$\begin{aligned} &\max_{\pi \in \Pi_c} \min_{e \in T^{(i)}} \bar{\mu}^{\pi}_e w_{i,e} \\ &\quad \text{s.t. } \bar{\mu}^{\pi}_e + \bar{\mu}^{\pi}_{e+1} \leq 1,\ \forall\ 0 \leq e < |T^{(i)}| - 1, \end{aligned} \tag{11}$$

which can be solved exactly.

Theorem 2. *Any throughput-optimal policy $\pi \in \Pi_c$ on $T^{(i)}$ under primary interference constraints has a maximum throughput of*

$$\lambda^*_i(\pi) = \min_{e \in T^{(i)}} \frac{w_{i,e} w_{i,e+1}}{w_{i,e} + w_{i,e+1}} \tag{12}$$

As a byproduct of the proof (found in [23]), one throughput-optimal set of activation rates is shown to be

$$\begin{aligned} \bar{\mu}^{\pi^*}_e &= \min\left\{\frac{w_{i,e-1}}{w_{i,e} + w_{i,e-1}}, \frac{w_{i,e+1}}{w_{i,e} + w_{i,e+1}}\right\},\ \forall\ 1 \leq e < |T^{(i)}| - 1, \\ \bar{\mu}^{\pi^*}_0 &= \frac{w_{i,1}}{w_{i,0} + w_{i,1}},\ \bar{\mu}^{\pi^*}_{-1} = \frac{w_{i,-2}}{w_{i,-1} + w_{i,-2}}, \end{aligned} \tag{13}$$

where we slightly abuse notation to denote $|T^{(i)}| - j$ as $-j$. Let the corresponding optimal throughput be λ^*_i. If the activation rates are feasible, algorithms exist to find a throughput-optimal policy π^* in polynomial time [18]. In order to ensure that rates are met and the number of activations $\eta^{\pi^*}_e = \bar{\mu}^{\pi^*}_e K^{\pi^*}$ is integer, the schedule length K^{π^*} must be at least as large as the least common multiple of the denominators of the activation rates, which can become arbitrarily large.

While π^* is guaranteed to be throughput-optimal if it exists, it is not deadline-minimizing in general. Finding a deadline-minimizing policy for this throughput is more difficult because, while throughput optimality requires only time average constraints, we have seen that deadline guarantees are largely dependent on schedule order. When slice widths are equal, $\bar{\mu}_e^{\pi^*} = \frac{1}{2}$ for all e, and the throughput-optimal point converges to the deadline-optimal (and therefore deadline-minimizing) point as described in Corollary 1. In the general case, finding a deadline-minimizing policy is equivalent to finding a cost-minimizing cycle on a graph, where vertices represent the size of each queue and edges represent valid link activations. This has complexity which is exponential in $|T^{(i)}|$. We omit the analysis due to space constraints and encourage readers to reference the technical report [23].

When schedule order is not optimized, packet delays can become large for a given schedule length and set of activation rates, as shown in the following result.

Theorem 3. *For any policy $\pi \in \Pi_c$,*

$$\tau_i^*(\pi, \lambda_i) \leq K^\pi \sum_{e \in T^{(i)}} (1 - \bar{\mu}_e^\pi) + 1 \tag{14}$$

under primary interference constraints, for any feasible throughput $0 < \lambda_i \leq \lambda_i^*(\pi)$. Moreover, there exists at least one policy where this bound is tight, and many policies where τ_i^* grows linearly with K^π.

We have seen that K^π can grow arbitrarily large for general activation rates, so the bound in Theorem 3 does not provide strong deadline guarantees. In fact, it shows there is at least one policy $\pi \in \Pi_c$ where packets experience delays within a constant factor of $K^\pi |T^{(i)}|$, and many policies where packets experience delays that grow with K^π. As detailed in the proof in [23], these policies are not pathological examples, but rather any policy with inter-scheduling times that grow with the schedule length. This includes policies that schedule each link in one contiguous block per scheduling period (as in [14] and the following line of work), or simply random orderings of activations which happen to have large inter-scheduling times. We formalize this idea in the next section, and to avoid delays that grow with K^π, we introduce policies with bounded inter-scheduling times. We show that by restricting ourselves to this class of policies, we can make deadline guarantees that are independent of the schedule length without sacrificing thoughput.

In Figure 2, we show the feasible throughput/deadline region for a flow under primary interference, highlighting the deadline-optimal point and the throughput-optimal point with the upper bound on packet delay from Theorem 3. The shaded region between the two optimal points is as equally hard to characterize as the minimum deadline under λ_i^*, but one can show the boundary is monotonic, and a proof sketch is in the technical report [23].

5 SCHEDULING FOR SERVICE GUARANTEES

We now move to the problem of scheduling multiple flows in a general network under primary interference, drawing on the feasibility results of the previous section. As a secondary goal, we seek to keep slices as close as possible to resource-minimizing, per the following definition, which follows from Lemma 1.

Definition 4. *A set of slices under a given policy $\pi \in \Pi_c$ and a set of flows $\mathcal{F}$ is resource-minimizing if $w_{i,e} = \frac{\lambda_i}{\bar{\mu}_e^\pi}$ for all $f_i \in \mathcal{F}$ and $e \in T^{(i)}$.*

Efficiently sized slices allow the network to support more deadline-constrained traffic, as well as more best-effort traffic with the unallocated capacity. We have seen that it is challenging to design policies with deadline guarantees that are tighter than the universal upper bound in Theorem 3. This is true in the case of a solitary flow, as shown in the previous section, and certainly remains true in the general case with multiple flows.

Theorem 4. *Given a set of flows $\mathcal{F}$ with non-trivial deadlines (i.e., tighter than the bound in Theorem 3), finding a policy $\pi \in \Pi_c$ that supports $\mathcal{F}$ under any set of slices has complexity which is exponential in $O\big(\sum_{f_i \in \mathcal{F}} |T^{(i)}|\big)$.*

We provide a brief proof sketch, and refer readers to the technical report [23] for more details. By fixing slice widths, we can constrain the state space of possible queue sizes to be finite, which allows us to represent them as vertices on a graph. Then, following the same idea used in the previous section for a solitary flow, we show that finding a policy which supports $\mathcal{F}$ is equivalent to finding a cycle on this graph, where edges represent valid link activations. This is $O(V)$ in the worst case, which is exponential in $O\big(\sum_{f_i \in \mathcal{F}} |T^{(i)}|\big)$.

From this result, it is clear that searching over all policies in Π_c is intractable. To narrow our search, it will prove helpful to identify a tighter bound on packet delays as a function of the scheduling policy. Using a similar argument to that in Lemma 2, which gives a lower bound on τ_i^* in terms of minimum inter-scheduling times, we will show an upper bound on τ_i^* subject to conditions on *maximum* inter-scheduling times.

To do so, we introduce a concept called delay deficit. Intuitively, delay deficit provides a quota of worst-case delay which a packet should expect to see at a link under some scheduling assumptions, and it tracks how long each packet has been in the network relative to this quota. The details are in the technical report [23], but at a high level, a negative delay deficit signifies that a packet has spent less than its allocated time on its route so far, and is ahead of schedule. A delay deficit larger than the delay quota at a packet's current link signifies that the packet is behind schedule. By induction on the delay deficit at each link, a packet's end-to-end delay can be bounded by the sum of delay quotas along its route, under certain slice conditions. In particular, it leads to the following bound.

Theorem 5. *For any policy $\pi \in \Pi_c$, with slice widths $w_{i,e} \geq \lambda_i \overline{k}_e^\pi$ for all $f_i \in \mathcal{F}$ and $e \in T^{(i)}$,*

$$\tau_i^*(\pi, \lambda_i) \leq \sum_{e \in T^{(i)}} \overline{k}_e^\pi, \ \forall f_i \in \mathcal{F}, \tag{15}$$

where $\overline{k}_e^\pi$ is the maximum inter-scheduling time of link e under π.

We note several things about this result. First, it subsumes the worst-case delay bound in Theorem 3. In the proof of that theorem, the worst-case scenario is described to have a maximum inter-scheduling time $\overline{k}_e^\pi = K^\pi(1 - \bar{\mu}_e^\pi)$ for all links e, which makes the bounds identical up to a difference of 1 slot. Second, when links are scheduled more regularly and $\overline{k}_e$ is independent of the schedule length, this bound can be significantly tighter than that in Theorem 3. In fact, it is a constant factor of $(\sum_{e\in T^{(i)}} \overline{k}_e^\pi)/|T^{(i)}|$ from the deadline-optimal lower bound on τ_i^* for a solitary flow in Theorem 1.

A final thing to note is that Theorem 5 requires slice widths that are not resource-minimizing in general. Define the additional capacity required beyond the resource-minimizing value as

$$\Delta w_{i,e}(\pi) \triangleq \lambda_i\big(\overline{k}_e^\pi - \frac{1}{\bar{\mu}_e^\pi}\big) \geq 0,\ \forall\ f_i \in \mathcal{F},\ e \in T^{(i)}. \tag{16}$$

Because the average inter-scheduling time is $1/\bar{\mu}_e^\pi$, the quantity $\Delta w_{i,e}(\pi)$ is small when inter-scheduling times are somewhat regular and $\overline{k}_e^\pi$ is not much larger than this average.

It is important to clarify that, with these conditions on slice widths, a link is not guaranteed to empty its queue each time it is scheduled. Any given packet can spend more than $\overline{k}_e^\pi$ slots at link e, but Theorem 5 guarantees it will make up for this by spending fewer than $\overline{k}_{e'}^\pi$ slots at some other link e' on its route. The strength of Theorem 5 lies in this fact, which enables us to keep $\Delta w_{i,e}(\pi)$ small and to allocate slices efficiently.

From (15) and (16), we see that minimizing maximum inter-scheduling times leads to both tighter delay bounds and more efficient slices. If inter-scheduling times are exactly equal, and $\underline{k}_e^\pi = \overline{k}_e^\pi = \frac{1}{\bar{\mu}_e^\pi}$ for all $e \in E$, then $\overline{k}_e^\pi$ is by definition minimized for a fixed $\bar{\mu}_e^\pi$. If this holds, we say that π is a *regular schedule.*

Corollary 2. *A policy $\pi \in \Pi_c$ supports a set of flows $\mathcal{F}$ with slice widths $w_{i,e} = \lambda_i \overline{k}_e^\pi$, for all $f_i \in \mathcal{F}$ and $e \in T^{(i)}$, if*

$$\begin{aligned} \sum_{e\in T^{(i)}} \overline{k}_e^\pi &\leq \tau_i,\ \forall\ f_i \in \mathcal{F}, \\ \sum_{f_i: e\in T^{(i)}} \lambda_i \overline{k}_e^\pi &\leq c_e,\ \forall\ e \in E. \end{aligned} \tag{17}$$

If π is a regular schedule, then slices are resource-minimizing.

While the bound in Theorem 5 is not tight in general, and smaller deadline guarantees may be possible, it highlights the importance of keeping inter-scheduling times small. This intuitively makes sense given our knowledge of the deadline-optimal ORR policy. The number of interfering links and the multi-directional flows in this general setting make it impossible to schedule links in order along a flow's route, as in the ORR policy. Rather, by bounding the inter-scheduling times, we can minimize how far the schedule deviates from this order, and do this simultaneously for all flows. This keeps end-to-end delays from growing too large, and allows us to tune the inter-scheduling times on some flows' routes to become "closer" to the ORR policy when subject to tighter deadlines. Setting $\overline{k}_e = 2$ for all links on a flow's route recovers the ORR policy exactly under primary interference.

Motivated by this, we develop efficient algorithms in the next section to construct schedules with bounded inter-scheduling times, focusing particularly on regular schedules. We will see that by using the sufficient conditions in Corollary 2, we are able to develop policies which support $\mathcal{F}$ in polynomial time. Because the conditions in Corollary 2 are not necessary, this does not violate Theorem 4, but rather accomplishes our goal of narrowing the policy search to a smaller subclass of Π_c, trading some performance for a large reduction in complexity.

6 ALGORITHM DEVELOPMENT

In this section, we continue to focus on primary interference, for which activation sets are matchings on the graph G. We have seen that regular schedules allow for resource-minimizing slices, which maximize network capacity, so we seek to find regular schedules whenever possible. Unfortunately such schedules often do not exist for a given set of activation rates (we will show this later), so we allow for *almost-regular* schedules, where inter-scheduling times can differ by at most one slot. The remainder of the section details the steps our algorithm takes to construct these schedules.

6.1 Initialization

Our algorithm takes as input a set of flows $\mathcal{F}$, and throughput and deadline constraints λ_i and τ_i respectively for each $f_i \in \mathcal{F}$. Assume we are able to design an almost-regular schedule, where link e has α inter-scheduling times of k_e and β inter-scheduling times of $k_e - 1$. Then for any schedule length K, $\alpha k_e + \beta(k_e - 1) = K$. Recall also that $\bar{\mu}_e = \eta_e/K = (\alpha+\beta)/K$, so by rearranging terms and substituting,

$$k_e = \frac{K}{\alpha+\beta} + \frac{\beta}{\alpha+\beta} = \frac{1}{\bar{\mu}_e} + \frac{\beta}{\alpha+\beta} = \Big\lceil \frac{1}{\bar{\mu}_e} \Big\rceil < \frac{1}{\bar{\mu}_e} + 1, \tag{18}$$

for any α and β, because k_e is integer-valued. Note that the last equality holds because $\beta/(\alpha+\beta) < 1$ and $1/\bar{\mu}_e$ is integer if and only if the schedule is regular and $\beta = 0$.

From Corollary 2 and the bound in (18), any set of activation rates which solve the following program meet service guarantees, provided the algorithm can construct an almost-regular schedule satisfying these rates.

$$\begin{aligned} \min\ & \sum_{e\in E} \bar{\mu}_e \\ \text{s.t.}\ & \sum_{e\in T^{(i)}} \Big(\frac{1}{\bar{\mu}_e} + 1\Big) \leq \tau_i,\ \forall\ f_i \in \mathcal{F}, \\ & \sum_{f_i: e\in T^{(i)}} \lambda_i \Big(\frac{1}{\bar{\mu}_e} + 1\Big) \leq c_e,\ \forall\ e \in E. \end{aligned} \tag{19}$$

This problem is convex, and we denote the solution as $\bar{\mu}^0$, also referred to as the initial link rates. The algorithm must now find an almost-regular schedule where $\bar{\mu}_e \geq \bar{\mu}_e^0$ for all links e.

6.2 Unique-Edge Matchings

One hurdle in finding schedules with regularity constraints under primary interference is that, in general, work-conserving policies assign links to more than one matching [18]. As a result, constructing a regular schedule of matchings does not guarantee the schedule is regular for each link, which complicates the problem.

Lemma 3. *Determining the existence of an almost-regular schedule, which satisfies the constraints in* (19) *under primary interference, is NP-complete.*

This result follows from the equivalence of our problem to the well-studied Periodic Event Scheduling Problem [29], which is known to be NP-complete [13]. The details can be found in the technical report [23], but we note that much of the hardness in the problem comes from the matching constraints and the ability of each link to belong to more than one matching. It may seem that we are back to square one, given that this problem is at least as hard as the problem we started with in Theorem 4. If we constrain the problem further, however, by forcing each link to belong to a single matching, we can make the problem tractable in a way that was initially unavailable.

Following this idea, we design an algorithm which assigns each link to a single matching in an efficient manner. Then when matchings are scheduled regularly, each link is also scheduled regularly. We refer to such matchings as *unique-edge matchings*, and will see that under this additional constraint we are able to find almost-regular schedules in polynomial time.

Clearly each link in a unique-edge matching is activated at the same rate as its matching. In order to ensure the constraints in (19) are met, our algorithm must guarantee the initial matching rate $\bar{\mu}_m^0 \geq \bar{\mu}_e^0$ for all links e belonging to matching m, and for all matchings m. Coupled with the unique-edge condition, this problem can be written as

$$\begin{aligned} \min_{M \subseteq M^1} \; & \sum_{m \in M} \bar{\mu}_m^0 \\ \text{s.t. } & \bar{\mu}_m^0 \geq \bar{\mu}_e^0, \; \forall \, e \in m, m \in M \\ & e \in M, \; \forall \, e \in E, \\ & m \cap m' = \varnothing, \; \forall \, m, m' \in M, \end{aligned} \tag{20}$$

where M^1 is the set of all feasible matchings. Because only one matching can be activated at a time, M is feasible if and only if $\sum_{m \in M} \bar{\mu}_m^0 \leq 1$, which motivates the objective function.

Despite being of practical interest, and related to the minimum edge sum coloring problem [4] and the weighted sum coloring problem [5, 16], this exact formulation has received no attention in the literature. We define a *Greedy Matching* (GM) algorithm, which has complexity $O(|E|^2)$ and provides an approximate solution to this problem, in Algorithm 1.

Algorithm 1: Greedy Matching (GM)

```
Input: Initial link activation rates $\bar{\mu}_e^0, \; \forall \, e \in E$
Output: Set of unique edge matchings $M^{GM}$
1  Sort links in $E$ by $\bar{\mu}_e^0$, largest to smallest
2  Define matching set $M^{GM}$
3  while $E$ is not empty do
4      Add new matching $m$ to $M^{GM}$
5      Add first link $e' \in E$ to $m$ and remove $e'$ from $E$
6      Set $\bar{\mu}_m^0 = \bar{\mu}_{e'}^0$
7      for $e \in E$ do
8          if $m \cup e$ is a valid matching then
9              Add $e$ to $m$ and remove $e$ from $E$
10 Return $M^{GM}$, $\bar{\mu}_m^0, \; \forall \, m \in M^{GM}$
```

The algorithm sorts links in decreasing order of initial rates, and greedily adds them to a matching if they do not interfere with any links previously added. Once all links have been considered, the matching is full and is given an initial rate equal to that of the first link added, which is guaranteed to be the largest. This greedy heuristic ensures the constraints in (20) are met and the matchings it returns are unique-edge matchings, while keeping the objective within a small factor of optimal as shown next.

Theorem 6. *Let M^* be a solution (i.e., an optimal set of matchings) to* (20). *The GM algorithm produces a set of matchings M^{GM} such that*

$$\sum_{m \in M^{GM}} \bar{\mu}_m^0 \leq \left(\frac{|M_{GM}|}{\Delta(G)} \right) \sum_{m \in M^*} \bar{\mu}_m^0, \tag{21}$$

where $\Delta(G)$ is the maximum degree of G. This is a factor of 2 from optimal in the worst case, provided $\Delta(G) = O(\log |V|)$.

6.3 Feasibility Conditions

Although we are satisfied with almost-regular schedules, we also examine the feasibility conditions for regular schedules, which will play a role in our algorithm development. Define a *step-down* vector as a sorted vector of values, where each element is an integer multiple of the next. When applied to activation rates, this property turns out to be quite useful in constructing regular schedules.

Lemma 4. *A regular schedule of unique-edge matchings exists if*

(1) $\sum_{m \in M^{GM}} \bar{\mu}_m = 1$,
(2) *the vector of rates $\bar{\mu}$ is step-down,*
(3) *and $k_m = \lceil \frac{1}{\bar{\mu}_m} \rceil = \frac{1}{\bar{\mu}_m}$ for all m.*

Furthermore, an almost-regular schedule exists if only conditions (1) *and* (2) *hold.*

Initial rates can always be increased without violating the conditions in (17), so if a set of rates is feasible, then they can be normalized to sum to 1 to meet condition (1). The other conditions are more difficult to meet, particularly at the same time, and a regular schedule is not guaranteed to exist for a feasible set of rates.

In [26], Li et al. construct a polynomial-time algorithm which takes a set of rates and outputs a set of possibly larger *augmented rates* $\hat{\mu}$ that are step-down. When applied to initial activation rates, the output satisfies condition (2) from Lemma 4. We do not replicate the algorithm here due to space constraints, but encourage readers to reference Algorithm 2 and the accompanying results in [26].

Because $\hat{\mu}$ does not always satisfy condition (3), there is no guarantee that we can form a regular schedule with this set of rates. However, when $\sum_m \hat{\mu}_m \leq 1$, the rates can be normalized to meet condition (1) without affecting the step-down property of the vector. This ensures that we can form an almost-regular schedule. A result from [26] shows that when the sum of input rates is at most ln 2, the sum of augmented rates is at most 1. Extrapolating to our setting yields the following result.

Lemma 5. *An almost-regular schedule of unique-edge matchings exists if $\sum_m \bar{\mu}_m^0 \leq \ln 2 \approx 0.69$.*

6.4 Schedule Construction

We finally move to the last step of the algorithm, which is schedule construction. Lemma 5 shows that when initial rates are small enough, the augmented rates algorithm produces a vector of rates that are step-down and can be normalized to sum to 1. We can construct an almost-regular schedule from these rates by first finding a longer, regular schedule where not all slots are filled, and then removing the unused slots. The algorithm for constructing schedules in this manner is first described by Li et al. in [26]. We replicate the algorithm here, and provide a full proof of correctness in the technical report [23], which was omitted in the original work.

```
Algorithm 2: Almost-Regular Schedule Construction
  Input: Flows $\mathcal{F}$, throughput/deadline constraints $(\lambda_i, \tau_i)$
  Output: Almost regular schedule $\pi \in \Pi_c$
1  Find initial link activations $\bar{\mu}_e^0$, $\forall\ e$, by solving (19)
2  Find unique-edge matchings $M^{GM}$ and initial rates $\bar{\mu}_m^0$, $\forall\ m$,
     using Greedy Matching
3  Find $\hat{\mu}$ using the Augmenting Rates algorithm from [26]
4  if $\sum_m \hat{\mu}_m > 1$ then
5      Return None
6  Normalize rates $\bar{\mu}_m^\pi = \hat{\mu}_m / \sum_m \hat{\mu}_m$ for all $m$
7  Set $M = |M^{GM}|$, $K^\pi = 1/\bar{\mu}_M^\pi$ and $\eta_m = \bar{\mu}_m^\pi K^\pi$ for all $m$
8  Form an empty schedule with length $K' = \lceil \frac{1}{\bar{\mu}_1^\pi} \rceil \eta_1$.
9  Assign slot 1 to $m_1$, along with every $K'/\eta_1$ subsequent slots
10 for $i = 2, 3, \ldots, |M|$ do
11     Set $S_0$ as the set of empty slots
12     for $j = 1, 2, \ldots, i-1$ do
13         Set $S_j \subseteq S_{j-1}$ as the set of slots most closely
             following a slot assigned to $m_j$
14     Assign the first slot in $S_{i-1}$ to $m_i$, along with every $K/\eta_i$
         subsequent slots
15 Remove the $K' - K^\pi$ unassigned slots from the schedule
16 Set $\overline{k}_e^\pi$ as the max inter-scheduling time of link $e$ in $\pi$
17 Set $w_{i,e} = \lambda_i \overline{k}_e^\pi$ for all $f_i \in \mathcal{F}$ and $e \in T^{(i)}$
18 Return $\pi$, $w$
```

Let $\bar{\mu}$ be the augmented vector of rates after normalizing, sorted from largest to smallest, $M = |M|$ be the number of matchings, and $K = k_M = 1/\bar{\mu}_M$ be the schedule length. From the step-down property, this is guaranteed to be an integer. To see this, let $z_{i,j} = \frac{\bar{\mu}_i}{\bar{\mu}_j}$ for all pairs of matchings i and j, and note that $z_{i,j}$ is integer-valued by the step-down property when $i \le j$. Then $\bar{\mu}_m = z_{m,M}\bar{\mu}_M$ for all m. Substituting into condition (1) and rearranging yields

$$K = \frac{1}{\bar{\mu}_M} = \sum_m z_{m,M}, \tag{22}$$

which by definition is an integer.

Define the number of occurences of matching m in the schedule as $\eta_m = \bar{\mu}_m K$, and define an auxiliary schedule of length $K' = k_1\eta_1 = \lceil \frac{1}{\bar{\mu}_1} \rceil \eta_1$. Note that the schedule length is an integer multiple of k_1, which allows matching 1 to be scheduled regularly every k_1 slots when it does not overlap or "collide" with any other matching in the schedule. Now consider an arbitrary matching m. It too can be scheduled regularly every $k'_m = \frac{K'}{\eta_m}$ slots (assuming no collisions) if k'_m is an integer. By definition,

$$k'_m = \frac{K'}{\bar{\mu}_m K} = \frac{z_{1,m}K'}{\bar{\mu}_1 K} = \frac{z_{1,m}K'}{\eta_1} = z_{1,m}k_1, \tag{23}$$

which is integer-valued. Therefore, every matching can be scheduled regularly in a schedule of length K', provided there are no collisions in the schedule. Collisions can be avoided by scheduling matchings in a greedy fashion, beginning with matching 1 and proceeding in increasing order of k'_m. Because the vector is already sorted, this is simply the ordering of the vector. For each matching m, pick an empty slot in the schedule and assign it to m, along with every k'_m subsequent slots. Due to the step-down property of the vector and the greedy ordering, it is easy to show that no collisons can occur following this method.

The resulting schedule is regular for all matchings, and has length K'. If $\frac{1}{\bar{\mu}_1}$ is integer-valued, then $K' = K$, and the method described above generates a regular schedule that supports $\mathcal{F}$. If not, there will be $K' - K$ empty slots in the schedule after all matchings have been added. These slots are simply removed, yielding a schedule of length K with activation rates $\bar{\mu}_m$ for each matching m. To ensure this schedule is almost-regular, the first slot assigned to matching m should be picked to spread out the remaining empty slots.

The full details of the schedule construction are described in Algorithm 2, which we refer to as the Almost-Regular Schedule Construction (ARSC) algorithm. This algorithm has complexity $O(|E|^2 + M^2K^2 + X)$, where M is the number of unique-edge matchings, K is the schedule length, and X is the complexity of the algorithm used to solve the convex program (19). In addition to implementing the procedure above, it yields the following result.

Theorem 7. *If the augmented rates found in step* 3 *of the ARSC algorithm satisfy $\sum_m \hat{\mu}_m \le 1$, then the algorithm returns an almost-regular schedule π, which meets all throughput and deadline guarantees for $\mathcal{F}$, and a corresponding set of slices, where each slice $w_{i,e}$ is within a factor of $\frac{1}{\overline{k}_e^\pi} \le \bar{\mu}_e^\pi$ of being resource-minimizing.*

The final part of this theorem is critical, because it shows that our algorithm does not waste network resources while meeting service guarantees. A more naive method of meeting guarantees is simply overprovisioning slices, but this result shows that the network can operate at near capacity while implementing ARSC.

We further illustrate the schedule construction and details of the ARSC algorithm through an example. Let the normalized augmented rates be the step-down vector $\bar{\mu} = \left(\frac{2}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{10}\right)$, the schedule length $K = 10$, and $\eta_1 = 4$. We begin by constructing a longer schedule of length $K' = \eta_1 k_1 = 12$ slots, and assign matching 1 to the first slot followed by every $k_1 = 3$ subsequent slots. This gives an initial schedule

$$\underline{1}\ \underline{\ }\ \underline{\ }\ \underline{1}\ \underline{\ }\ \underline{\ }\ \underline{1}\ \underline{\ }\ \underline{\ }\ \underline{1}\ \underline{\ }\ \underline{\ }.$$

For each subsequent matching i, the algorithm takes the set of empty slots S_1 that most closely follow matching 1 in the schedule (here $S_1 = \{2, 5, 8, 11\}$). Note that because the schedule is cyclic, we assume it wraps around when computing the closest following slots. If $i > 2$, it then forms a set $S_2 \subseteq S_1$ with the set of empty slots that most closely follow matching 2. It proceeds through each matching which has already been scheduled in the same fashion, and finally assigns the first slot in S_{i-1} to matching i, followed by slots at subsequent spacings of K/η_i. Following this method, the first slot in S_1 is slot 2, so matching 2 is assigned to this slot, followed by slots at equal spacings of $K/\eta_2 = 6$. This yields

$$\underline{1}\ \underline{2}\ \underline{\ }\ \underline{1}\ \underline{\ }\ \underline{\ }\ \underline{1}\ \underline{2}\ \underline{\ }\ \underline{1}\ \underline{\ }\ \underline{\ }.$$

Moving on to matching 3, we have $S_1 = \{5, 11\}$. Both of these slots lag matching 2 by 3 slots in the schedule, so $S_2 = S_1$, and we assign matching 3 to the first slot 5, followed by slots at increments of $K/\eta_3 = 6$. This gives us

$$\underline{1}\ \underline{2}\ \underline{\ }\ \underline{1}\ \underline{3}\ \underline{\ }\ \underline{1}\ \underline{2}\ \underline{\ }\ \underline{1}\ \underline{3}\ \underline{\ }.$$

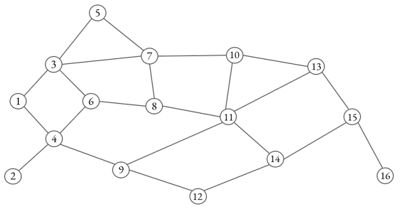

Figure 3: Example Network Diagram

For matching 4, $S_1 = \{3, 6, 9, 12\}$, and $S_2 = S_3 = \{3, 9\}$. Therefore we assign matching 4 to slot 3, and because $\eta_4 = 12$, this is the only slot where it is assigned. Finally, for matching 5, $S_1 = \{6, 9, 12\}$, and $S_2 = S_3 = S_4 = 9$. Adding matching 5 and removing the remaining empty slots ultimately yields the schedule

$$\underline{1}\ \underline{2}\ \underline{4}\ \underline{1}\ \underline{3}\ \underline{1}\ \underline{2}\ \underline{5}\ \underline{1}\ \underline{3}.$$

One can verify that this final schedule has length $K = 10$, activation rates match the initial $\bar{\mu}$, and the schedule is almost-regular.

6.5 Discussion

The ARSC algorithm provides an efficient, polynomial-time method for constructing schedules to meet hard throughput and deadline constraints, under primary interference and general network topologies. It can meet tight deadline guarantees, and it allocates slices within a small factor of being resource-minimizing. Of course, Theorem 4 shows that finding a feasible policy in general suffers from exponential complexity, so ARSC cannot find a schedule for every set of flows where one exists. The algorithm runs in polynomial time by constraining the problem and taking suboptimal steps.

Forcing the class of policies to be regular or almost-regular is one such restriction, as are the unique-edge matching condition and the requirement that augmented rate vectors be step-down. These conditions are necessary for the ARSC algorithm to produce a feasible schedule, but are not necessary for a feasible schedule to exist. The gap between the set of all feasible schedules and the set returned by ARSC is the price we pay for the reduction in complexity. Lemma 5 shows that when the sum of initial matching rates is less than $\ln 2 \approx 0.69$, ARSC will always succeed, but there is no necessary condition on the algorithm's success that we have found. We explore the algorithm's feasible solution rate through simulations in the next section.

7 NUMERICAL RESULTS

We support our results in this section by simulating the ARSC algorithm and observing its performance numerically. In all our simulations we use the example network in Figure 3, and note that while the diagram shows bidirectional links, we treat each of these as two directional links which interfere as in the analysis. We further assume the network topology and link capacities are fixed.

In each experiment, we generate 32 flows with randomly chosen source/destination pairs, and fix their routes using shortest path routing. We assume that each flow has an identical throughput λ and deadline τ. For perspective on our algorithm's performance, we compute the largest value λ^* that λ can take independent of any deadline constraints, by finding a throughput-optimal set of matchings. We then scale the throughput of each flow using this value.

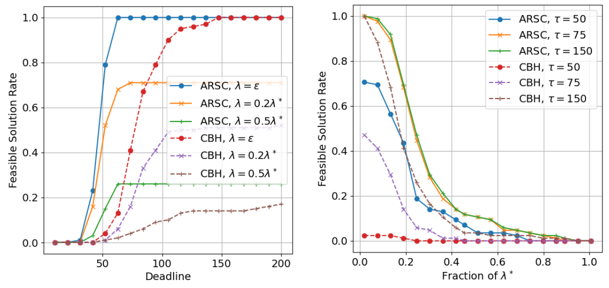

Figure 4: Feasibility Rate

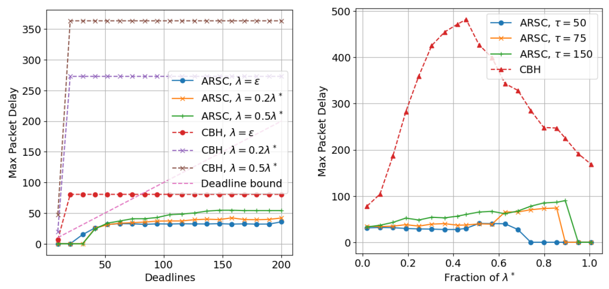

Figure 5: Max Packet Delay

In the previous section, the ARSC algorithm was shown to meet service guarantees when it returns a feasible almost-regular schedule, so we first examine how often this occurs. In Figure 4, we show the feasible solution rate of the ARSC algorithm under varying throughputs λ and deadlines τ by generating 100 random sets of 32 flows and averaging the results. We directly compare these results to the Credit-Based Heuristic (CBH) algorithm in [10], the closest to our work and the most recent result in the line of work starting with [14], which makes use of contiguous block scheduling and which we have referenced on several occasions.

On the left, we sweep deadlines and run both algorithms under three values of λ, where ϵ is the smallest throughput allowed by the implementation, and the other values are scaled by λ^*. For an arbitrarily small $\lambda = \epsilon$, we expect ARSC to always return a policy for a deadline of 60 and above, because the longest path any flow can take is 6 hops, and the network can be colored with 10 colors. Therefore, a simple round-robin policy will ensure the deadline is met, and because a round-robin schedule is regular by definition, ARSC will find it. The plot confirms that this is indeed the case.

In any direct comparison between ARSC and CBH, ARSC shows significant improvement. This is most notable in the deadline range between 40 and 70, during which CBH still has a small solution rate but where ARSC reaches 100% for a throughput of ϵ and 70% for a throughput of $0.2\lambda^*$. Below this range, using the same arguments of path length and graph coloring referenced above, the probability that any policy can find a solution for all 32 flows becomes increasingly low. By the opposite argument, it becomes easier to do so as deadlines increase beyond this range. This region where meeting deadlines is just barely feasible is where our algorithm shines, proving that ARSC can meet tight deadline guarantees. On the right side of Figure 5, we sweep throughput and perform the same comparison for three fixed deadlines, showing more clearly the tradeoff between feasibililty and throughput.

In Figure 5, we plot the maximum delay seen by any packet under each simulated policy, averaged over the same 100 sets of randomly generated flows. On the left, we again sweep deadlines and plot results for both ARSC and CBH under varying throughputs. We also plot the deadline bound for reference. When there is no feasible policy under ARSC, the max delay is plotted as 0. As expected, no packet sees a delay larger than the deadline bound under ARSC. The difference in delay between throughput levels is relatively small, which makes sense if the limiting factor is interference constraints and schedule order, rather than link capacity.

The CBH algorithm sees much larger packet delays on average, which we expect given that link inter-scheduling times are close to the schedule length. This demonstrates our result that regular schedules lead to much smaller packet delays, and thus tighter achievable deadlines.

On the right, we sweep throughput and plot results for varying deadlines. Because CBH is computed independently of the deadline bound, the policy remains the same and is only plotted once. Once again, ARSC produces consistent results and low delays across throughputs and deadlines, while CBH sees much larger delays.

8 CONCLUSION

In this paper, we studied the effect of wireless interference on scheduling for service guarantees. We defined the feasible region of throughput and deadline guarantees for a solitary flow, and showed that without carefully structuring the order of a scheduling policy, packets can experience large scheduling delay. To alleviate this problem, we showed that under regular schedules, tight deadline guarantees can be made which are independent of the schedule length. Finally, we developed an algorithm to construct almost-regular schedules in polynomial time, which are guaranteed to meet service requirements for all flows, while leaving sufficient capacity for best-effort traffic to achieve near-optimal throughput. Future work includes optimizing routing as well as scheduling, and dealing with unreliable links, changing wireless topologies, and stochastic traffic.

ACKNOWLEDGMENTS

This material is based upon work supported by the Department of the Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Air Force.

This work was also supported in part by NSF grants CNS-2148183 and CNS-2148128.

REFERENCES

[1] Accessed: 03-21-2024. *Quality of Service (QoS) in 5G Networks.* https://5ghub.us/quality-of-service-qos-in-5g-networks/
[2] Accessed: 03-21-2024. *T-Mobile Launches First-Ever 5G Network Slicing Beta for Developers.* https://www.t-mobile.com/news/network/t-mobile-launches-first-ever-5g-network-slicing-beta-for-developers
[3] Hussein Al-Zubaidy, Jörg Liebeherr, and Almut Burchard. 2013. A (min,×) network calculus for multi-hop fading channels. In *2013 Proceedings IEEE INFOCOM.* IEEE, 1833–1841.
[4] Amotz Bar-Noy, Mihir Bellare, Magnús M Halldórsson, Hadas Shachnai, and Tami Tamir. 1998. On Chromatic Sums and Distributed Resource Allocation. *Information and Computation* 140, 2 (Feb. 1998), 183–202.
[5] Amotz Bar-Noy, Magnús M Halldórsson, Guy Kortsarz, Ravit Salman, and Hadas Shachnai. 2000. Sum multicoloring of graphs. *Journal of Algorithms* 37, 2 (2000), 422–450.
[6] Almut Burchard, Jörg Liebeherr, and Stephen D Patek. 2006. A min-plus calculus for end-to-end statistical service guarantees. *IEEE Transactions on Information Theory* 52, 9 (2006), 4105–4114.
[7] P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini. 2009. Link scheduling with end-to-end delay constraints in Wireless Mesh Networks. In *2009 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks & Workshops.* IEEE, Kos, Greece, 1–9.
[8] P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini. 2011. Efficient link scheduling for online admission control of real-time traffic in wireless mesh networks. *Computer Communications* 34, 8 (June 2011), 922–934.
[9] Paola Cappanera, Luciano Lenzini, Alessandro Lori, Giovanni Stea, and Gigliola Vaglini. 2013. Optimal joint routing and link scheduling for real-time traffic in TDMA Wireless Mesh Networks. *Computer Networks* 57, 11 (Aug. 2013), 2301–2312.
[10] Shanti Chilukuri and Anirudha Sahoo. 2015. Delay-aware TDMA Scheduling for Multi-Hop Wireless Networks. In *Proceedings of the 16th International Conference on Distributed Computing and Networking.* ACM, Goa India, 1–10.
[11] Rene L Cruz. 1991. A calculus for network delay. I. Network elements in isolation. *IEEE Transactions on information theory* 37, 1 (1991), 114–131.
[12] Rene L Cruz. 1991. A calculus for network delay. II. Network analysis. *IEEE Transactions on information theory* 37, 1 (1991), 132–141.
[13] W. Dauscha, H. D. Modrow, and A. Neumann. 1985. On cyclic sequence types for constructing cyclic schedules. *Zeitschrift für Operations Research* 29, 1 (March 1985), 1–30.
[14] Petar Djukic and Shahrokh Valaee. 2007. Quality-of-service provisioning for multi-service TDMA mesh networks. In *Managing Traffic Performance in Converged Networks: 20th International Teletraffic Congress, ITC20 2007, Ottawa, Canada, June 17-21, 2007. Proceedings.* Springer, 841–852.
[15] P. Djukic and S. Valaee. 2009. Delay Aware Link Scheduling for Multi-Hop TDMA Wireless Networks. *IEEE/ACM Transactions on Networking* 17, 3 (June 2009), 870–883.
[16] Leah Epstein, Magnús M. Halldórsson, Asaf Levin, and Hadas Shachnai. 2009. Weighted Sum Coloring in Batch Scheduling of Conflicting Jobs. *Algorithmica* 55, 4 (Dec. 2009), 643–665.
[17] Markus Fidler. 2006. An end-to-end probabilistic network calculus with moment generating functions. In *200614th IEEE International Workshop on Quality of Service.* IEEE, 261–270.
[18] B. Hajek and G. Sasaki. 1988. Link scheduling in polynomial time. *IEEE Transactions on Information Theory* 34, 5 (Sept. 1988), 910–917.
[19] I-Hong Hou. 2013. Scheduling heterogeneous real-time traffic over fading wireless channels. *IEEE/ACM Transactions on Networking* 22, 5 (2013), 1631–1644.
[20] I.-H. Hou, V. Borkar, and P. R. Kumar. 2009. A Theory of QoS for Wireless. In *IEEE INFOCOM 2009.* 486–494.
[21] I-Hong Hou and PR Kumar. 2010. Utility-optimal scheduling in time-varying wireless networks with delay constraints. In *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing.* 31–40.
[22] Kamal Jain, Jitendra Padhye, Venkata N Padmanabhan, and Lili Qiu. 2003. Impact of interference on multi-hop wireless network performance. In *Proceedings of the 9th annual international conference on Mobile computing and networking.* 66–80.
[23] Nicholas Jones and Eytan Modiano. 2024. Optimal Slicing and Scheduling with Service Guarantees in Multi-Hop Wireless Networks. arXiv:2404.08637
[24] Murali Kodialam and Thyaga Nandagopal. 2003. Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem. In *Proceedings of the 9th annual international conference on Mobile computing and networking.* 42–54.
[25] Chengzhi Li, Almut Burchard, and Jörg Liebeherr. 2007. A network calculus with effective bandwidth. *IEEE/ACM Transactions on Networking* 15, 6 (2007), 1442–1453.
[26] Chengzhang Li, Qingyu Liu, Shaoran Li, Yongce Chen, Y. Thomas Hou, and Wenjing Lou. 2021. On Scheduling with AoI Violation Tolerance. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications.* IEEE, Vancouver, BC, Canada, 1–9.
[27] Ruogu Li and Atilla Eryilmaz. 2012. Scheduling for end-to-end deadline-constrained traffic with reliability requirements in multihop networks. *IEEE/ACM Transactions on Networking* 20, 5 (2012), 1649–1662.
[28] Xin Liu, Weichang Wang, and Lei Ying. 2019. Spatial–temporal routing for supporting end-to-end hard deadlines in multi-hop networks. *Performance Evaluation* 135 (2019), 102007.
[29] Paolo Serafini and Walter Ukovich. 1989. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics* 2, 4 (1989), 550–581.
[30] Rahul Singh and PR Kumar. 2018. Throughput optimal decentralized scheduling of multihop networks with end-to-end deadline constraints: Unreliable links. *IEEE Trans. Automat. Control* 64, 1 (2018), 127–142.
[31] Rahul Singh and PR Kumar. 2021. Adaptive CSMA for decentralized scheduling of multi-hop networks with end-to-end deadline constraints. *IEEE/ACM Transactions on Networking* 29, 3 (2021), 1224–1237.
[32] Leandros Tassiulas and Anthony Ephremides. 1990. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. In *29th IEEE Conference on Decision and Control.* IEEE, 2130–2132.