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 as a mixed-integer program, and show that it grows exponentially in the size of the network. Drawing on results from the solitary flow setting, we show that scheduling links in a regular fashion leads to smaller delay, and we derive tighter upper bounds on end-to-end delay for regular schedules. Finally, we design a polynomial-time algorithm that returns an (almost) regular schedule, optimized to meet service guarantees for all flows.

ACM Reference Format:
Nicholas Jones and Eytan Modiano. . Optimal Slicing and Scheduling with Service Guarantees in Multi-Hop Wireless Networks. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 18 pages.

1 INTRODUCTION

Future wireless networks will need to support traffic with stringent throughput and delay requirements, including real-time control and virtual reality systems, and cannot rely on best-effort service to meet these needs. 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 specified throughput, and while still in the early stages, work has begun to implement these guarantees using network slicing [2]. A largely open question, however, is how to make these service guarantees in wireless networks with limited resources, unreliable links, and interference constraints.

In this work we use network slicing to decouple the queueing dynamics between traffic flows. This is not only practical for making service guarantees, but useful for highlighting how wireless interference affects a network's ability to meet these guarantees. We show that interference can still lead to large scheduling delay in the absence of queueing delay, and we characterize the feasible throughput/deadline region for a flow under a fixed route and set of slices. We show that scheduling links in a regular manner can significantly reduce scheduling delay, and we provide polynomial-time algorithms to design regular schedules guaranteed to meet service agreements while maximizing capacity left over for best-effort traffic.

There has been a multitude of work done on wireless scheduling for maximizing throughput [17, 21, 22, 31]. In particular, Hajek and Sasaki [17] designed an algorithm to find a minimum schedule length which meets a set of link demands in polynomial time, and Kodialam et al. [22] used Shannon's algorithm for coloring a multigraph to find schedules even more efficiently that are guaranteed to achieve at least $2/3$ of 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 [10, 11], which characterizes the average rate and burstiness of packet arrivals using a traffic shaping envelope, and then uses the convolution of service processes to bound the delay each packet experiences over multiple hops in a wired network. Several works have extended this framework to the wireless setting using a variant called *stochastic* network calculus [3, 5, 15, 23], which bounds the tails of arrival and service processes to obtain a high-probability bound on end-to-end delay.

A second QoS framework for single-hop wireless networks in the stochastic setting was developed by Hou et al. [19]. It assumes a constant arrival process from each source and a strict deadline by which each packet must be delivered over an unreliable channel. By tracking the rates at which packets from each source are delivered, they 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 [20] to solve utility maximization and in [18] to support Markov arrival processes.

Several works have extended a version of this framework to multi-hop. In [25], 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 [26], the authors design a spatio-temporal architecture with virtual links to solve a similar problem. In [29] and [30], the authors analyze a multi-hop wireless network with unreliable links. By analyzing each packet individually, they develop both centralized and decentralized policies for maximizing throughput with hard deadlines, using relaxed link capacity constraints and assuming no interference between links.

The closest to our work is [13] and [14], which considers a multi-hop wireless network with interference constraints and finds a transmission schedule which meets deadline guarantees under constant traffic arrivals. They develop a mixed-integer program to find an optimal ordering of link activations and bound packet delay as a function of the schedule length. They show that when the direction of traffic is uniform, the optimal ordering can be found in polynomial time. The efficiency of these schedules was improved in [9] by optimizing slot re-use for non-interfering links. The authors of [6, 7] generalize the arrival processes to calculus-style envelopes and consider the case of sink-tree networks, while [8] incorporates routing.

In this paper we use a similar model, but allow for a broader class of scheduling policies. The line of work beginning with [13] considers policies where each link is scheduled in one contiguous block of time slots each scheduling period, which leads to end-to-end packet delay that grows with the schedule length. We show that for general flows with non-uniform traffic, a schedule can grow arbitrarily long when slices are kept small to maximize the use of capacity, so we place no restrictions on the scheduling policy except those dictated by the interference constraints. This allows us to meet tight deadline requirements that are shorter than the length of the schedule, without overprovisioning slices. Furthermore, the policies we consider hold for general networks and are not restricted to tree topologies.

We characterize the feasible throughput/deadline region for a solitary flow by developing necessary conditions on feasibiilty, and both throughput- and deadline-optimal policies under a general interference model. This serves as a bound on the feasible region in the general case where flows are inherently coupled through scheduling. In this general case, we formulate a mixed-integer program to find a policy which satisfies service guarantees, and show that the size of the program grows exponentially in the number of links. To obtain a more tractable solution, we derive an upper bound on end-to-end delay under policies where links are scheduled regularly, and show that is within a constant factor of the lower bound. This provides a set of sufficient conditions for meeting deadline guarantees. Finally, we develop a polynomial-time algorithm that constructs a regular schedule optimized to meet these conditions without overprovisioning slices.

The remainder of this paper is organized as follows. In Section 2, we introduce the system model and the class of policies we will consider. In Section 3, we prove bounds on the feasibility region and optimal policies for solitary flows under general interference models. In Section 4, we show specific feasibility results for different interference models, highlighting primary interference. In Section 5, we extend the problem to the case of general flows, formulate a mixed-integer program to solve the feasibility problem, and derive sufficient conditions for meeting deadline guarantees under regular schedules. In Section 6, we develop a polynomial-time algorithm, which finds (almost) regular schedules optimized to meet these conditions, thereby satisfying service agreements. Finally, in Section 7 we show simulations supporting our results.

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 each time 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 centralized 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. In the special cases of no interference and complete interference, where only one link can be activated at a time, define ϕ to be 0 and $|E| - 1$ respectively. Finally, 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 the set of flows as $\mathcal{F}$, and let each flow $f_i \in \mathcal{F}$ have arrival rate λ_i and deadline τ_i respectively. Arrival rates are fixed, so exactly λ_i packets arrive in each slot. We will speak of packets as discrete units, but for simplicity of analysis we allow λ_i to take fractional values. Arrival rates can be relaxed to a calculus-style traffic envelope, but again for simplicity of analysis we assume traffic shaping occurs before packets arrive at the source, and that the source link sees constant arrivals. 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. Traffic arrives at the beginning of a slot, 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 the packet expires. Generalizing to fractional packet values, we say that all traffic meets its deadline if all λ_i arrivals are delivered before expiring. 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 $T + 1$, but packets remaining in the network are given time to be served before another set of flows arrives.

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 $w_{i,e}$. When there is no risk of confusion, we will also refer to the slice dedicated to f_i on link e as $w_{i,e}$. Because link capacities are fixed and bounded, 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 delay of each flow. Let $Q_{i,e}(t)$ be the size of the queue belonging 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 $t < 0$.

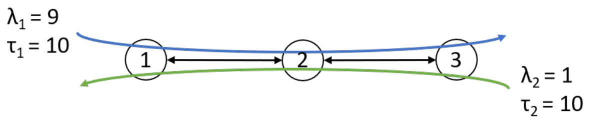

Figure 1: Illustrative Two-Hop Example

2.2 Policy Structure

Define Π as the set of admissable scheduling policies, which are work-conserving and satisfy the interference constraints ϕ. We designate variables which are a function of the scheduling policy π with a superscript. Let $\mu^{\pi}(t) \in M^{\phi}$ be the set of links activated at time t under π, and let $\mu_e^{\pi}(t) = \mathbb{1}(e \in \mu^{\pi}(t))$. The work-conserving property ensures that each link $e \in \mu^{\pi}(t)$ serves the smaller of its queue size and its slice capacity in packets each time it is activated. Because we allow fractional packet values, fractional traffic can be served.

We are interested in policies which meet the following criteria.

Definition 1. *A policy $\pi \in \Pi$ supports the set of flows $\mathcal{F}$ on a time interval if and only if no packets expire during that interval. If no interval is specified, it is assumed to hold for all $t \geq 0$.*

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, 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. Consider the simple two-hop example in Figure 1. Here we have a network consisting of two bidirectional links, and two flows traveling in opposite directions. The flows have equal deadline requirements of 8 slots, but vastly different throughput requirements. We assume primary interference, and because each of the four directional links borders each other, only one of the four can transmit in each slot.

First consider the simple round-robin schedule $\{(1,2), (2,3), (3,2), (2,1)\}$, repeated every four slots. To achieve constant deadlines, slice widths at each link must be at least $w_{i,e} = \frac{\lambda_i}{\bar{\mu}_e}$, where $\bar{\mu}_e = \frac{1}{4}$ is the time average activation rate for each e (we will formalize this argument later). Therefore, under this policy, we have $w_{1,(1,2)} = w_{1,(2,3)} \geq 36$ and $w_{2,(3,2)} = w_{2,(2,1)} \geq 4$. The total capacity allocated is the sum of all the slice widths, and is equal to 80. With these slice widths, each link is able to serve its entire queue when it is scheduled, and one can show that any packet that arrives in either flow sees an end-to-end delay of at most 5 slots, which satisfies the deadline requirement. This leads to our first takeaway: *round-robin policies generally lead to low packet delay.*

This round-robin policy meets the service agreements for both flows, but allocates more capacity than is necessary. It can be shown that a schedule of length 8 with $\bar{\mu}_{(1,2)} = \bar{\mu}_{(2,3)} = \frac{3}{8}$, and $\bar{\mu}_{(3,2)} = \bar{\mu}_{(2,1)} = \frac{1}{8}$ only requires slice widths of $w_{1,(1,2)} = w_{1,(2,3)} \geq 24$ and $w_{2,(3,2)} = w_{2,(2,1)} \geq 8$. The total capacity allocated under this policy is 64, a 20% reduction from the round-robin policy. This is in fact the smallest amount of capacity which can support these flows under any schedule. Now consider the end-to-end delay seen by packets in flow 2. Links $(3,2)$ and $(2,1)$ are served once every 8 slots, and when scheduled consecutively in that order, the maximum delay seen by a packet is 9 slots, which still meets deadline constraints but is nearly double the delay under the round-robin policy. Packets in flow 1 still see a maximum delay of 5, so this policy both meets service agreements and uses the smallest amount of capacity to do so, which we say is resource-minimizing. This leads to our second takeaway: *there is a tradeoff between minimizing slice widths/maximizing throughput and minimizing delay.*

Finally, consider the schedule $\{(1,2), (2,3), (1,2), (2,3), (1,2), (2,3), (2,1), (3,2)\}$, and note that this has the same schedule length and activations as the policy above, but a slightly different order. Now consider a packet from flow 2 that arrives in the first slot of this schedule. It must wait 8 slots before being served by link $(3,2)$, and then another 7 slots before being served by link $(2,1)$. This schedule, therefore, does not meet the service agreements because it violates the deadline constraints for flow 2. This gives our third takeaway: *schedule order, not just activation rates, can drastically affect delay.*

Finally, imagine the deadline requirements were 8 slots (the schedule length in the policies above) instead of 10. Then the resource-minimizing policy would not meet the requirements under any schedule order. In fact, no matter how many slots are assigned to links $(3,2)$ and $(2,1)$, no policy can meet deadlines shorter than the schedule length when the slots assigned to each link occur consecutively (we will show this formally later). By spreading the out in a more regular fashion in the schedule, however, tighter deadlines can be met. Even in this small example, the varying link demands lead to a factor of 2 difference in schedule length from the shortest (round-robin) schedule to a resource-minimizing one. As the size of the network and the variability of link demands grow, efficient slicing requires longer and longer schedules, which leads to our final takeaway: *tighter deadlines can be achieved with more regular schedules.*

Packets arrive at a fixed rate in our model, so the dynamics of the system can be described exactly under a fixed policy in Π. A consequence of this fact combined with finite deadlines is that queue sizes are bounded under any policy that supports $\mathcal{F}$. This allows us to analyze the inherently difficult problem of delay guarantees in multi-hop wireless networks in a more tractable way. In particular, it leads to the following result.

Theorem 1. *If there exists a policy in Π that supports $\mathcal{F}$, then there must exist at least one such policy π that is cyclic with period K^{π}, so that*

$$\mu^{\pi}(t) = \mu^{\pi}(t + K^{\pi}), \ \forall\ t \geq 0, \tag{1}$$

and under π,

$$Q^{\pi}_{i,e}(t) = Q^{\pi}_{i,e}(t + K^{\pi}),\ \forall\, f_i \in \mathcal{F},\ e \in E,\ t_0 \le t \le T, \tag{2}$$

for some $t_0 > 0$ *and sufficiently large* T.

Proof. Recall that arrival rates are fixed on the interval $0 \le t \le T$, so exactly λ_i packets of f_i arrive in each slot. In order for a policy to support $\mathcal{F}$, no packet can exist in the system for longer than its deadline, so at most $\lambda_i \tau_i$ packets from f_i can be present in the system at the end of a slot. This condition is both necessary and sufficient on this interval because packets are served in a FCFS manner, so π supports $\mathcal{F}$ up to time T if and only if $\sum_{e \in T^{(i)}} Q^{\pi}_{i,e}(t) \le \lambda_i \tau_i$ for all $f_i \in \mathcal{F}$ on this interval. By assumption, at least one policy π' exists that supports $\mathcal{F}$, so this condition must hold under π'. Define the state of the system at time t as the length of each queue and denote it by $s^{\pi'}(t)$. Because queue lengths are bounded under π', there exist a finite number of states which can be visited over any time horizon T. In particular, for sufficiently large T, there must exist a state $s^{\pi'}(t_0)$ which occurs at time t_0 and then occurs again at time $t_0 + K^{\pi'}$ for some $K^{\pi'} > 0$. Let $s^{\pi'}(t_0)$ be the first state where this event occurs, and let the set of actions taken in the time interval $[t_0, t_0 + K^{\pi'})$ be the policy π with $K^{\pi} = K^{\pi'}$ and $\mu^{\pi}(t) = \mu^{\pi'}\big((t - t_0) \bmod K^{\pi} + t_0\big)$ for all $t \ge 0$. Then for all $t_0 \le t \le T$, we have $s^{\pi}(t) = s^{\pi'}\big((t - t_0) \bmod K^{\pi} + t_0\big)$, so (6) holds and π supports $\mathcal{F}$ for all $t_0 \le t \le T$.

Note that $t_0 > 0$ because queues are empty at $t = 0$ and the system requires time to "ramp up". However, we next show that π supports $\mathcal{F}$ on the interval $0 \le t < t_0$ using induction on the queue sizes. The queue evolution equations are given by

$$Q^{\pi}_{i,e}(t+1) = \min\{Q^{\pi}_{i,e}(t) + \tilde{\lambda}_{i,e}(t) - \mu^{\pi}_{e}(t) w_{i,e},\ 0\}, \tag{3}$$

for all $0 \le t \le T$, where

$$\tilde{\lambda}_{i,e}(t) = \begin{cases} \lambda_i, & e = T^{(i)}_0, \\ \mu^{\pi}_{e_{-1}}(t) \cdot \min\{w_{i,e_{-1}},\ Q^{\pi}_{i,e_{-1}}(t)\}, & e \ne T^{(i)}_0, \end{cases} \tag{4}$$

and with a slight abuse of notation we denote the link preceding e in $T^{(i)}$ as e_{-1}, where it is understood we are referring to f_i. Now assume that $Q^{\pi}_{i,e}(t) \le Q^{\pi}_{i,e}(t + K^{\pi})$ for all i and e. Then, from the queue evolution equations and given that $\mu^{\pi}(t) = \mu^{\pi}(t + K^{\pi})$, this implies that $Q^{\pi}_{i,e}(t+1) \le Q^{\pi}_{i,e}(t + K^{\pi} + 1)$ for all i and e as well. By definition, $Q^{\pi}_{i,e}(0) \le Q^{\pi}_{i,e}(K^{\pi})$ for all i and e, which completes the induction step. Therefore,

$$\sum_{e \in T^{(i)}} Q^{\pi}_{i,e}(t) \le \sum_{e \in T^{(i)}} Q^{\pi}_{i,e}(t + nK^{\pi}) \le \lambda_i \tau_i \tag{5}$$

for all f_i and $0 \le t < t_0$, and some $n > 0$ such that $t_0 \le t + nK^{\pi} < t_0 + K^{\pi}$, which shows that π supports $\mathcal{F}$ on the interval $0 \le t < t_0$.

It can easily be shown that π also supports $\mathcal{F}$ on the interval $t > T$. Assume for all $t > T$, dummy packets arrive with the same rate λ_i for each flow, until all real packets have been delivered. Then, because π supports every flow under regular arrivals, it must also support every flow under the dummy arrivals. This completes the proof. □

This result also provides the following necessary and sufficient condition for meeting deadlines.

Corollary 1. *A cyclic policy* $\pi \in \Pi$ *with sufficiently large* T *supports* $\mathcal{F}$ *if and only if*

$$\max_{t \ge 0} \sum_{e \in T^{(i)}} Q^{\pi}_{i,e}(t) \le \lambda_i \tau_i,\ \forall\, f_i \in \mathcal{F}. \tag{6}$$

Proof. Because T and λ_i are finite, the sum is bounded and the maximum is guaranteed to exist. From the proof of Theorem 1, the condition (6) is necessary and sufficient for any π to support $\mathcal{F}$ on the interval $0 \le t \le T$ because of the constant arrivals and FCFS queues. The proof also shows that any policy which supports $\mathcal{F}$ on the interval $t \le T$ also supports $\mathcal{F}$ for $t > T$ by assuming dummy arrivals until all real packets have been delivered. Provided T is sufficiently large for π to have completed at least one cycle, the queue sizes with dummy arrivals will be no larger than those before time T, and the result follows. □

Corollary 1 is incredibly useful for finding and verifying that a policy supports a set of flows. Once a policy is fixed, the system is completely deterministic, and iteratively solving the queue evolution equations allows one to track queue sizes at each t, verifying that deadlines are being met without keeping track of the age of each individual packet. The utility of this result as well as Theorem 1 motivate us to consider only cyclic policies in the remainder of this work, without loss of optimality. Denote this class of policies as $\Pi_c \subseteq \Pi$. We will also assume in the remainder of this work that T is sufficiently large for any policy to have completed at least one cycle.

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

$$\bar{\mu}^{\pi}_{e} \triangleq \frac{1}{K^{\pi}} \sum_{t=0}^{K^{\pi}} \mu^{\pi}_{e}(t), \tag{7}$$

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

$$\bar{w}^{\pi}_{i,e} \triangleq \bar{\mu}^{\pi}_{e} w_{i,e}. \tag{8}$$

3 GENERAL FEASIBILITY

Having fully defined our policy class, we now turn to characterizing the feasibility region. Define the feasible region for a route $T^{(i)}$ and set of slice widths $\{w_{i,e},\ \forall\ e \in T^{(i)}\}$ 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^{ϕ}_{i}. 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

$$\Lambda^{\phi} \subseteq \prod_{T^{(i)} \in \mathcal{T}} \Lambda^{\phi}_{i}, \tag{9}$$

where the scheduling policy must optimize over all flows jointly and in general this subset is strict.

Analyzing Λ^{ϕ}_{i} then provides two benefits by letting us define optimal policies for a single route, as well as bounds on feasibility in the general case. It is easy to design policies with low throughput and large deadlines, so we expect Λ^{ϕ}_{i} to define some maximum achievable throughput and some minimum achievable deadline. We begin by characterizing throughput optimality.

3.1 Throughput Optimality

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

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

$$\lambda_i^*(\pi) = \min_{e \in T^{(i)}} \bar{w}_{i,e}^{\pi}. \tag{10}$$

Proof. Assume for contradiction that $\lambda_i > \bar{w}_{i,e}^{\pi}$ for some f_i and e. In one scheduling period K^π for $t \leq T$, $\lambda_i K^\pi$ packets are added to the queue and at most $\bar{w}_{i,e}^{\pi} K^\pi$ packets are served. Then $Q_{i,e}(t + K^\pi) > Q_{i,e}(t)$ for all $t \leq T$, so from Theorem 1, either $\pi \notin \Pi_c$ or does not support f_i, which is a contradiction. □

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, any throughput-optimal policy maximizes this quantity over all $\pi \in \Pi_c$. In particular, it is a solution to

$$\begin{aligned} &\max_{\pi \in \Pi_c} \min_{e \in T^{(i)}} \bar{\mu}_e^{\pi} w_{i,e} \\ &\text{s.t. } \mu^\pi(t) \in M^\phi, \ \forall \ 0 \leq t \leq K^\pi, \end{aligned} \tag{11}$$

and we denote the solution as λ_i^*. We will define exact solutions to this problem for different ϕ in the next section, but note that on average the optimization tries to drive $\bar{\mu}_e^{\pi} w_{i,e}$ to equality, so in general links wth smaller slices tend to be activated more frequently.

3.2 Deadline Optimality

Next define $\tau_i^*(\pi, \lambda_i)$ as the smallest deadline that π can meet on $T^{(i)}$ with a throughput of λ_i, and let $\tau_i^*(\pi) = \lim_{\lambda_i \to 0} \tau_i^*(\pi, \lambda_i)$ be the smallest deadline that π can meet for any $\lambda_i > 0$. Note that because λ_i can be arbitarily close to zero, $\tau_i^*(\pi)$ is also independent of slice widths.

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

$$\underline{k}_{e,e+1}^{\pi} \triangleq \min_{t_e \in \mathcal{T}_e^\pi} \ \min_{t_{e+1} > t_e : t_{e+1} \in \mathcal{T}_{e+1}^\pi} (t_{e+1} - t_e), \tag{12}$$

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

$$\overline{k}_{e,e+1}^{\pi} \triangleq \max_{t_e \in \mathcal{T}_e^\pi} \ \min_{t_{e+1} > t_e : t_{e+1} \in \mathcal{T}_{e+1}^\pi} (t_{e+1} - t_e). \tag{13}$$

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 $\underline{k}_e^\pi$ and $\overline{k}_e^\pi$ respectively for ease of notation. Then we can lower bound minimum deadlines as follows.

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

$$\tau_i^*(\pi) \geq \underline{k}_0^\pi + \sum_{0 \leq e < |T^{(i)}|-1} \underline{k}_{e,e+1}^{\pi} + 1, \tag{14}$$

where link $j = T_j^{(i)}$ for all j.

Proof. Any packet which is delivered must traverse all links on its route in order, and under a policy π, every packet served by link e that arrives at link $e+1$ must wait at least $\underline{k}_{e,e+1}^{\pi}$ slots before being served. Therefore, the smallest amount of time between being served at the source link and being served at the destination link is $\sum_{0 \leq e < |T^{(i)}|-1} \underline{k}_{e,e+1}^{\pi}$. At least λ_i packets must wait $\underline{k}_0^\pi$ slots from when they arrive at the source link until they are served, and it takes one slot to be delivered once served at the destination link. The result follows. □

This bound starts to formalize the takeaway from the example in the previous section, that schedule order plays an important role in meeting small deadlines. In particular, it motivates us to minimize inter-scheduling times between consecutive links on a route. 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 (14) 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 time $t = 0$, activate the first link in the route, followed by every $\phi + 1$ subsequent links; at time $t = 1$, activate the second link followed by every $\phi + 1$ subsequent links, and so on. Formally, this can be expressed as

$$\begin{aligned} \mu^{ORR(i)}(t) \triangleq \{&T_{t \bmod \phi + m(\phi+1)}^{(i)}, \ \forall \ m \in \mathbb{Z}^+ \cup \{0\} \mid \\ &t \bmod \phi + m(\phi+1) < |T^{(i)}|\}, \end{aligned} \tag{15}$$

and note that this schedule has a period of $\phi + 1$.

Theorem 2. *The ORR policy is deadline-optimal under any interference model $\phi \in \Phi$, with*

$$\tau_i^*\big(ORR(i)\big) = |T^{(i)}| + \phi, \tag{16}$$

and

$$\bar{\mu}_e^{ORR(i)} = \frac{1}{\phi + 1}, \ \forall \ e \in T^{(i)}. \tag{17}$$

Proof. The ORR policy schedules links in order from source to destination in subsequent time slots according to the definition above. The activation rate follows because each link is only activated once per scheduling period. Now assume a packet arrives at the source at some time t, and that slice widths are large enough to serve all enqueued packets when a link is scheduled. This is non-restrictive because τ_i^* is defined as the smallest feasible deadline for an arbitrarily small throughput. At time t mod $(\phi+1)$, the packet is served at link $T_0^{(i)}$, and following the ORR schedule, it is served at each subsequent link in the next $|T^{(i)}|-1$ slots. In the worst case, the packet must wait ϕ slots at the source before being served, so it spends a maximum of $|T^{(i)}|+\phi$ slots in the network, which verifies τ_i^* for the ORR policy.

It remains to show that no other policy can achieve a smaller deadline for all packets. Recall that only one of $\{T_0^{(i)},\dots,T_\phi^{(i)}\}$ can be scheduled in the same slot. We claim that it must take at least some packets $2\phi+1$ slots to reach link $T_{\phi+1}^{(i)}$. The fewest slots a packet can take is $\phi+1$, so we define $\Delta(t)$ as the number of additional slots it takes beyond this minimum for packets which arrive at the source at time t. Then if $\Delta(t)\geq\phi$ for any t, our claim must hold. Note that $\Delta(t)=0$ at time t when packets arrive, and it is incremented by one each time a scheduling decision is made that does not schedule those packets.

Assume our claim does not hold, i.e., that $\Delta(t)<\phi$ for all t. First note that packets which arrive at t can allow at most $\Delta(t)$ slots of arrivals behind them to "catch up". We say a slot of arrivals $t+j$ is caught up to t if it is enqueued at the same link as the slot of arrivals t. Each slot of arrivals which catches up to t increments $\Delta(t)$, so arrivals from slot $t+\phi$ cannot be caught up to t under our assumption on $\Delta(t)$. Let $t+j^*\leq t+\phi$ be the first slot which is not caught up to t when packets from slot t reach $T_{\phi+1}^{(i)}$. Because the previous slot of arrivals is caught up to t, it must have been scheduled ϕ times independently of the arrivals from slot $t+j^*$. Therefore, $\Delta(t+j^*)\geq\phi$, which is a contradiction. This proves the result. □

Note that the ORR policy meets the bound in (14) with equality, because the minimum inter-scheduling times are $\underline{k}_0=\phi$ and $\underline{k}_{e,e+1}=1$ for all e. It also formalizes many of the takeaways from the example in Section 2. It shows that round-robin policies do in fact lead to low delay, and that schedule order is important for achieving this. Consider the same activation sets as the ORR policy, but scheduled in the opposite order. One can show that the worst-case packet delay with this new schedule is $\phi\cdot|T^{(i)}|+1$. For interference models where $\phi>1$, this leads to a multiplicative increase in delay. Finally, consider the maximum throughput under the ORR policy. From (10), $\lambda_i^*(ORR(i))=\min_{e\in T^{(i)}}\frac{w_{i,e}}{\phi+1}$. This is not throughput-optimal in general, which highlights the tradeoff between maximizing throughput and minimizing deadlines. When slice widths are equal, however, this tradeoff does not occur.

Corollary 2. *When slice widths are equal across $T^{(i)}$, the ORR(i) policy is both throughput-optimal and deadline-optimal.*

Proof. From Theorem 2, any ORR policy is deadline-optimal, so it remains only to show that it is throughput-optimal. Under equal slice widths, (11) becomes

$$\begin{aligned}&\max_{\pi\in\Pi_c}\min_{e\in T^{(i)}}\bar{\mu}_e^\pi\\&\text{s.t. }\mu^\pi(t)\in M^\phi,\ \forall\,0\leq t\leq K^\pi.\end{aligned}\tag{18}$$

Under ϕ, only one of every $\phi+1$ consecutive slots in $T^{(i)}$ can be activated at once. If $|T^{(i)}|\leq\phi$, only one link can be scheduled at a time so let $\phi=|T^{(i)}|-1$ without loss of generality. Then,

$$\sum_{e=j}^{j+\phi}\mu_e^\pi(t)\leq 1,\ \forall\,t\geq 0,\ 0\leq j<|T^{(i)}|-\phi,\tag{19}$$

and any $\pi\in\Pi_c$, where we slightly abuse notation to denote $T_j^{(i)}$ as j in the subscript. Averaging these constraints over each scheduling period,

$$\sum_{e=j}^{j+\phi}\bar{\mu}_e^\pi\leq 1,\ \forall\,0\leq j<|T^{(i)}|-\phi,\tag{20}$$

and finally summing over these constraints,

$$\begin{aligned}&\bar{\mu}_0+\bar{\mu}_{-1}+2(\bar{\mu}_1+\bar{\mu}_{-2})+\cdots+\phi(\bar{\mu}_{\phi-1}+\bar{\mu}_{-\phi})\\&\qquad+(\phi+1)\sum_{j=\phi}^{|T^{(i)}|-\phi-1}\bar{\mu}_j\leq|T^{(i)}|-\phi,\end{aligned}\tag{21}$$

where we again slightly abuse notation to denote $T_{|T^{(i)}|-j}^{(i)}$ as $-j$. Because the sum relaxes the constraints, this is a necessary but not sufficient condition for a feasible policy.

Counting terms, there are $(\phi+1)(|T^{(i)}|-\phi)$ activation rates summed in this expression. Under the ORR policy, each link e has $\bar{\mu}_e^{ORR(i)}=\frac{1}{\phi+1}$, so summing over all of them shows that the bound in (21) is tight under the ORR policy. Because the bound is a necessary condition for feasibility, any policy for which it is tight must be a solution to (18). This can be shown because increasing $\bar{\mu}_e$ for any e causes a decrease in $\bar{\mu}_{e'}$ for some e', which strictly decreases the objective in (18). Therefore, ORR must be throughput-optimal. □

4 FEASIBILITY UNDER DIFFERENT INTERFERENCE MODELS

Having characterized bounds on the feasibility region and the structure of throughput- and deadline-optimal policies under general interference, we now examine three specific interference models and derive exact results for these cases.

4.1 No Interference

When there is no interference, the system is equivalent to a wired network. In this case, every link can be activated at each slot, so there is no scheduling policy. Therefore, $\bar{\mu}_e=1$ for all e, and $\lambda_i^*=\min_{e\in T^{(i)}}w_{i,e}$. Furthermore, this is equivalent to an ORR policy with $\phi=0$, so $\tau_i^*=|T^{(i)}|$. The feasible region is then defined as

$$\Lambda_i^0\triangleq\{(\lambda_i,\tau_i)\mid 0\leq\lambda_i\leq\min_{e\in T^{(i)}}w_{i,e},\tau_i\geq|T^{(i)}|\}.\tag{22}$$

Furthermore, because there is no scheduling policy in this case, the general achievable region $\Lambda^0=\cup_{T^{(i)}\in\mathcal{T}}\Lambda_i^0$.

4.2 Total Interference

Next consider the total interference scenario, where only one link can be active at a time. From Theorem 2, $\tau_i^*=|T^{(i)}|+\phi=2|T^{(i)}|-1$

under an ORR(i) policy, where recall that $\phi = |E| - 1 = |T^{(i)}| - 1$ for a single flow. Furthermore, $\bar{\mu}_e^{ORR(i)} = \frac{1}{|T^{(i)}|}$ for all e.

Because the main focus of this work is on primary interference, we only briefly list the properties of throughput optimality for the total interference case, and we omit proofs due to space constraints. Similar to what we will see in the primary interference case, a throughput-optimal policy π has a maximum throughput of

$$\lambda_i^*(\pi) = \frac{1}{|T^{(i)}|} H(\mathcal{W}), \tag{23}$$

for a set of slice widths $\mathcal{W}$, and where $H(\cdot)$ represents the harmonic mean.

The achievable deadlines under a throughput-optimal policy are hard to characterize, but are on the order of $O(|T^{(i)}|^2)$. One can see this by examining the deadline condition from Corollary 1, and dividing both sides by λ_i. Plugging in the expression for λ_i^* gives the necessary and sufficient condition

$$\left\lceil \frac{|T^{(i)}|}{H(\mathcal{W})} \sum_{e \in T^{(i)}} Q_{i,e}(t) \right\rceil \leq \tau_i, \tag{24}$$

where the ceiling is taken because deadlines are necessarily integer-valued. It is easy to see that this bound is $O(|T^{(i)}|^2)$.

4.3 Primary Interference

Finally, we consider primary interference, which is the main focus of this work, and begin by characterizing the deadline-optimal point. From Theorem 2, the minimum achievable deadline is $\tau_i^* = |T^{(i)}| + \phi = |T^{(i)}| + 1$ under an ORR(i) policy. Furthermore, this policy can support a maximum throughput of $\lambda_i^*(ORR(i)) = \frac{1}{2} \min_{e \in T^{(i)}} w_{i,e}$.

A throughput-optimal policy can be derived from (11) 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

$$\begin{aligned} &\max_{\pi \in \Pi_c} \min_{e \in T^{(i)}} \bar{\mu}_e^\pi w_{i,e} \\ &\text{s.t. } \bar{\mu}_e^\pi + \bar{\mu}_{e+1}^\pi \leq 1, \ \forall\, 0 \leq e < |T^{(i)}| - 1, \end{aligned} \tag{25}$$

which can be solved exactly.

Theorem 3. *Let π be any throughput-optimal policy on $T^{(i)}$ under primary interference constraints. Then it has a maximum throughput of*

$$\lambda_i^*(\pi) = \min_{(e,e+1) \in T^{(i)}} \frac{w_{i,e} w_{i,e+1}}{w_{i,e} + w_{i,e+1}} \tag{26}$$

Proof. As noted previously, the interference constraint for a single route under primary interference is $\bar{\mu}_e + \bar{\mu}_{e+1} \leq 1$ for all $(e, e+1)$, and from Lemma 1, $\lambda_i^*(\pi) = \min_{e \in T^{(i)}} \bar{w}_{i,e}^\pi$. Then the maximum throughput which can pass through $(e, e+1)$ is the solution to

$$\begin{aligned} &\max_{\bar{\mu}_e, \bar{\mu}_{e+1}} \min\{\bar{\mu}_e w_{i,e}, \bar{\mu}_{e+1} w_{i,e+1}\} \\ &\text{s.t. } \bar{\mu}_e + \bar{\mu}_{e+1} \leq 1. \end{aligned} \tag{27}$$

Because $\bar{\mu}_e$ and $\bar{\mu}_{e+1}$ are continuous, the solution occurs when $\bar{\mu}_e + \bar{\mu}_{e+1} = 1$ and $\bar{\mu}_e w_{i,e} = \bar{\mu}_{e+1} w_{i,e+1}$. Solving this set of equations yields

$$\bar{\mu}_e = \frac{w_{i,e+1}}{w_{i,e} + w_{i,e+1}}, \ \bar{\mu}_{e+1} = \frac{w_{i,e}}{w_{i,e} + w_{i,e+1}}, \tag{28}$$

and the maximum throughput which can pass through the pair of links is

$$\bar{\mu}_e w_{i,e} = \bar{\mu}_{e+1} w_{i,e+1} = \frac{w_{i,e} w_{i,e+1}}{w_{i,e} + w_{i,e+1}}. \tag{29}$$

Now define the pair of links $(e^*, e^* + 1)$ which minimize this quantity as the bottleneck link pair, and $\lambda_i^*(\pi)$ as the bottleneck throughput. Each link in $T^{(i)}$ belongs to two pairs, except the first and last link on the route. Define a set of throughput-optimal activations as follows. For each link e, find the two throughput-optimal activation rates from (28) belonging to its two link pairs, and assign the smaller of these to e, so that

$$\bar{\mu}_e^\pi = \min\left\{ \frac{w_{i,e-1}}{w_{i,e} + w_{i,e-1}}, \frac{w_{i,e+1}}{w_{i,e} + w_{i,e+1}} \right\}. \tag{30}$$

The endpoint links on the route are assigned the value of (28) for their single link pair. Then each link supports at least as much throughput as $\lambda_i^*(\pi)$, because by definition this is the bottleneck throughput, and this set of activations is guaranteed to meet scheduling constraints because each activation pair in (28) is a feasible solution to (27), and each link in the pair either uses this rate or a strictly smaller one. This completes the proof. □

Given a feasible set of activation rates under primary interference, polynomial-time algorithms exist to find a corresponding feasible schedule [17]. Denote the set of throughput-optimal rates (30) as $\bar{\mu}^{\pi^*}$, and the corresponding maximum throughput as λ_i^*. Finally, assume that a feasible schedule is found and let the length of the schedule be K^{π^*}. In order to ensure activation rates are met and the number of activations $\eta_e^\pi = \bar{\mu}_e^\pi K^\pi$ is integer, K^π must be at least as large as the least common multiple of the denominators of the activation rates, which for arbitrary activation rates can become arbitrarily large.

While this policy is guaranteed to be throughput-optimal, 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 2. For non-uniform slice widths, however, $\bar{\mu}_e^{\pi^*} < \frac{1}{2}$ for at least one link e, and the lower bound $\tau_i^*(\pi^*) \geq |T^{(i)}| + 1$ is not tight. In this case, a deadline-minimizing policy that supports a throughput of λ_i^* is the solution to the following mixed-integer program, which seeks to minimize the equivalent deadline characterization from

Corollary 1.

$$\min_{\pi \in \Pi_c} \max_{0 \le t \le \hat{K}} \frac{1}{\lambda_i^*} \sum_{e \in T^{(i)}} Q_{i,e}(t) \tag{31a}$$

$$\text{s.t. } \mu_e^\pi(t) + \mu_{e+1}^\pi(t) \le 1, \; \forall\, 0 \le e < |T^{(i)}|, \; t \ge 0, \tag{31b}$$

$$Q_{i,0}(t+1) = Q_{i,0}(t) + \lambda_i^* - \min\{Q_{i,0}(t) + \lambda_i^*, w_{i,0}\}\mu_0^\pi(t), \forall\, t \ge 0, \tag{31c}$$

$$Q_{i,e}(t+1) = Q_{i,e}(t) + \min\{Q_{i,e-1}(t), w_{i,e-1}\}\mu_{e-1}^\pi(t) - \min\{Q_{i,e}(t), w_{i,e}\}\mu_e^\pi(t), \forall\, 0 < e < |T^{(i)}|, t \ge 0 \tag{31d}$$

$$\mu_e^\pi(t) \in \{0,1\}, \; \forall\, e \in T^{(i)}, \; t \ge 0, \tag{31e}$$

$$Q_{i,e}(t_0 + K^\pi) = Q_{i,e}(t_0), \; \forall\, e \in T^{(i)}, \tag{31f}$$

$$t_0 + K^\pi \le \hat{K}, \tag{31g}$$

$$t_0, K^\pi \in \mathbb{Z}^+. \tag{31h}$$

Here $\hat{K}$ is assumed to be some reasonable upper bound on the time before the network ramps up and completes one scheduling period, and $\mathbb{Z}^+$ is taken to be the set of positive integers strictly greater than zero. Note that the activation rates and schedule length are not fixed, which allows us to optimize over all schedules which support a throughput of λ_i^*. Furthermore, the constraint (31b) ensures that interference constraints are met, (31e) ensures that scheduling decisions are binary in each slot, (31c) and (31d) ensure that queue evolutions are correct, and (31f) ensures the resulting policy is in Π_c. Implicit in this formulation is that queue sizes are initialized to zero.

The constraints (31c) and (31d) are nonlinear because of the queue evolution equations, but can be linearized in the following manner [27]. Define $\psi_{i,e}(t)$ as an auxiliary variable for each e and time step t, and replace (31c) and (31d) with

$$\begin{aligned}
&Q_{i,0}(t+1) = Q_{i,0}(t) + \lambda_i^* - \psi_{i,0}(t), \; \forall\, t \ge 0,\\
&Q_{i,e}(t+1) = Q_{i,e}(t) + \psi_{i,e-1}(t) - \psi_{i,e}(t), \; \forall\, 0 < e < |T^{(i)}|, t \ge 0,\\
&\psi_{i,e}(t) \le w_{i,e}\mu_e(t), \; \forall e \in T^{(i)}, t \ge 0,\\
&\psi_{i,e}(t) \le Q_{i,e}(t), \; \forall e \in T^{(i)}, t \ge 0,\\
&\psi_{i,e}(t) \ge 0, \; \forall e \in T^{(i)}, t \ge 0.
\end{aligned} \tag{32}$$

Then, when $\mu_e(t) = 0$, $\psi_{i,e}(t) = 0$, and when $\mu_e(t) = 1$, $\psi_{i,e}(t) \le \min\{w_{i,e}, Q_{i,e}(t)\}$. Augmenting the objective function to be

$$\min_{\pi \in \Pi_c} \max_{0 \le t \le \hat{K}} \left\lceil \frac{1}{\lambda_i^*} \sum_{e \in T^{(i)}} Q_{i,e}(t) \right\rceil - \sum_{e \in T^{(i)}} \sum_{t=0}^{\hat{K}} \psi_{i,e}(t) \tag{33}$$

ensures that the upper bound on $\psi_{i,e(t)}$ is always drive to equality. We can also linearize (31f) by introducing $\hat{K}$ binary variables, $\{y_1, \dots, y_{\hat{K}}\}$, indicating whether t_0 is equal to that value. We use the same approach with variables $\{z_1, \dots, z_{\hat{K}}\}$ for the quantity $t_0 + K^\pi$. Then (31f) can be replaced with

$$\sum_{t=0}^{\hat{K}} y_t = 1, \; \sum_{t=0}^{\hat{K}} t \cdot y_t = t_0, \tag{34a}$$

$$\sum_{t=0}^{\hat{K}} z_t = 1, \; \sum_{t=0}^{\hat{K}} t \cdot z_t = t_0 + K^\pi, \tag{34b}$$

$$\sum_{t=0}^{\hat{K}} y_t Q_{i,e}(t) = \sum_{t=0}^{\hat{K}} z_t Q_{i,e}(t), \; \forall\, e \in T^{(i)}, \tag{34c}$$

$$y_t \in \{0,1\}, \; z_t \in \{0,1\}, \; \forall\, 0 \le t \le \hat{K}, \tag{34d}$$

which are linear except for the bilinear constraint (34c). Upper bounding the queue sizes with some $\hat{Q}$ and introducing additional auxiliary variables ζ and ν, this can be linearized by

$$\sum_{t=0}^{\hat{K}} \zeta_{i,e}(t) = \sum_{t=0}^{\hat{K}} \nu_{i,e}(t), \; \forall\, e \in T^{(i)}, \tag{35a}$$

$$\zeta_{i,e}(t) \le y_t \hat{Q}, \; \zeta_{i,e}(t) \ge 0, \tag{35b}$$

$$\zeta_{i,e}(t) \le Q_{i,e}(t), \; \zeta_{i,e}(t) \ge Q_{i,e}(t) - (1 - y_t)\hat{Q}, \tag{35c}$$

$$\nu_{i,e}(t) \le z_t \hat{Q}, \; \nu_{i,e}(t) \ge 0, \tag{35d}$$

$$\nu_{i,e}(t) \le Q_{i,e}(t), \; \nu_{i,e}(t) \ge Q_{i,e}(t) - (1 - z_t)\hat{Q}, \tag{35e}$$

where (35b) - (35e) hold for all $e \in T^{(i)}$ and $0 \le t \le \hat{K}$. The resulting optimization is equivalent to (31). Furthermore, because all constraints have been linearized, it is now a mixed-integer linear program (MILP) with $(2+|T^{(i)}|)\hat{K}$ binary variables, $4|T^{(i)}|\hat{K}$ continuous variables, and $13|T^{(i)}|\hat{K}$ constraints, up to constant additive terms. This can be solved with Gurobi or another mixed-integer optimization solver, but does not scale efficiently because the size of the program, including the number of binary variables, grows linearly in $\hat{K}$.

The complexity of this problem illustrates the difficulty of minimizing deadlines for general policies even in this simplest case of a single route. Motivated by this, we seek to find bounds on minimum deadlines for general policies, and start by introducing the concept of resource-minimizing slices.

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 $w_{i,e} = \frac{\lambda_i}{\mu_e^\pi}$ for all $f_i \in \mathcal{F}$ and $e \in T^{(i)}$.*

This definition follows from Lemma 1, because no smaller slice widths can support $\mathcal{F}$. Now consider a complementary route $\tilde{T}^{(i)}$ to that in (31), with the same λ_i^* but where slice widths are resource-minimizing. Any achievable deadline in this complementary network must also be achievable in the original network under the same policy, because larger slices can only cause packets to be delivered faster. We will show that we can bound minimum deadlines in this complementary network, beginning with the following result.

Lemma 3. *If a policy $\pi \in \Pi_c$ supports a set of flows $\mathcal{F}$ under primary interference, and slices are resource-minimizing, then for every $f_i \in \mathcal{F}$ and $e \in T^{(i)}$, there exists at least one slot $t_{i,e}$ in each scheduling period where $Q_{i,e}(t_{i,e}) = 0$.*

Proof. Under resource-minimizing slices, $\bar{w}_{i,e}^{\pi} = \lambda_i$ for all $f_i \in \mathcal{F}$ and $e \in T^{(i)}$ by definition. Then the total number of f_i packets served in each period is at most $\eta_e^{\pi} w_{i,e} = \bar{w}_{i,e}^{\pi} K^{\pi} = \lambda_i K^{\pi}$.

Every e after the first link in $T^{(i)}$ sees at most $\lambda_i K^{\pi}$ arrivals per scheduling period, because this is the most packets which can be served at the previous link in the route. Furthermore, the first link in the route sees exactly this number of arrivals per scheduling period, so this holds for all links. Now assume that $Q_{i,e}(t) = 0$ at some time t, and let $t_{i,e}(t)$ be the first time after t when any number of packets have arrived and been served, and the queue is empty again, so that $Q_{i,e}(t_{i,e}(t)) = 0$. Note that this always occurs in a slot immediately after link e is served.

We claim that $t_{i,e}(t) \le t + K^{\pi}$ for any t. To see this, recall that link e serves at most $\eta_e^{\pi} w_{i,e}$ packets per period, but that any policy in Π_c is work conserving, so it always serves the smaller of $w_{i,e}$ and its queue size when it is scheduled. Let t' be the last time link e is scheduled before time $t + K^{\pi}$, or equivalently the η_e^{π}-th time it is scheduled after time t. Then, either there are more than $w_{i,e}$ packets in link e's queue the first $\eta_e^{\pi} - 1$ times it is scheduled after time t, or the queue is emptied in one of these scheduling events, and under the primary interference assumption, no packets can be served at the previous link and arrive at link e in the same slot. Therefore, in the latter case, $t_{i,e}(t)$ is the slot immediately following when it is emptied. In the former case, this implies that link e serves $(\eta_e^{\pi} - 1)w_{i,e}$ packets in the interval $[t, t')$, and because at most $\eta_e^{\pi} w_{i,e}$ packets can arrive during that interval, the queue must be emptied when it is scheduled at t', and $t_{i,e}(t) = t' + 1 \le t + K^{\pi}$.

Now let $\tilde{t} = t_{i,e}(t)$, and by the same argument, there must exist a $t_{i,e}(\tilde{t}) \le \tilde{t} + K^{\pi}$ where packets have arrived and been served, and $Q_{i,e}(t_{i,e}(\tilde{t})) = 0$. By induction, this holds for any scheduling period K^{π} if $Q_{i,e}(t) = 0$ at some t. Because queues are initialized to zero, this completes the induction step, and the proof is complete. Note that after some t_0, when the network has reached steady state, $t_{i,e}$ will occur at regular intervals from Theorem 1, but that these times are not guaranteed to coincide for different flows. □

This leads to the following result.

Theorem 4. *Let $\pi \in \Pi_c$ be any admissable policy under primary interference constraints. Then the smallest deadline that π can meet is bounded by*

$$\tau_i^*(\pi, \lambda_i) \le K^{\pi} \sum_{e \in T^{(i)}} (1 - \bar{\mu}_e^{\pi}) + 1 \tag{36}$$

for any feasible throughput $0 < \lambda_i \le \lambda_i^*(\pi)$, and slice widths $w_{i,e} \ge \frac{\lambda_i}{\bar{\mu}_e^{\pi}}$. Moreover, there exists at least one policy where this bound is tight, and many policies where τ_i^* grows linearly with K^{π}.

Proof. Assume without loss of generality that slices are resource-minimizing. Allowing slice widths to be larger than this can only decrease the deadline bound, so this assumption is not only non-restrictive but necessary. From Lemma 3, under resource-minimizing slices there exists a slot for each flow in each scheduling period where $Q_{i,e}^{\pi}(t) = 0$. Without loss of generality, consider flow f_i and let $t_{i,e}$ be any such time when $Q_{i,e}$ is empty. Under a fixed set of activation rates $\bar{\mu}^{\pi}$, link e is activated exactly $\eta_e^{\pi} = \bar{\mu}_e^{\pi} K^{\pi}$ times per scheduling period. Therefore, it must be activated η_e^{π} times in the interval $t_{i,e} \le t < t_{i,e} + K^{\pi}$.

Assume that $t \ge t_0$, so the network has already reached steady-state and $Q_{i,e}(t_{i,e} + K^{\pi}) = Q_{i,e}(t_{i,e}) = 0$. This is non-restrictive because before t_0 there are fewer packets in each queue. One can simply add dummy packets to ensure that $Q_{i,e}(t) = Q_{i,e}(t + n_t K^{\pi})$ for some integer n_t such that $t + n_t K^{\pi} \ge t_0$ for all $t < t_0$. Then in each scheduling period, exactly $\eta_e^{\pi} w_{i,e} = \lambda_i K^{\pi}$ packets arrive and are served at link e, so it always serves a full slice of packets. This can be shown by induction and the fact that $Q_{i,e}(t) = Q_{i,e}(t + K^{\pi})$. Because $\lambda_i K^{\pi}$ packets arrive at the source link in a scheduling period, the same number are served by the source link in this period. Therefore the same number arrive at the second link, and so the same number are served at the second link, and so on for each link. Because link e is scheduled η_e times and serves a full $w_{i,e}$ packets each time, the maximum delay that packets can experience at link e occurs when link e is scheduled the η_e time slots leading up to time $t_{i,e} + K^{\pi}$, regardless of when packets arrived.

Now consider the next link $e + 1$ in the route. It also receives $\lambda_i K^{\pi}$ arrivals and must serve this number of packets in each scheduling period. As shown for link e, the maximum delay packets can experience occurs when it is scheduled in η_{e+1} consecutive slots leading up to time $t_{i,e+1} + K^{\pi}$, where the $\lambda_i K^{\pi}$ packets from link e arrive in the interval $t_{i,e+1} \le t < t_{i,e+1} + K^{\pi}$. In particular, the worst-case delay occurs when link $e + 1$ begins receiving packets from link e at time $t_{i,e+1} + 1$. The same holds for each link in $T^{(i)}$, and so the worst-case delay occurs when links are scheduled in blocks such that any link e begins scheduling its block immediately after link $e + 1$ finishes scheduling.

Because packets are scheduled in blocks, we consider the first and last packet in each block. Denote the time the first packet arrives at the source as $t(s_0)$ and the time the last packet in the block arrives at the source as $t(s_{-1}) = t(s_0) + K^{\pi} - 1$. Similarly, denote the time the first packet is delivered to its destination as $t(d_0)$ and the time the last packet is delivered as $t(d_{-1}) \le t(d_0) + K^{\pi} - 1$. The total delay seen by the first packet is therefore larger than the last packet, and because all intermediate packets see a gradient of delay between these two values, the first packet sees the largest delay. It is always the first packet served in a block, so at each link e it sees a delay of $K^{\pi} - \eta_e^{\pi} = K^{\pi}(1 - \bar{\mu}_e^{\pi})$ before being served. Finally, it takes one slot to be delivered to the destination after being served at the last link, so the bound follows.

This bound is tight for at least one policy by construction of the policy just described. We say that there are many policies where τ_i^* grows linearly with K^{π}, because any policy which schedules links in blocks incurs a delay at each link that is linear in K^{π}. In fact, from Lemma 2, any policy with inter-scheduling times that depend on the schedule length incurs such a delay. □

Once again, because K^{π} can grow arbitarily large for general activation rates, the bound in Theorem 4 does not provide strong deadline guarantees. In fact, it shows that there exists 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}. These policies are not pathological examples, but rather any policy with inter-scheduling times that are a function of the schedule length. This includes policies that schedule

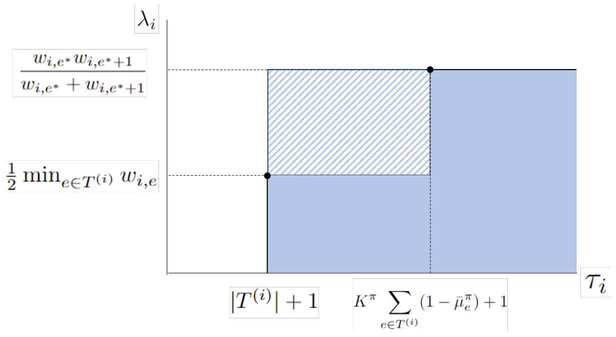

Figure 2: Feasible Region Under Primary Interference

each link in one contiguous block per scheduling period, or simply random orderings of activations which happen to have large inter-scheduling times. To avoid delays that grow with K^π, we introduce policies in the next section based on regular schedules, and 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 region for a flow under primary interference, highlighting the deadline-optimal point we have characterized with exact throughput and deadlines, and the throughput-optimal point in terms of exact throughput and the upper bound on achievable deadlines from Theorem 4. The shaded region between these two operating points is as equally hard to characterize as the throughput-optimal deadline bound, and involves solving (31) for a given value of λ_i to find the associated minimum deadline.

The throughput/deadline tradeoff in this region can be shown to be monotonic, however, and we provide a brief proof sketch. Consider a network with some throughput λ_i, and with minimum deadlines and associated policy π found by solving (31). Now imagine a second identical network with some $\lambda_i' \leq \lambda_i$ under the same policy π. Each packet in the second network sees the same service opportunities as the first, and has the same number or fewer packets ahead of it in its queue at all times. Therefore, it cannot see a larger delay than what is seen by packets in the original network, which is by definition the original minimum deadline. The minimum deadline in the second network must be at least as small as the largest delay that any packet sees, so it must be at least as small as the original minimum deadline. Therefore, decreasing throughput cannot lead to larger deadlines, and the boundary of the region is monotonic between the throughput- and deadline-optimal points.

5 SCHEDULING MULTIPLE FLOWS

We now move to the problem of scheduling multiple flows in a general network to meet service guarantees, drawing on the feasibility results of the previous sections. As a secondary goal, we seek to minimize the total capacity allocated to slices to save as many resources as possible for best-effort traffic. Using the necessary and sufficient condition for meeting deadlines in Corollary 1, and drawing on the mixed-integer program (MIP) in the previous section, we formulate the following MIP for a given set of flows $\mathcal{F}$.

$$\min_{\pi \in \Pi_c, t_0, w} \sum_{f_i \in \mathcal{F}} \sum_{e \in T^{(i)}} w_{i,e} \tag{37a}$$

$$\text{s.t. } \mu(t) \in M^\phi, \ \forall\, 0 \leq t \leq \hat{K}, \tag{37b}$$

$$Q_{i,0}(t+1) = Q_{i,0}(t) + \lambda_i^* - \min\{Q_{i,0}(t) + \lambda_i^*, w_{i,0}\}\mu_{i,0}^\pi(t), \quad \forall\, f_i \in \mathcal{F},\ 0 \leq t \leq \hat{K}, \tag{37c}$$

$$Q_{i,e}(t+1) = Q_{i,e}(t) + \min\{Q_{i,e-1}(t), w_{i,e-1}\}\mu_{e-1}^\pi(t) - \min\{Q_{i,e}(t), w_{i,e}\}\mu_e^\pi(t), \tag{37d}$$

$$Q_{i,e}(t_0) = Q_{i,e}(t_0 + K^\pi), \ \forall\, f_i \in \mathcal{F}, e \in T^{(i)}, \tag{37e}$$

$$\sum_{e \in T^{(i)}} Q_{i,e}(t) \leq \lambda_i \tau_i, \ \forall\, 0 \leq t \leq \hat{K}, f_i \in \mathcal{F}, \tag{37f}$$

$$\sum_{f_i : e \in T^{(i)}} w_{i,e} \leq c_e, \ \forall\, e \in E, \tag{37g}$$

$$t_0 + K^\pi \leq \hat{K}, \tag{37h}$$

$$K, t_0 \in \mathbb{Z}^+, \tag{37i}$$

Here again $\hat{K}$ is assumed to be a reasonable upper bound on the time before the network ramps up and completes one scheduling period. We further note that (37d) holds for all $0 \leq t \leq \hat{K}, f_i \in \mathcal{F}, e \in T^{(i)}$, and we slightly abuse notation in (37c) to denote the activation for the first link in $T^{(i)}$ at time t as $\mu_{i,0}(t)$.

Theorem 5. *If a feasible solution to* (37) *exists, then the resulting policy $\pi \in \Pi_c$ supports all flows in $\mathcal{F}$, and the resulting slices are resource-minimizing.*

Proof. Any feasible solution to the problem is a feasible policy in Π_c because it satisfies link activation constraints by (37b) and is cyclic by (37e). The deadline constraint (37f) verifies that deadlines are met, and the queue evolution constraints (37c) and (37d) implicitly verify that throughput guarantees are met. The capacity constraint (37g) verifies that slice widths are feasible, and the objective minimizes the sum of slice widths, so the solution is necessarily resource-minimizing. □

Note the similarity of this program to (31), but that there are several distinct differences. In (31) slices are fixed and the objective seeks to find the minimum feasible deadline, while in this program slices are variable and the objective is to minimize slices subject to throughput and deadline constraints. In addition, this program optimizes over all flows in $\mathcal{F}$, while (31) deals with a single flow. Finally, because the network consists of more than a single route in this program, $\mu(t)$ must be a valid activation set from M^ϕ in each slot. Let $M_{e,j}^\phi$ be a binary variable indicating whether link e belongs to activation set j, and let $m_j^\pi(t)$ be a binary variable indicating whether activation set j is scheduled at time t. Then (37b) can be replaced by

$$\begin{aligned} &\mu_e^\pi(t) = \sum_{j=1}^{|M^\phi|} m_j^\pi(t) M_{e,j}^\phi, \ \forall\ e \in E, 0 \le t \le \hat{K}, \\ &\sum_{j=1}^{|M^\phi|} m_j^\pi(t) = 1, \ \forall\ 0 \le t \le \hat{K}, \\ &m_j^\pi(t) \in \{0,1\}, \ \forall\ 1 \le j \le |M^\phi|, \ 0 \le t \le \hat{K}. \end{aligned} \tag{38}$$

Using the same approach taken for (31), the nonlinearities in this program can be linearized, resulting in a MILP with $(2+|E|+|M^\phi|)\hat{K}$ binary variables, $4|\mathcal{F}||E|\hat{K}$ continuous variables, and $13|\mathcal{F}||E|\hat{K}$ constraints, up to constant additive terms. Once again, this does not scale efficiently, because the size of the problem grows with $\hat{K}$, and in particular the number of binary variables grows with $|M^\phi|\hat{K}$. Under primary interference, the number of activation sets, or matchings, is exponential in $|E|$. This makes solving (37) quite difficult for networks with more than a handful of links, so in the rest of this work we investigate alternative approaches to the problem.

5.1 Delay Deficit

Motivated by the complexity of optimizing over all policies in Π_c, and having seen that many such policies suffer from large delays, we seek to restrict the class of policies we consider to be a subset of Π_c with tighter upper bounds on delay. Then showing a feasible policy π exists in this smaller class is a sufficient condition for meeting any deadline larger than the upper bound on delay associated with π. In particular, we want to minimize the gap between this upper bound and the lower bound on packet delays from Lemma 2. Analogous to this lower bound, which depends on the minimum inter-scheduling time along a route, we show an upper bound dependent on the *maximum* inter-scheduling time.

In particular, we want to minimize the gap that exists between the lower bound on packet delays from Lemma 2 and the upper bound in this subclass of Π_c. The lower bound depends on the minimum inter-scheduling time along a route, and we will show an upper bound depenedent on the *maximum* inter-scheduling time.

First 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. Define the age of packet l, i.e., the time since it arrived in the network, at time t as $a^l(t)$, and without loss of generality let packet l belong to f_i. Denote the delay quota for link e under policy $\pi \in \Pi_c$ as σ_e^π.

The delay deficit of a packet l currently enqueued at $T_j^{(i)}$ at time t is defined as

$$\delta_{i,j}^l(t) \triangleq a^l(t) - \sum_{j'=0}^{j-1} \sigma_{i,j'}^\pi, \tag{39}$$

where we slightly abuse notation to let $\sigma_{i,j}$ denote the delay quota of $T_j^{(i)}$. This quantity evolves in the following way. When packet l arrives at the source, $\delta^l(t) = 0$. Then, it is incremented by one in each subsequent slot until it reaches its destination, and it is decremented by σ_e^π whenever it is served at link e. A negative delay deficit signifies that a packet has spent less than its allocated time on its route thus far, so it 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.

Using the delay deficit concept and under certain conditions on slice widths, the end-to-end delay that a packet experiences can be bounded by the sum of delay quotas along its route. In particular, it leads to the following result.

Theorem 6. *Let $\pi \in \Pi_c$ be any admissable policy, and let slice widths $w_{i,e} = \lambda_i \overline{k}_e^\pi$ for all $f_i \in \mathcal{F}$ and $e \in T^{(i)}$. Then the smallest deadlines that π can meet are bounded by*

$$\tau_i^*(\pi, \lambda_i) \le \sum_{e \in T^{(i)}} \overline{k}_e^\pi, \tag{40}$$

for any feasible throughput λ_i, and all $f_i \in \mathcal{F}$.

Proof. Define the delay quota at each link e as $\sigma_e^\pi = \overline{k}_e^\pi$. Then the bound in (40) becomes $\tau_i^*(\pi, \lambda_i) \le \sum_{e \in T^{(i)}} \sigma_{i,e}^\pi$.

Under these delay quotas, we claim that if a packet l arrives at link e at time t and $\delta^l(t) \le 0$, then at whatever time t' it arrives at link $e+1$, $\delta^l(t') \le 0$. To see this, assume each queue has a counter which tracks the number of packets with strictly positive delay deficit. Our assumption is that packets arrive at link e with a negative delay deficit, so they are not added to the counter on arrival. From (39), a strictly positive delay deficit implies that a packet's age $a^l(t) > \sum_{j=0}^{e-1} \sigma_{i,j}^\pi$ at time t, and because λ_i packets arrive at the source link in each slot, at most λ_i packets can reach this age and be added to the counter in each slot.

Now we claim that when any link e is scheduled, it serves all packets with positive delay deficits, and will show this by induction. Assume that link e is scheduled at time t and it serves all packets with $\delta(t) > 0$. Then the next time it is scheduled at time t', it again must serve all packets with $\delta(t') > 0$. To see this, recall that link e is scheduled at least every $\overline{k}_e^\pi$ slots by definition, so $t' \le t + \overline{k}_e^\pi$. Then at most $\lambda_i \overline{k}_e^\pi$ packets can have a positive delay deficit at t', and because $w_{i,e} = \lambda_i \overline{k}_e^\pi$, all these packets are served. The first time that link e is scheduled, there can be no more than $\lambda_i \overline{k}_e^\pi$ packets in the network, so it necessarily serves all packets with positive delay deficits, and this completes the induction step.

Therefore, every packet l is served the first time $t' - 1$ that link e is scheduled after δ^l becomes positive, so $\delta^l(t'-1) < \overline{k}_e^\pi$ when it is served. This is exactly the delay quota of link e, so when packet l arrives at the next link at time t', $\delta^l(t') \le 0$. By definition, packets arrive at the source link with $\delta^l = 0$, which completes the induction step. Now consider a ficticious exit link at the destination of each flow. Each packet must arrive at this link with a delay deficit at most 0, which means its age is at most $\sum_{e \in T^{(i)}} \sigma_{i,e}^\pi = \sum_{e \in T^{(i)}} \overline{k}_e^\pi$. This completes the proof. □

We note several things about this result. First, it subsumes the worst-case primary interference bound from Theorem 4. In particular, in the proof of Theorem 4, the worst-case scenario is described with a maximum inter-scheduling time $\overline{k}_e^\pi = K^\pi(1 - \bar{\mu}_e^\pi)$ for all links e. Then the bounds are 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. In fact, it is within a factor of $\max_{e\in T^{(i)}} \overline{k}_e^\pi$ from the deadline-optimal lower bound for an isolated flow in Theorem 2.

The third thing to note is that, while this policy can significantly improve the deadline bound, it is not resource-minimizing in general. The amount of additional slice capacity used to satisfy the delay deficit requirement is

$$\Delta w_{i,e}(\pi) \triangleq \lambda_i\big(\overline{k}_e^\pi - \frac{1}{\bar{\mu}_e^\pi}\big) \geq 0,\ \forall\, f_i \in \mathcal{F},\ e \in T^{(i)}, \tag{41}$$

which is used in the following result.

Corollary 3. *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*

$$\begin{aligned} &\sum_{e\in T^{(i)}} \overline{k}_e^\pi \leq \tau_i,\ \forall\, f_i \in \mathcal{F},\\ &\sum_{f_i:e\in T^{(i)}} \lambda_i \overline{k}_e^\pi \leq c_e,\ \forall\, e \in E,\\ &\frac{1}{\bar{\mu}_e^\pi} \leq \overline{k}_e^\pi \in \mathbb{Z}^+,\ \forall\, e \in E. \end{aligned} \tag{42}$$

Furthermore, when $\underline{k}_e^\pi = \overline{k}_e^\pi = \frac{1}{\bar{\mu}_e^\pi}$ for all $e \in E$, slices are resource-minimizing and deadline bounds are minimized for fixed activation rates. If this holds, we say that π is a regular schedule.

Proof. The sufficient conditions come from the deadline bound in Theorem 6 and link capacity constraints. If both of these are satisfied, then π is guaranteed to support $\mathcal{F}$ by Theorem 6. To show the second part, note that by definition, (40) holds with $\Delta w_{i,e}(\pi) = 0$ if and only if $\overline{k}_e^\pi = \frac{1}{\bar{\mu}_e^\pi}$ for all $e \in E$, which is true only when each link is scheduled exactly every $\overline{k}_e^\pi$ slots, and the minimum and maximum inter-scheduling times are the same. □

From this result, we see that to most efficiently utilize the bound in Theorem 6 as a sufficient condition for meeting deadlines, we should focus on regular schedules that meet the conditions in Corollary 3. These depend only on inter-scheduling times, which for regular schedules are the reciprocal of the activation rates. Finally, note that while the constraints in (42) form a MILP, there are only $|E|$ integer variables, so it can be solved efficiently.

While not optimal in general, the deadline upper bound in Theorem 6 is independent of the schedule length and within a constant factor of the deadline-optimal lower bound for a solitary flow. In many cases, therefore, this is sufficient for meeting service guarantees without solving the MILP derived from (37), provided that schedules can be constructed with guarantees on regularity. In the next section, we develop efficient algorithms to construct such schedules.

6 REGULAR LINK SCHEDULING

Many of the results in the previous sections hold for general interference constraints, but to gain meaningful results in this section, while considering a model that is accurate for many real wireless networks, we restrict our analysis to consider only primary interference. As such, we will refer to activation sets in this section as matchings, because activation sets under primary interference are equivalent to link matchings on the graph G. Under these constraints, and motivated by Corollary 3, we seek to find efficient algorithms to generate regular schedules. When speaking of regular schedules, we will refer to the inter-scheduling time of link e as k_e, where $k_e = \overline{k}_e = \underline{k}_e = \frac{1}{\bar{\mu}_e}$ for a regular schedule. While most results in this section are presented in terms of $\bar{\mu}_e$, this relationship between activation rates and inter-scheduling times for regular schedules allows us to equivalently define results in terms of either quantity, and we will occasionally alternate between the two when appropriate.

One hurdle in finding regular schedules under primary interference is that, in general, throughput-maximizing policies assign links to more than one matching [17], so constructing a regular schedule of matchings does not guarantee the schedule is regular for each link. In fact, finding a feasible schedule subject to matching constraints and a given set of inter-scheduling times is a special case of the Periodic Event Scheduling Problem (PESP) [28]. The general formulation of this problem asks if there exists a feasible schedule π for a given schedule length K^π, such that inter-scheduling times for each link and/or pair of links are guaranteed to fall within a specified interval. Setting intervals for each link e to be k_e^π, and intervals for each interfering pair of links (e, e') to be $[1, K^\pi - 1]$ guarantees that links are scheduled regularly and that no interfering links are scheduled at the same time.

Without the matching constraints, a schedule can be found in $O\big((K^\pi)^3\big)$ time, if a feasible schedule exists, but with the matching constraints this is known to be NP complete [12]. To find feasible solutions to this problem efficiently, we design a heuristic algorithm which assigns each link to a single matching in an efficient manner, so that when matchings are scheduled regularly, each link is also scheduled regularly. We refer to such matchings as *unique edge matchings*. Finding regular schedules for unique edge matchings reduces to the PESP without matching constraints, which can be solved in polynomial time. Our algorithm does not guarantee solutions for every set of feasible link activations, but simulation results in Section 7 show that it performs well in practice.

Under unique edge matchings, each link is activated at the rate assigned to its matching. To ensure that the sufficient conditions for meeting service guarantees are met and that inter-scheduling times are regular, let $\bar{\mu}^0$ be the solution to

$$\begin{aligned} \min\ &\sum_{e\in E} \bar{\mu}_e\\ \text{s.t.}\ &(42),\\ &\overline{k}_e = \frac{1}{\bar{\mu}_e}, \end{aligned} \tag{43}$$

and assign it as the initial activation rate $\bar{\mu}_e^0$ of each link e. Let the activation rate of each matching m be $\bar{\mu}_m = \max_{e\in m} \bar{\mu}_e^0$, so that each link is activated at least as much as its initial rate. Then, because only one matching can be activated at a time, a set of unique edge matchings M is feasible if and only if $\sum_{m\in M} \bar{\mu}_m \leq 1$. Given this, we seek to find matchings which solve

$$\begin{aligned} \min_{M'\subseteq M^1}\ &\sum_{m\in M'} \max_{e\in m} \bar{\mu}_e^0\\ \text{s.t. } &m \cap m' = \varnothing,\ \forall\, m, m' \in M'. \end{aligned} \tag{44}$$

Define a *Greedy Matching* (GM) algorithm to solve this problem as follows. Sort the initial rates of each link from largest to smallest and create a matching m_1. Then, beginning with the largest rate, greedily add each link to the matching if it does not interfere with any links already present. Once all links have been considered, create a new matching m_2 and repeat the process for all links which were not added to m_1. Continue until all links have been added to a matching.

While not necessarily optimal, this algorithm is guaranteed to produce a unique edge matching. There are scenarios, however, when assigning each link to a single matching in this manner is overly restrictive. Consider an example of three links with initial activation rates $\bar{\mu}_1^0 = \frac{1}{2}$, $\bar{\mu}_2^0 = \frac{1}{4}$, and $\bar{\mu}_3^0 = \frac{1}{4}$, and assume that links 2 and 3 interfere but neither interferes with link 1. Then the GM algorithm returns an optimal solution to (44), where links 1 and 2 are assigned to the first matching, link 3 to a second matching, and the value of the objective function is 3/4. Now consider a policy where each time link 1 is scheduled, it alternates between scheduling links 2 and 3 along with it. This relaxes the unique matching constraint, but ensures regularity by creating a "super-link" consisting of links 2 and 3, each of which are scheduled at alternating times when link 1 is scheduled, thereby creating a "super-matching". Because $\bar{\mu}_2^0 = \bar{\mu}_3^0 \leq \frac{1}{2}\bar{\mu}_1^0$, this alternating schedule satisfies all activation constraints, and yields a total activation rate of 1/2. We formalize these ideas below.

Definition 5. *A super-link e with inter-scheduling period k_e is a set of links (or super-links) $\{e_1, e_2, \ldots, e_n\}$ such that, for each $e_i \in e$, $e_i \perp e_j$ for at least one other $e_j \in e$, and $k_{e_i} \geq nk_e$.*

Definition 6. *A super-matching m is a set of links and superlinks where, for all pairwise $(e, e') \in m$, $k_e = k_{e'}$, and $e \not\perp e'$.*

We use the notation $e \perp e'$ to say that links e and e' interfere or, if either e or e' is a super-link, that $e_i \perp e_j$ for any $e_i \in e$ and any $e_j \in e'$. We say that $e \not\perp e'$ otherwise. The intuition behind these ideas is that n_e interfering links with inter-scheduling times greater than or equal to $n_e k_e$ can be grouped together into a super-link with inter-scheduling time k_e. Then each time the super-link is scheduled, it schedules each of the links which comprise it in a round-robin fashion, so that the inter-scheduling time of each link is less than or equal to its initial inter-scheduling time. Any two links or super-links which don't interfere and have the same scheduling period can be added to a super-matching, which is always scheduled together. Therefore, regular scheduling of super-matching is still a PESP without matching constraints, and can be solved efficiently.

Lemma 4. *Let $\hat{M}^1$ be the set of all feasible matchings and super-matchings. Then the solution to*

$$\begin{aligned} &\min_{M' \subseteq \hat{M}^1} \sum_{m \in M'} \max_{e \in m} \bar{\mu}_e^0 \\ &\text{s.t. } m \cap m' = \emptyset,\ \forall\, m, m' \in M', \end{aligned} \tag{45}$$

is less than or equal to the solution to (44).

Proof. Because $\hat{M}^1$ contains all the original matchings, $M^1 \subseteq \hat{M}^1$, so (45) is a relaxed version of (44) and will always have a solution at least as small. □

Intuitively, this result makes sense because multiple interfering links can be added to the same super-matching without increasing its activation rate, but in doing so can possibly reduce the sum of activation rates in (45). Based on this result, we expand our GM algorithm to consider the set of unique edge super-matchings $\hat{M}^1$. One could conjecture that by enforcing regular schedules, we are indirectly forcing each link to belong to a single super-matching. Then the solution to (45) would be the optimal solution over all possible matchings without the unique edge constraint. Unfortunately, there are counterexamples which prove this is not the case, under certain conditions on inter-scheduling times and the structure of super-matchings. While beyond the scope of this paper, the specificity of these conditions lead one to believe that in most cases the solution to (45) is close to the optimal solution, and simulation results in Section 7 verify that this is the case.

Our augmented algorithm, which we call the *Greedy Super-Matching* algorithm, first constructs a set of all possible links and super-links $\mathcal{E}$ with activation rates at most equal to $\max_{e \in E} \bar{\mu}_e^0$, where the set of super-links containing link e_i is denoted as $\mathcal{E}_i$. It then starts with the first super-matching and greedily adds each link/super-link if feasible, beginning with the largest activation rates, until no more can be added. The algorithm repeats this process for the next super-matching, and continues until all links/super-links have been added. Naturally, when a link or super-link is added to a super-matching, all other links and super-links containing those links are removed from $\mathcal{E}$ as well. This ensures that each link is added to exactly one super-matching. We formally define the algorithm below.

Let M^{GSM} be the set of super-matchings generated by the Greedy Super-Matching algorithm, and let $\chi'_{GSM}(G) = |M^{GSM}|$ be the number of super-matchings produced by the algorithm. Then the following holds.

Theorem 7. *Let M^* be the optimal set of super-matchings given by* (45). *The GSM algorithm produces a set of super-matching M^{GSM} such that*

$$\sum_{m \in M^{GSM}} \bar{\mu}_m \leq \left(\frac{\chi'_{GSM}(G)}{\Delta(G)} \right) \sum_{m \in M^*} \bar{\mu}_m, \tag{46}$$

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

Furthermore, the GSM algorithm runs in at most $O(|E|^{\rho+1})$ time, where $\rho = \max_{(e,e') \in E} (\bar{\mu}_e^0 / \bar{\mu}_{e'}^0)$.

Proof. Assume M^* is known, and let χ'_{M^*} be the number of matchings in M^*. Arrange the super-matchings in a histogram by their respective values of $\bar{\mu}_m$, from largest to smallest, assigning $\bar{\mu}_m$ to the height of column m. The resulting plot has χ'_{M^*} columns arranged in decreasing order and a total area equal to $\sum_{m \in M^*} \bar{\mu}_m$, which is precisely the optimal solution to (45). Proceed in a similar way with M^{GSM}. By definition, the super-matchings are already sorted by $\bar{\mu}_m$, so the resulting plot has χ'_{GSM} columns arranged in decreasing order, and a total area equal to $\sum_{m \in M^{GSM}} \bar{\mu}_m$.

Let $\bar{\mu}^*_{(i)},\ i \in \{1, \ldots, \chi'_{M^*}\}$ be the value of the i-th column of the (sorted) M^*, and $\bar{\mu}^{GSM}_{(i)},\ i \in \{1, \ldots, \chi'_{GSM}\}$ be the equivalent under GSM. Then we claim that

$$\bar{\mu}^*_{(i)} \geq \bar{\mu}^{GSM}_{(i)},\ \forall\, i \in \{1, \ldots, \chi'_{M^*}\}, \tag{47}$$

Algorithm 1: Greedy Super-Matching (GSM)

```
Input  : Link activation rates $\bar{\mu}$
Output : Set of unique edge matchings $M^{GSM}$
Sort links by $\bar{\mu}_e^0$, smallest to largest, in a list $\tilde{E}$.
for $i = 1, 2, \ldots, |E|$ do
    $e_{i0} \triangleq \{e_i\},\ \bar{\mu}_{e_{i0}} = \bar{\mu}^0_{e_i},\ \mathcal{E}_i \triangleq \{e_{i0}\}$
    for $e \in \mathcal{E}_i$ do
        for $j = 1, 2, \ldots, |E|$ do
            if $e_j \notin e$ and $e_j \perp e$ then
                if $(|e| + 1)\bar{\mu}^0_{e_j} \le \bar{\mu}^0_{|E|}$ then
                    $e' = e \cup e_j,\ \bar{\mu}_{e'} = (|e| + 1)\bar{\mu}^0_{e_j}$
                    Add $e'$ to $\mathcal{E}_i$ with activation rate $\bar{\mu}_{e'}$
                end
                else
                    break
                end
            end
        end
    end
end
Sort $\mathcal{E}$ by activation rate, largest to smallest
Define super-matchings set $M^{GSM}$
while $\mathcal{E}$ is not empty do
    Add super-matching $m$ to $M^{GSM}$
    for $i = 1, 2, \ldots, |\mathcal{E}|$ do
        if $e_i \not\perp m$ then
            $m = m \cup e_i$
            Remove $e_i$ and all links and super-links that
              overlap with $e_i$ from $\mathcal{E}$
        end
    end
end
Return $M^{GSM}$
```

from the greediness of the GSM algorithm. Assume that the optimal solution is also designed by an algorithm constructing super-matchings in order of sorted columns, and at each step i let $E^*_{(i)}$ be the set of links/super-links not yet added to a super-matching. Likewise, let $E^{GSM}_{(i)}$ be the equivalent for GSM. Then, if

$$\max_{e \in E^*_{(i)}} \bar{\mu}^0_e \ge \max_{e \in E^{GSM}_{(i)}} \bar{\mu}^0_e, \tag{48}$$

is true at some step i, then by the greediness of GSM it is also true at step $i + 1$. Both algorithms start with the same set of links/super-links, so it must hold at $i = 1$, which completes the induction step and proves that it holds for all $i \in \{1, \ldots, \min\{\chi'_{M^*}, \chi'_{GSM}\}\}$. Furthermore, because columns are sorted in decreasing order and every link/super-link is added to a super-matching, this implies that each algorithm adds the maximum remaining element in each step, so $\bar{\mu}^*_{(i)} \ge \bar{\mu}^{GSM}_{(i)}$ for all $i \in \{1, \ldots, \min\{\chi'_{M^*}, \chi'_{GSM}\}\}$ as well. Finally, because the area under each histogram is by definition minimized by M^*, $\chi'_{M^*} \le \chi'_{GSM}$ and (47) holds.

Now imagine the horizontal axis of the histogram of M^* is scaled by a factor of $\frac{|E|}{\chi'_{M^*}}$. The area scales by the same factor, and the plot now has nonzero values from 0 to $|E|$ on the horizontal axis. Define this function as $H^*(e)$. Similarly, let the horizontal axis of the histogram of M^{GSM} be scaled, but this time by a factor of $\frac{|E|}{\chi'_{GSM}}$. Once again, the area scales by the same factor and the plot has nonzero values from 0 to $|E|$. Define this function as $H^{GSM}(e)$.

Then $H^*(e) \ge H^{GSM}(e)$ for all $e \in \{1, \ldots, |E|\}$. This follows from (47), the fact that $H^*(e)$ and $H^{GSM}(e)$ are decreasing functions of e, and the fact that $\chi'_{M^*} \le \chi'_{GSM}$, so the scaled columns of H^* are wider than those of H^{GSM}. It follows that the total area under H^* is no less than the area under H^{GSM}, or more explicitly,

$$\frac{|E|}{\chi'_{M^*}} \sum_{m \in M^*} \bar{\mu}_m \ge \frac{|E|}{\chi'_{GSM}} \sum_{m \in M^{GSM}} \bar{\mu}_m. \tag{49}$$

Rearranging terms,

$$\sum_{m \in M^{GSM}} \bar{\mu}_m \le \left(\frac{\chi'_{GSM}}{\chi'_{M^*}}\right) \sum_{m \in M^*} \bar{\mu}_m, \tag{50}$$

and recognizing that $\chi'_{M^*} \ge \chi'(G)$ for any M^*, and $\chi'(G) \ge \Delta(G)$ by Vizing's Theorem [16, Sec. 9.3], the result follows.

Finally, note that compared to the GM algorithm, $\chi'_{GSM} \le \chi'_{GM}$. This can be seen by the greediness of GSM and the fact that it can add super-links. Therefore, $\bar{\mu}^{GSM}_{(i)} \ge \bar{\mu}^{GM}_{(i)}$ for all $1 \le i \le \chi'_{GSM}$, because the total area under *GSM* is guaranteed to be no greater than that under *GM* from Lemma 4. The GM algorithm is nothing more than a greedy edge coloring on the graph G, where the edges are revealed to the algorithm in order of weight. It has been shown that greedy edge coloring uses at most $2\Delta(G) - 1$ colors, independently of the order in which edges are revealed, provided that $\Delta(G) = O(\log |V|)$ [4]. This completes the first part of the proof.

To show the complexity result, note that in each step of the algorithm to construct super-links, it examines each existing super-link and decides whether each remaining link can be added. Because $\rho = \max_{(e,e') \in E}(\bar{\mu}_e / \bar{\mu}_{e'})$, at most ρ links can be added to each super-link, so the maximum number of super-links that can exist is

$$\sum_{n=0}^{\rho} \binom{|E|}{n} \le \sum_{n=0}^{\rho} \frac{|E|^n}{n!} = \sum_{n=0}^{\rho} \frac{\rho^n}{n!}(|E|/\rho)^n \le \sum_{n=0}^{\rho} \frac{\rho^n}{n!}(|E|/\rho)^\rho$$
$$\le (|E|/\rho)^\rho \sum_{n=0}^{\infty} \frac{\rho^n}{n!} = e^\rho (|E|/\rho)^\rho = O(|E|^\rho), \tag{51}$$

for a fixed ρ. For each super-link, the algorithm requires $O(|E|)$ steps, which gives $O(|E|^{\rho+1})$. In the second part of the algorithm, it again goes through each super-link in the worst case, and does this at most $\chi'_{GSM} \le |E|$ times. Therefore, the algorithm has an overall complexity of $O(|E|^{\rho+1})$. □

Thus far, we have shown how to construct a set of unique edge super-matchings M^{GSM}, where link e is scheduled every n_e times that the super-matching m is scheduled, for some integer $n_e > 0$. Then, a regular schedule of super-matchings with inter-scheduling times k_m guarantees a regular schedule of links with inter-scheduling times $n_e k_m$. As mentioned previously, polynomial-time algorithms exist to find such a schedule if it is feasible. In particular, the authors of [24] provide an algorithm to construct a schedule which is *almost regular*, in the sense that inter-scheduling times differ by at most one slot.

This schedule can be found in $O\big((K^\pi)^4\big)$ time by first creating augmented activation rates $\hat{\mu}_m$ such that $\hat{\mu}_m \geq \bar{\mu}_m$ for all $m \in M^{GSM}$, and, when sorted from largest to smallest, each activation rate is an integer multiple of the next. This is referred to as a *step-down* vector of rates. When the sum of these augmented rates $\sum_{m \in M^{GSM}} \hat{\mu}_m \leq 1$, a feasible almost-regular schedule can be constructed. We do not replicate the augmenting algorithm here due to space constraints, but encourage readers to reference Algorithm 2 and the accompanying results in [24]. Using these augmented activation rates, an almost regular schedule can be constructed with the following algorithm, also borrowed from [24].

Algorithm 2: Almost-Regular Schedule Construction (ARSC)

```
Input  : Step-down set of augmented activation rates $\hat{\mu}_m$
         for all $m \in M^{GSM}$
Output : Almost regular schedule $\pi \in \Pi_c$ and rates $\bar{\mu}^\pi$
1  Set $\bar{\mu}_m = \hat{\mu}_m / \sum_m \hat{\mu}_m$ for all $m$, $K^\pi = 1/\bar{\mu}_{|M|}$, and
     $\eta_m = \bar{\mu}_m K^\pi$ for all $m$.
2  Form an empty schedule with length $K = \lceil \frac{K^\pi}{\eta_1} \rceil \eta_1$.
3  Assign slot 1 to $m_1$, along with every $K/\eta_1$ slots after.
4  for $i = 2, 3, \ldots, |M|$ do
5      Set $\pi_0$ as the set of empty slots
6      for $j = 1, 2, \ldots, i-1$ do
7          Set $\pi_j \subseteq \pi_{j-1}$ as the set of slots with the smallest
             elapsed time since a slot assigned to $m_j$.
8      end
9      Assign the first slot in $\pi_{i-1}$ to $m_i$, along with every
         $K/\eta_i$ slots after.
10 end
11 Remove the $K - K^\pi$ unassigned slots from the schedule.
12 return $\pi$
```

Lemma 5. *Given a set of super-matchings M^{GSM} with feasible step-down augmented activation rates $\hat{\mu}_m \geq \bar{\mu}_m$ for all $m \in M$, the ARSC algorithm returns an almost-regular schedule π with activation rates $\bar{\mu}_e^\pi \geq \bar{\mu}_e^0$, and link inter-scheduling times bounded by $\underline{k}_e^\pi = n_e(k_m^\pi - 1)$ and $\overline{k}_e^\pi = n_e k_m^\pi$, where $k_m^\pi = \lceil \frac{1}{n_e \bar{\mu}_e^\pi} \rceil$.*

Furthermore, feasible augmented rates are guaranteed to exist if

$$\sum_{m \in M^{GSM}} \bar{\mu}_m \leq \ln 2. \tag{52}$$

Proof. It is shown in [24] that an almost-regular schedule π is guaranteed to exist given a feasible set of step-down activation rates $\hat{\mu}$, and where $\bar{\mu}_m^\pi \geq \hat{\mu}_m$ for all $m \in M^{GSM}$.

From the definition of an almost-regular schedule, each super-matching m must be scheduled every k_m^π or $k_m^\pi - 1$ slots in a schedule of length K^π, for some k_m^π. Let α be the number of inter-scheduling times equal to k_m^π, and β be the number of times equal to $k_m^\pi - 1$. Then

$$\alpha k_m^\pi + \beta(k_m^\pi - 1) = K^\pi. \tag{53}$$

Based on the activation rate, $\alpha + \beta = K^\pi \bar{\mu}_m^\pi$, and rearranging terms and substituting this into (53) yields

$$K^\pi \bar{\mu}_m^\pi k_m^\pi - \beta = K^\pi. \tag{54}$$

Finally, rearranging terms,

$$k_m^\pi = \frac{K^\pi + \beta}{K^\pi \bar{\mu}_m^\pi}. \tag{55}$$

Without loss of generality, assume that the inter-scheduling time must be equal to k_m^π at least once. If this were not true, we could simply set $k_m^\pi = k_m^\pi - 1$. Then $\beta \leq K^\pi \bar{\mu}_m^\pi - 1$, so maximizing k_m^π over β yields

$$k_m^\pi \leq \frac{K^\pi + K^\pi \bar{\mu}_m^\pi - 1}{K^\pi \bar{\mu}_m^\pi} = 1 + \frac{1}{\bar{\mu}_m^\pi} - \frac{1}{K^\pi \bar{\mu}_m^\pi} < 1 + \frac{1}{\bar{\mu}_m^\pi} = \left\lceil \frac{1}{\bar{\mu}_m^\pi} \right\rceil, \tag{56}$$

because k_m^π is integer-valued. Any non-super-link in m is scheduled each time m is scheduled and has a value of $n_e = 1$, so the result follows. For each link e that belongs to a super-link, $\bar{\mu}_e^\pi = \frac{1}{n_e} \bar{\mu}_m^\pi$, so $k_m^\pi = \lceil \frac{1}{n_e \bar{\mu}_e^\pi} \rceil$. Because link e is only scheduled every n_e activations of m, and m can have inter-scheduling times of either k_m^π or $k_m^\pi - 1$, the inter-scheduling times of link e are bounded by $n_e k_m^\pi$ and $n_e(k_m^\pi - 1)$.

The sufficient condition on activation rates is shown in [24], and we omit the proof here due to space constraints. □

The ARSC algorithm provides the final step in the process of constructing regular schedules to meet service guarantees, and has a complexity of $O\big((K^\pi)^3\big)$. Below we present the full Polynomial-Time Supporting Policy (PTSS) algorithm for efficiently constructing a scheduling policy to support a set of flows. The full algorithm has a complexity of $O\big(|E|^{\rho+1} + (K^\pi)^4\big)$, where $\rho = \max_{(e,e') \in E}(\bar{\mu}_e^0 / \bar{\mu}_{e'}^0)$.

Algorithm 3: Polynomial-Time Supporting Schedule (PTSS)

```
Input  : Set of flows $\mathcal{F}$
Output : Polynomial-time supporting policy $\pi \in \Pi_c$
1  Solve (43) for initial link activations $\bar{\mu}^0$
2  Run the GSM algorithm on $\bar{\mu}^0$ and return a set of
     super-matchings $M^{GSM}$ with activation rates $\bar{\mu}$
3  Run the Augmenting Demand algorithm from [24] on
     $M^{GSM}$ and return a step-down set of activation rates $\hat{\mu}$
4  if $\sum_{m \in M^{GSM}} \hat{\mu}_m \leq 1$ then
5      Run the ARSC algorithm on $\hat{\mu}$ and return an
         almost-regular schedule $\pi^*$ with activation rates $\bar{\mu}^{\pi^*}$
6      Set slice widths to be $w_{i,e} = \lambda_i \overline{k}_e^{\pi^*}$ for all $f_i \in \mathcal{F}$ and
         $e \in T^{(i)}$
7      Return $\pi^*$
8  end
```

Theorem 8. *If the PTSS algorithm returns a feasible policy π^* for a set of flows $\mathcal{F}$, then the policy supports $\mathcal{F}$, thereby meeting all throughput and deadline constraints. Furthermore, each slice $w_{i,e}$ is within a factor of $\frac{1}{k_m^{\pi^*}} \leq n_e \bar{\mu}_e^{\pi^*}$ of being resource-minimizing.*

Proof. To show that π^* supports $\mathcal{F}$, one needs only to show that it satisfies the sufficient conditions in Corollary 3. By definition, the initial link activations $\bar{\mu}^0$ are a solution to (43), and therefore satisfy the conditions. If we can show that $\overline{k}_e^{\pi^*} \leq \overline{k}_e^0$ for all $e \in E$, then the conditions must also be met under π^*.

The initial link activations are by definition regular, so $\overline{k}_e^0 = \frac{1}{\bar{\mu}_e^0}$ is integer for all e. Furthermore, any link e added to a super-link is scheduled every n_e times the super-link is scheduled, which is also by definition regular. Activation rates can only be increased when added to super-links, so

$$\frac{1}{\bar{\mu}_e^0} \geq n_e\Big(\frac{1}{n_e \bar{\mu}_e^{GSM}}\Big) = n_e\Big\lceil \frac{1}{n_e \bar{\mu}_e^{GSM}} \Big\rceil, \tag{57}$$

because of the regularity. The Augmenting Demand algorithm generates some $\hat{\mu}_e \geq \bar{\mu}_e^{GSM}$, and assume the resulting step-down activation are feasible. Then the ARSC algorithm returns π^* with $\bar{\mu}_e^{\pi^*} \geq \hat{\mu}_e$. Therefore, buildling on (57),

$$\overline{k}_e^0 = \frac{1}{\bar{\mu}_e^0} \geq n_e\Big\lceil \frac{1}{n_e \bar{\mu}_e^{GSM}} \Big\rceil \geq n_e\Big\lceil \frac{1}{n_e \bar{\mu}_e^{\pi^*}} \Big\rceil = \overline{k}_e^{\pi}, \tag{58}$$

where the last equality follows from Lemma 5. Therefore, π^* supports the set of flows $\mathcal{F}$.

The algorithm sets slice widths equal to $w_{i,e} = \lambda_i \overline{k}_e^{\pi^*} = \lambda_i n_e k_m^{\pi^*}$. Furthermore, because links are scheduled every $n_e(k_m^{\pi^*} - 1)$ or $n_e k_m^{\pi^*}$ slots, $\frac{1}{\bar{\mu}_e^{\pi^*}} \geq n_e(k_m^{\pi^*} - 1)$ Then, from (41), the excess slice width is

$$\Delta w_{i,e}(\pi^*) \leq \lambda_i(n_e k_e^{\pi^*} - n_e(k_m^{\pi^*} - 1)) = n_e \lambda_i, \tag{59}$$

and the fraction of excess slice width is

$$\frac{\Delta w_{i,e}(\pi^*)}{w_{i,e}} \leq \frac{n_e \lambda_i}{n_e \lambda_i k_m^{\pi^*}} = \frac{1}{k_m^{\pi^*}} \leq n_e \bar{\mu}_e^{\pi^*}. \tag{60}$$

□

6.1 Discussion

The PTSS algorithm provides an efficient, polynomial-time method for constructing a schedule to support a set of flows under primary interference, and general network topologies and service agreements. It allows us to guarantee deadline upper bounds that are a constant factor away from the lower bound, and it only allocates a small fraction of additional capacity to slices beyond the resource-minimizing amount. Of course, we saw in (37) that finding a feasible policy in general is a MILP that grows exponentially in the number of links, so naturally PTSS cannot find a solution for every feasible set of flows that arrives. We are able to solve PTSS in polynomial time by adding restrictions to the problem and taking suboptimal steps.

In particular, the algorithm is able to meet deadline constraints as tight as $\sum_{e \in T^{(i)}} \overline{k}_e \approx \sum_{e \in T^{(i)}} \frac{1}{\bar{\mu}_e}$. If a flow requires deadlines tighter than this bound then the PTSS algorithm will not find a feasible solution, but in practice this bound is generally close to the lower bound in Lemma 2 and provides powerful deadline guarantees.

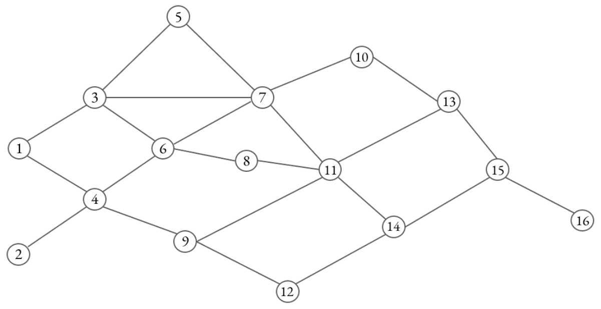

Figure 3: Example Network Diagram

Furthermore, constraining the class of policies to be regular or almost-regular introduces suboptimality, as do the GSM and Augmenting Rates algorithms by restricting links to belong to a single super-matching, and forcing augmented activation rates to be step-down. These conditions are necessary for the ARSC algorithm, but can lead to scenarios where the PTSS algorithm does not return a feasible schedule when one exists. In particular, if the condition in Lemma 5 is not met, the Augmenting Rates algorithm is not guaranteed to return a feasible set of activation rates. This can occur when the initial activation rates are close to the network capacity, and we explore through simulations in the next section how much additional capacity is required in practice to ensure the algorithm returns a feasible schedule.

7 NUMERICAL RESULTS

We simulate our results in this section using the example network in Figure 3, noting that while the diagram shows bidirectional links, we treat each of these as two directional links which interfere, as in the analysis. In each simulation we assume the topology and link capacities are fixed. By definition the PTSS algorithm in the previous section meets service guarantees if it can find a feasible almost-regular schedule, so we first examine how often this occurs.

We generate 32 flows with uniformly random source/destination pairs and fix their routes using shortest path routing. We assume that each flow has an identical throughput λ, and for perspective on our algorithm's performance, we compute the largest value λ^* that λ can take without being subject to deadline constraints, by computing a throughput-optimal set of matchings. Then under varying throughputs and deadlines, we find greedy matchings and solve the PTSS algorithm. Under an arbitrarily small $\lambda = \epsilon$, we expect the algorithm to always achieve a deadline of 60, because the longest path any flow can take is 6 hops, and the network can be colored with 10 colors. Then a simple round-robin policy will ensure the deadline is met.

Beginning with one set of randomly generated flows, we attempt to solve PTSS for $\lambda = \epsilon$, as well as 0.2 and 0.5 of λ^*, and report whether the algorithm finds a feasible solution under varying deadlines, ranging from 30 to 75. This is repeated for 100 sets of random flows, and the average results are plotted in Figure **??**. This figure clearly shows the tradeoff between throughput and deadlines that the algorithm can support. We also plot the average rate of feasible solutions as a function of λ for three different deadlines. Both these

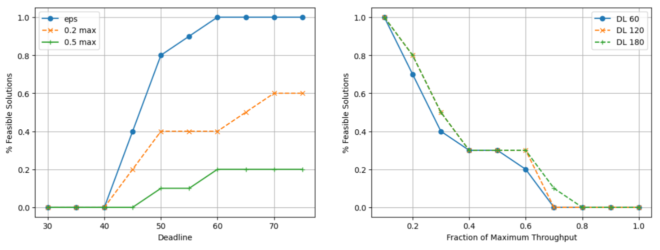

Figure 4: Feasible PTSS Solutions

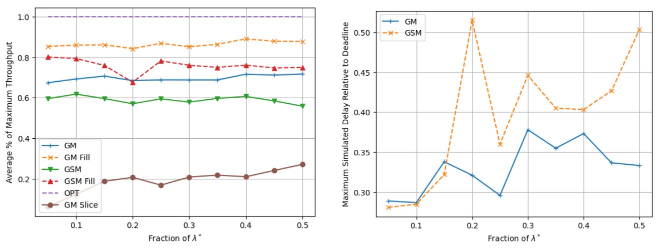

Figure 5: Achievable Throughput and Simulated Packet Delays Under PTSS

results show that, aside from very tight deadlines, the feasibility region of the PTSS algorithm is dominated largely by the achievable throughput.

For rates in the throughput-feasible region, shown to be approximately less than $0.5\lambda^*$, we examine the amount of deadline-constrained traffic the network can support, and what capacity is available for best-effort traffic. We again simulate over 100 random sets of flows, and in each case we compute the total deadline-constrained throughput, represented as the slice capacity in Figure 5, as well as the total throughput of deadline-constrained and best-effort traffic which can be achieved under the GM policy. We also show the achievable throughput under GSM, as well as GM Fill and GSM Fill, where after greedy matchings are constructed, links are added to each matching in a greedy fashion so that each is always maximal. Finally, we show the actual maximum packet delays experienced by flows under both the GM and GSM policies, as a fraction of their guaranteed deadline.

8 CONCLUSION

In this paper, we analyzed the impact of wireless interference and scheduling on service guarantees. We defined the feasible region of throughput and deadline guarantees for a given flow in isolation, and showed that without carefully structuring the order of a scheduling policy, packets can experience large delay and tight deadline guarantees cannot be met. To alleviate this problem, we showed that under regular schedules and minor assumptions on slice capacity, tight deadline guarantees can be made which are independent of the schedule length. We then designed algorithms to construct regular schedules in polynomial time, and showed simulation results supporting our analysis.

REFERENCES

[1] [n. d.]. *Quality of Service (QoS) in 5G Networks.* https://5ghub.us/quality-of-service-qos-in-5g-networks/ Accessed: 03-21-2024.

[2] [n. d.]. *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 Accessed: 03-21-2024.

[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, Rajeev Motwani, and Joseph Naor. 1992. The greedy algorithm is optimal for on-line edge coloring. *Inform. Process. Lett.* 44, 5 (Dec. 1992), 251–253. https://doi.org/10.1016/0020-0190(92)90209-E

[5] 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.

[6] 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. https://doi.org/10.1109/WOWMOM.2009.5282472

[7] 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. https://doi.org/10.1016/j.comcom.2011.02.004

[8] 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. https://doi.org/10.1016/j.comnet.2012.11.021

[9] 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. https://doi.org/10.1145/2684464.2684493

[10] Rene L Cruz. 1991. A calculus for network delay. I. Network elements in isolation. *IEEE Transactions on information theory* 37, 1 (1991), 114–131.

[11] Rene L Cruz. 1991. A calculus for network delay. II. Network analysis. *IEEE Transactions on information theory* 37, 1 (1991), 132–141.

[12] 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. https://doi.org/10.1007/BF01920492

[13] 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.

[14] 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. https://doi.org/10.1109/TNET.2008.2005219

[15] 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.

[16] Jonathan L Gross, Jay Yellen, and Mark Anderson. 2018. *Graph theory and its applications.* Chapman and Hall/CRC.

[17] B. Hajek and G. Sasaki. 1988. Link scheduling in polynomial time. *IEEE Transactions on Information Theory* 34, 5 (Sept. 1988), 910–917. https://doi.org/10.1109/18.21215

[18] I-Hong Hou. 2013. Scheduling heterogeneous real-time traffic over fading wireless channels. *IEEE/ACM Transactions on Networking* 22, 5 (2013), 1631–1644.

[19] I-H Hou, Vivek Borkar, and PR Kumar. 2009. *A theory of QoS for wireless.* IEEE.

[20] 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.

[21] 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.

[22] 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.

[23] Chengzhi Li, Almut Burchard, and Jörg Liebeherr. 2007. A network calculus with effective bandwidth. *IEEE/ACM Transactions on Networking* 15, 6 (2007), 1442–1453.

[24] 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. https://doi.org/10.1109/INFOCOM42981.2021.9488685

[25] 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.

[26] 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.

[27] Garth P McCormick. 1976. Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical programming* 10, 1 (1976), 147–175.

[28] Paolo Serafini and Walter Ukovich. 1989. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics* 2, 4 (1989), 550–581.

[29] 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.

[30] 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.

[31] 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.