Reduced-State, Optimal Scheduling for Decentralized Medium Access Control of a Class of Wireless Networks

Avinash Mohan®, Member, IEEE, Aditya Gopalan®, Member, IEEE, and Anurag Kumar, Fellow, IEEE

Abstract—Motivated by medium access control for resource-challenged wireless Internet of Things (IoT) networks, we consider the problem of queue scheduling with reduced queue state information. In particular, we consider a time-slotted scheduling model with \( N \) wireless links, such that links \( i \) and \( i + 1 \), \( 1 \leq i \leq N - 1 \) cannot transmit together. Our aim in this paper is to study throughput-optimal, and even delay optimal, scheduling policies that require only the empty-nonempty state of the packet queues associated with these links (Queue Nonemptiness Based, or QNB, policies). We focus on Maximum Size Matching (MSM) policies, and provide an analysis of all the QNB-MSM policies for \( N = 3 \), thereby comparing their performance, and revisiting a delay optimal scheduling result. Our study shows that, while scheduling a maximum size matching would seem intuitive, there are important performance differences between different QNB-MSM policies. Further, it is not necessary for a QNB policy to be MSM for it to be throughput optimal. We develop a new Policy Splicing technique to combine scheduling policies for small networks to construct throughput-optimal policies for larger networks, some of which also aim for low delay. For \( N = 3 \) there exists a QNB-MSM policy that is sum-queue length optimal over the entire stability region. We shibboleth, however, that for \( N \geq 4 \), there is no QNB scheduling policy that is sum-queue length optimal over all arrival rate vectors in the capacity region. Our throughput-optimality results rely on two new arguments: a Lyapunov drift lemma specially adapted to policies that are queue length-agnostic, and a priority queueing analysis for showing strong stability. We then extend our results to a more general class of interference constraints that we call cluster-of-cliques (CoC) conflict graphs. We consider two types of CoC networks, namely, Linear Arrays of Cliques (LaCoC) and Star-of-Cliques (SoC) networks. We develop QNB policies for these classes of networks, study their stability and delay properties, and propose and analyze techniques to reduce the amount of state information to be disseminated across the network for scheduling.

Index Terms—Wireless sensor networks, medium access control (MAC) protocols, resource challenged networks, low delay wireless scheduling, Internet of Things.

I. INTRODUCTION

The Internet of Things (IoT) paradigm is expected to make possible applications where vast numbers of devices coexist on a communication network. A typical example is a wireless network comprising low-cost sensors that forward measurements from their locations to a fusion center. Given the variety of applications supported by, and the ubiquitous nature of these networks, for IoT solutions to be viable, the embedded IoT devices will have to cost very little (less than $1, according to some estimates [1]). Such devices will be resource challenged, possess very limited communication and computing capabilities and support little memory. Control policies for such networks will, perforce, need to be simple and not result in excessive overheads.

These constraints are starkly different from those encountered in the transmission of traditional voice and packet data over wireline or cellular wireless networks. In cellular systems, for example, most scheduling decisions come from the base station and hence, control is centralized [2, Chap. 6, 13], but some Quality of Service (QoS) is expected – low packet delay, for instance [3], [4]. In contention access systems such as WiFi, scheduling is distributed but service is best effort [5] or, at best, differentiated – such as with the “Enhanced Distributed Channel Access” [6] and the “Enhanced Distributed Coordination Function” (E-DCF) mechanisms in the IEEE 802.11e standard [7]. In many IoT applications, e.g., condition monitoring or predictive maintenance, there is a need for low overhead distributed scheduling (for the earlier reasons) while also providing QoS.

Consequently, resource allocation techniques developed to handle services such as file transfer and packet-voice might not be appropriate for wireless networks of resource challenged nodes, that need to provide QoS to the applications they are carrying. Most existing medium access protocols and scheduling algorithms suffer from limitations – the WiFi protocol and its Differentiated Services version are simple, but do not really provide any guarantees. Recent standards for the IoT, such as 6TiSCH on the other hand, come with guarantees but require a lot of information exchange and central coordination [8], [9]. Our aim in this paper is to propose decentralized Medium Access Control (MAC) protocols with a focus on low packet delay (i.e., latency) and reduced exchange of control information across the network.

The traditional approach for dynamic resource allocation has been to use backlog or queue length information to opportunistically schedule transmissions. One of the seminal
contributions to scheduling in constrained queueing systems is the work of Tassiulas and Ephremides [10]. These authors modeled a wireless network as a network of queues with pair-wise scheduling constraints (corresponding to wireless interference, half-duplex operation, etc.), and several flows over the network, each with its ingress queue and egress queue. A time-slotted model was considered in [10], with global queue scheduling decisions needing to be made at the beginning of every slot. Scheduling constraints (such as link interference or half-duplex constraints) were modeled by pair-wise scheduling constraints, represented by an interference graph; queues adjacent in the interference graph cannot be scheduled in the same slot. With stochastic arrivals to each flow to be routed from their ingress to egress points, the authors develop MaxWeight a centralized scheduling algorithm which requires the queue lengths of all nodes in every scheduling slot and show that it is throughput-optimal, i.e., it stochastically stabilizes all queues under any stabilizable arrival rate.

Attempts to decentralize MaxWeight include approximations based on message passing between nodes [11], [12], or using queue lengths to modulate backoff parameters in Carrier Sense Multiple Access (CSMA) and ALOHA [13], [14]. Both of these methods, while being throughput-optimal, suffer from poor delay performance. Another method to reduce the amount of information required for scheduling is proposed in [15], where, for two classes of constrained queueing systems, algorithms relying only on the empty-nonempty state of queues is proposed and analyzed for delay performance. Our interest lies in the second half of [15], wherein a scheduling algorithm is proposed for a system of $N$ parallel queues in which adjacent queues cannot be served simultaneously. For such a network, the authors provide a technique to improve the delay performance of a given scheduling policy (assuming one exists). For $N = 3$, the authors provide a policy that is mean delay optimal over the entire stability region. For $N = 4$, Ji et al. [15] provide a policy that heavy-traffic delay optimal. It is not yet clear if it is possible to extend these algorithms to general wireless networks while preserving performance guarantees such as throughput-optimality. In Sec. I-A below, we explain our contributions in greater detail.

A. Our Contributions and Organization

We consider a system of $N$ wireless links (transmitter-receiver pairs). Time is slotted and each transmitter has an independent arrival process of packets embedded at slot boundaries. Only one packet can be transmitted across a link in any slot. There are scheduling constraints (link activation constraints) that constrain which links can simultaneously transmit in any slot. Packets that arrive and cannot be immediately transmitted, wait in a queue at the transmitter.1

When the links are “colocated,” i.e., all links interfere with each other, only one link can be activated in any slot. In a colocated network, for stabilizable arrival rates (see Sec. II), it is known that any policy that transmits a packet from any nonempty queue is not only stabilizing, but is delay optimal in the strong sense of stochastically minimising the sum-queues process (see Defn. 6).

We begin with the system model in Sec. II, wherein we describe the two classes of interference networks considered in this article: “path-graph networks” (Sec. II-B) and “cluster-of-cliques” networks (Sec. II-C). We restrict our study to two subclasses of scheduling policies: Maximum Size Matching (MSM) policies (Sec. III-A) that serve a set of links with the largest number of nonempty non-conflicting queues in every slot, and Queue Nondemptiness based (QNB) policies that use only the empty-nonempty statuses of network queues for scheduling (Sec. III-B).

We then provide a complete characterization of the set of QNB-MSM policies for path-graph networks for the case with $N = 3$ queues (Sec. IV). The fact that the policies we discuss do not require any information about the queues except their empty-nonempty status helps satisfy our reduced state information requirement. We establish several interesting results about (in)stability and delay optimality. Specifically, for $N = 3$, we supply a formal proof that there is a delay optimal policy in the QNB-MSM class of policies, an observation that had been made in [15]. We find that, for $N = 3$, one of the QNB-MSM policies is not even throughput-optimal, thus bringing forth the need for careful design of the scheduler.

In our work, we provide a study of QNB-MSM scheduling policies (explicitly constructing several such policies along the way), and show how scheduling policies for larger networks can be constructed by the novel method of policy splicing. Continuing with path-graph networks, we propose a “policy splicing” technique (see Figures 3 and 4) to combine policies for small networks to construct low delay policies for larger networks (Sections V and V-D). We use this technique to propose QNB-MSM scheduling policies for several such networks. We also provide an in-depth analysis of delay (Sec. VI), culminating in a result that shows that there do not exist delay optimal QNB-MSM policies for such networks with $N \geq 4$ queues (Thm. 18).

We then extend our theory of MSM policies to schedule transmissions over cluster-of-cliques constraint networks (Sec. VII) and also discuss multiple methods to further reduce the amount of state information that has to be exchanged across the network to make these protocols amenable to distributed implementation. We finally use this theory to propose a throughput-optimal protocol, akin to the QZMAC protocol [16], wherein scheduling decisions are taken using only the information about activity on the channel (or lack thereof) that can be sensed by the nodes and will study its performance in detail (Sec. VIII). We then present numerical results (Sec. IX) showing the performance of our proposed policies, and comparisons with standard, high-overhead state-based policies such as the MaxWeight- family [17]. These simulation studies show that MaxWeight and MaxWeight- while guaranteeing queue stability, can perform poorly in delay performance in comparison with QNB-MSM policies.

II. THE SCHEDULING PROBLEM: MODELS AND NOTATION

In Sec. II-A, we describe the general network model, and specify the optimal scheduling problem in Sec. II-A.1. Then, in Sec. II-B and Sec. II-C we restrict the general model to the cases that we provide results for in the remainder of the paper.

There are several interfering links (transmitter-receiver pairs), where each transmitting node has a stream of arriving packets. Time is slotted, and all links are synchronized to the time slots. In each slot, each scheduled link can transmit one packet. Packets that are not transmitted remain in the queues. Thus, we have a discrete time queue scheduling problem that belongs to the general class introduced in [10]. Note, from the preceding discussion, that activating a link in a time slot is the same as serving its associated queue.
A. The General Queue Scheduling Model

We consider a system comprising $N$ queues, where, as mentioned before, each queue models a radio link in a wireless network. The leading edges of time slots are indexed $0, 1, 2, \ldots$. Exogenous arrivals to the queues are embedded at slot boundaries, $t = 0, 1, 2, \ldots$, with the number of packets arriving to Queue $i$ at time $t$ being denoted by the random variable $A_i(t)$. $A_i(t)$ is assumed IID across time and independent across queues and is modelled as a Bernoulli random variable with mean $\lambda_i$ i.e., $P(A_i(t) = 1) = \lambda_i$, $\forall t \geq 1$. However, we will remove this restriction to include batch IID arrivals in Sec. VII. We use $Q(t) = [Q_1(t), \ldots, Q_N(t)]$ to denote the vector of all queue lengths at time $t$. The queue length process is embedded at the beginnings of time slots, so $Q_i(t)$, $\forall t \geq 0$, is measured at $t+$, i.e., just after the arrival. The duration of a slot is assumed to include packet transmission time, the receive-transmit turn around time at the receiver, the MAC layer acknowledgement (ACK) time, and any scheduling overhead. Packet transmissions are assumed to take exactly one time slot and succeed with probability $4$. The random variable, $D_i(t)$, indicating the departure of a packet from Queue $i$ at time $t$, is such that $D_i(t) = 1$ if and only if Queue $i$ is scheduled in slot $t$ and $Q_i(t) > 0$, else $D_i(t) = 0$; here, the departure is assumed to end just before the leading edge of slot $(t + 1)$, i.e., at $(t + 1)$.

The offered service process to Queue $i$, $\{S_i(t), t \geq 0\}$, is defined as follows: $S_i(t) = 1$ whenever Queue $i$ is given access to the channel, so that $D_i(t) = S_i(t)Q_i(t)$, $\forall t \geq 0, 1 \leq i \leq N$. Depending on the interference constraints, it may be possible to serve only a subset of one or more queues in a given slot. For example, (2) gives the constraints for path-graph interference networks and (4) for Star-of-Cliques networks. The vector $S(t) := [S_1(t), \ldots, S_N(t)]$ satisfying the interference constraints is called an activation vector. Thus, for every queue $i$,

$$Q_i(t + 1) = Q_i(t) - D_i(t) + A_i(t + 1) = (Q_i(t) - S_i(t)) + A_i(t + 1), \quad \forall t \geq 0.$$ 

Denote by $\zeta(t) := [\{Q_1(t) > 0\}, \ldots, \{Q_N(t) > 0\}]$ the system's occupancy vector at time $t$, i.e., the empty-nonempty state of each of the $N$ queues. Let $V \subset \{0, 1\}^N$ be the set of all activation vectors. A scheduling policy $\pi := \{\mu_0, \mu_1, \ldots\}$ decides which queues are allowed to transmit in each slot as a function of the available history $\mathcal{H}_t$, which comprises the past states and actions known to the controller, and the current (known) queue state. Specifically, $\mu_t : \mathcal{H}_t \to V$ is an $N \times 1$ vector, and $S_i(t) = \mu_i(t)$. When the schedule depends only on state and not on time, the resulting policies are of the form $\pi = \{\mu, \mu_1, \ldots\}$, and are said to be stationary. We will focus on stationary policies in this article.

1) Performance Metric: By stability of the process $\{Q(t), t \geq 0\}$ we will mean that

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} \mathbb{P}(Q_i(t) > 0) Q_i(t) < \infty. \tag{1}$$

This condition is commonly known as strong stability [18]. A policy that ensures (1) is said to be stabilizing, and an arrival rate vector for which a stabilizing policy exists is said to be stabilizable. The closure of the set of all stabilizable rate vectors is called the throughput capacity region of the network [10], and a policy that is stabilizing for every arrival rate vector in the interior of this region is called throughput-optimal (T.O.). The set of arrival rates that are stabilizable under a given fixed policy is called stability region of the policy.

B. Path Graph Interference Networks

The first system we will study in the subsequent sections is modelled by $N$ parallel queues (see Fig. 1). The scheduling constraints are the same as the second model in Tassiulas and Ephremides 1994 [15], namely that Queue $i$ and Queue $i + 1$ cannot be served simultaneously for $1 \leq i \leq N - 1$. These interference constraints enforce the following rule on the offered service process $S(t)$, $\forall t \geq 0$

$$S_i(t) + S_{i+1}(t) \leq 1, \quad \forall t \geq 0, 1 \leq i \leq N - 1. \tag{2}$$

The conflict graph associated with the system is a called path graph [19], [20]. Standard analysis [10] show that the capacity region of this network is the set

$$\Lambda_N := \{\lambda \in \mathbb{R}_+^N \mid \lambda_i + \lambda_{i+1} \leq 1, \forall 1 \leq i \leq N - 1\}, \tag{3}$$

whose interior, $\Lambda_N^o$, is the set of all stabilizable rate vectors.

C. The Cluster-of-Cliques (CoC) Graph Networks

In the remainder of the paper, we will refer to the conflict graph associated with a collocated network, i.e., a fully connected graph or subgraph, as a clique. The system under consideration comprises multiple cliques and the exact nature of the interference relations across cliques are described in detail below. The number of packets arriving to Queue $j$ in Clique $i$ at time $t$ is denoted by the random variable $A_{i,j}(t)$. As before, $Q_{i,j}(t)$, the backlog of Queue $j$ in Clique $i$ is measured at $t+$, $t \geq 0$, i.e., just after the arrival. Once again, as before, for every $(i, j)$,

$$Q_{i,j}(t + 1) = Q_{i,j}(t) - D_{i,j}(t) + A_{i,j}(t + 1) = (Q_{i,j}(t) - S_{i,j}(t)) + A_{i,j}(t + 1), \quad \forall t \geq 0.$$ 

Depending on the underlying conflict graph, the CoC networks studied in this paper are broadly classified into two classes

Star-of-Cliques networks (SoC): Consider an interference graph consisting of a central fully-connected subgraph (central clique) surrounded by $N - 1$ cliques (see Fig. 2b). In other words, the network's conflict graph consists of $N$ cliques denoted $C_1, \ldots, C_N$, and clique $C_i$ consists of $N_i$ vertices – an arbitrary number of cliques each having arbitrarily many
communication links (queues). Transmissions in $C_1$ interfere with those in all other cliques while the transmissions in $C_i$, $i \geq 2$ interfere with those in $C_1$ only. Coming to the offered service processes, for any two queues $Q_{i,j}$ and $Q_{k,l}$ in the system, the interference constraints enforce the rule

$$S_{i,j}(t) + S_{k,l}(t) \leq 1, \quad \forall t \geq 0, \quad \text{if } i = k, \text{ or } i = 1, \text{ or } k = 1.$$  

(4)

Let $N = \sum_{i=1}^{N} N_i$ denote the total number of queues in the system. The capacity region of this system is given by

$$\Lambda^{(N)}_s := \left\{ \lambda \in \mathbb{R}_+^N : \sum_{j=1}^{N_i} \lambda_{1,j} + \sum_{k=1}^{N_i} \lambda_{i,k} \leq 1, \quad i \in \{2, \cdots, N\} \right\}$$

(5)

(the subscript $s$ highlights the fact that this is the Star-of-Cliques model).

**Linear-Array-of-Cliques (L AoC):** This system consists of $N$ cliques $C_1, C_2, \cdots, C_N$, but unlike the SoC model, all transmissions in $C_{i-1}$ interfere with those in $C_i$, $i \in \{2, \cdots, N\}$ and vice-versa (see Fig. 2a). As in the SoC model, Clique $C_i$ comprises $N_i$ queues and $N = \sum_{i=1}^{N} N_i$ denotes the total number of queues in the system. Since transmissions in adjacent cliques interfere with each other, for the offered service processes of any two queues $Q_{i,j}$ and $Q_{k,l}$ in the system, we have

$$S_{i,j}(t) + S_{k,l}(t) \leq 1, \quad \forall t \geq 0, \quad \text{if } k = i + 1, \forall 1 \leq i \leq N - 1.$$  

(6)

The capacity region of this system is given by

$$\Lambda^{(N)}_t := \left\{ \lambda \in \mathbb{R}_+^N : \sum_{j=1}^{N_i} \lambda_{1,j} + \sum_{k=1}^{N_i} \lambda_{i+1,k} \leq 1, \quad i \in \{1, \cdots, N - 1\} \right\}$$  

(7)

(the subscript $t$ highlights the fact that this is the Linear-Array-of-Cliques model). As before, the vector $S(t) := [S_1(t), \ldots, S_N(t)] \in \{0, 1\}^N$ is called an activation vector if it satisfies the constraints in (4) and (6) in the SoC and L AoC systems, respectively. We now begin our study with path graph interference networks.

**III. Scheduling in Path Graph Models**

**A. Maximum Size Matching (MSM) Policies**

The sets of transmitters and receivers can be viewed as nodes in a bipartite (communication) graph, whose edges are the links between these transmitters and receivers, see Fig. 1. The link activation constraints are superimposed on the bipartite communication graph. With this structure in mind, and to conform to standard terminology from bipartite graphs [21], the following definition holds.

**Definition 1:** A policy $\pi$ is a Maximum Size Matching (MSM) policy if in every slot the policy schedules the maximum number of links with nonempty transmitter queues subject to the interference constraints.

For example, in a path graph network with $N = 7$ queues, if $Q(t) = [1, 2, 0, 0, 4, 3, 3]$, a policy that schedules queues 1, 5 and 7 or 2, 5 and 7 is MSM while a policy that schedules queues 1, 7 only, is not MSM. It might be expected that the policy must schedule as many queues as possible to maximise throughput and minimise delay. Indeed, it is shown in [15] that any policy defined on such path-graph networks can be improved into an MSM policy that will provide stochastically better delay. Interestingly, we show later that even non-MSM policies can be stabilising.

In this paper, we will refer to Queues 1 and $N$ in a path graph as the “outer” queues and the other $N - 2$ queues as the “inner” queues. The inner queues are, in a sense, more constrained for scheduling, since they cannot be served in any slot in which either adjacent queue is scheduled, while service to Queue 1 depends only on whether Queue 2 is being served, which makes it less constrained from the perspective of service.

Lem. 4.1, in [15], defines a class of policies that is more restrictive than MSM that can be described informally and succinctly as follows.

1) the policy should be MSM, and
2) the policy should prioritize inner queues over outer queues while breaking ties.

Specifically, the authors provide a sufficient but not necessary condition for an activation vector to serve the largest number of nonempty queues in a slot. Recall that $\zeta(t) = [\mathbb{I}(Q_1(t) > 0), \ldots, \mathbb{I}(Q_N(t) > 0)]$ and define $U(\zeta(t)) \subset V$ as the set of all activation vectors that serve the largest number of queues in slot $t$ when the occupancy vector is $\zeta(t)$. With this, one simply needs to ensure that in every slot, the policy chooses activation vectors only from $U(\zeta)$, to ensure that it is MSM. In fact, we will show that in several interference graphs, $\zeta(t)$ is sufficient not only for stability but also for delay optimality.

**Notation:** Classes of scheduling policies

- $\Pi^{(N)}$: the class of all policies.
- $\Gamma^{(N)}_{M}$: the class of all MSM policies.
- $\Pi^{(N)}_{M}$: the class of all policies that take only the occupancy vector $\zeta(t)$ as input and activate the largest number of non empty queues in every slot, i.e., MSM policies that...
require only the empty or nonempty status of the queues in the network. We will call the policies that use only $\zeta(t)$ Queue Non-emptiness Based (QNB) policies; see Sec. III-B.

- $\Pi^{(N)}$: the class of all MSM policies within $\Pi_M^{(N)}$ that additionally break ties in favour of inner queues (see condition 2 earlier in this subsection).

Note that $\Pi^{(N)} \supseteq \Pi_M^{(N)} \supseteq \Pi^{(3)}$. Going back to our 7-queue example, when $\zeta(t) = [1, 1, 0, 0, 1, 1, 1]$, policies that choose $S(t) = [1, 0, 0, 0, 1, 0, 1]$ can be in $\Pi^{(7)}$ but not in $\Pi^{(7)}$, while those that choose $S(t) = [0, 1, 0, 0, 1, 0, 1]$ can be in $\Pi^{(7)}$.

### B. Queue Nonemptiness-Based (QNB) Scheduling

In the previous subsection we defined “queue nonemptiness based” policies, i.e., those that require only the knowledge of the occupancy vector, $\zeta(t)$. Clearly, this contains much less information than the vector $Q(t)$ that MaxWeight requires. While $\zeta(t)$ needs only $N$ bits per slot for encoding, $Q(t)$ may require an arbitrarily large number of bits per queue per slot, depending on the buffer size and quantization used. So now, $\pi = \{\mu, \mu, \cdots\}$ with $\mu : \{0, 1\}^N \rightarrow V \subseteq \{0, 1\}^N$, the set of all activation vectors.

Although it is well-known that fully-connected interference graphs admit throughput-optimal, queue nonemptiness-based scheduling algorithms, (e.g., schedule any nonempty queue) it is not immediately clear how to stabilize other interference graphs with reduced state policies. Moreover, it is not clear if using a reduced state scheduler automatically ensures delay optimality, since even MaxWeight, which uses complete knowledge of $Q(t)$ in every slot, is only known to be asymptotically delay optimal in such networks [22].

We now provide a key sufficient condition for a scheduling policy, that guarantees throughput-optimality. This result will form the basis for constructing strongly stable policies that use only $\zeta(t), t \geq 0$, throughout the remainder of the paper.

**Lemma 2:** Consider the class of systems described in Sec. II-B, and define property $P$ as

$$D_i(t) + D_{i+1}(t) = 0 \iff Q_i(t) + Q_{i+1}(t) = 0,$$

for all $t \geq 0$, and for $1 \leq i \leq N - 1$. Any policy that satisfies property $P$ in every slot $t$, is throughput-optimal.

**Remark 3:** Note that condition $(P)$ depends only on the reduced state $\zeta(t)$. In words, $(P)$ reads: “for a pair of neighboring queues, there is no departure from either of these queues if and only if both the queues are empty.” One direction is clear: when both queues are empty there can be no departures. For example, with $N = 4$ and $\zeta(t) = (1, 1, 1, 1)$, $S(t) = (1, 0, 1, 0)$ satisfies condition $(P)$, but $S(t) = (1, 0, 0, 1)$ does not.

Also note that extensions of this property for the cluster-of-cliques system will be discussed and presented in Sec. VII, when we take up a detailed study of these conflict graphs.

**Proof Sketch.** The proof of Lem. 2 is based on a novel Lyapunov function $L(t) : \mathbb{R}^N \rightarrow \mathbb{R}^+$, defined as

$$L(Q(t)) := \sum_{i=1}^{N-1} (Q_i(t) + Q_{i+1}(t))^2. \quad (8)$$

Instead of showing negative drift per queue state, as is common in the analysis of several MaxWeight-style algorithms, we provide an averaging argument to show overall negative drift of the Lyapunov function, and appeal to the telescoping sum technique to prove strong stability. The proof is deferred to Sec. XI-D in the supplementary material.

### IV. Path Graph Conflict Model

**With $N = 3$: QNB Scheduling**

In this section, we first completely characterize $\Pi^{(3)}$ and the subclass $\Pi^{(3)}$, and explore stability and delay optimality for this system. This study will provide some insights into the nature of MSM policies in general and, more importantly, in this process, the policies we propose here will act as building blocks for policies for larger-$N$ systems. Before we embark on this analysis, we would like to make a few preliminary observations about $\Pi^{(3)}$. For the reader’s convenience a glossary of notation is provided in the supplementary material: XI-B.

Note that with 3 queues, in any given slot $t$, a policy can choose either $S(t) = [1, 0, 1]$ which serves Queues 1 and 3, or $[0, 1, 0]$ which serves Queue 2. So, a queue nonemptiness-based policy maps every state vector $\zeta(t)$, of which there are 8 alternatives, to one of these two activation vectors, giving us $2^8 = 256$ QNB policies in all. Suppose $| A |$ denotes the cardinality of set $A$. It is easily shown that upon imposing the MSM condition, this number reduces to 4, i.e., $| \Pi^{(3)} | = 4$ in our techreport [23, Sec. V].

Dependong on the mapping from $\zeta(t)$ to the activation vector, we denote the 4 MSM policies $\pi_{TD}^{(3)}, \pi_{BU}^{(3)}, \pi_{IQ}^{(3)}$, $\pi_{OQ}^{(3)}$. We will follow the scheme below in the remainder of the paper.

- The subscripts “TD” and “BU” stand for “Top-Down” and “Bottom-Up,” respectively and the reason for this nomenclature will become apparent shortly.
- A “~” in the superscript always represents a policy in $\Pi^{(N)}$, regardless of any subscripts. It indicates that these policies always break ties in favor of inner queues. For example, in Sec. IV-A, $\pi_{IQ}^{(3)} \in \Pi^{(3)}$.

The complete descriptions of all these policies are given in Table I. Let us consider each of these policies in turn. In what follows we will describe and analyze each of these policies in detail. Notice from the entries corresponding to the rows $\zeta = [011]$ and $\zeta = [110]$ that $\pi_{TD}^{(3)}$ and $\pi_{BU}^{(3)}$ are complementary policies, and so are $\pi_{IQ}^{(3)}$ and $\pi_{OQ}^{(3)}$. Specifically, each of these four policies induces the following priority order, which will become clear when we consider each of them individually later:

- $\pi_{TD}^{(3)}$ gives decreasing priority to Queues 1, 2 and 3 in that order. This policy clearly gives absolute priority to Queue 1, i.e., serves Queue 1 whenever it is nonempty. Hence, the subscript “TD,” since this policy, in a
sense, establishes a “Top-Down” priority (thinking of Queue 1 as the “top,” and Queue 3 as the “bottom”),

- \( \pi_{BU}^{(3)} \) gives increasing priority to Queues 1, 2, and 3 in that order,
- \( \pi_{IQ}^{(3)} \) gives maximum priority to Queue 2 the inner queue (check rows in Table. I corresponding to \( \zeta = [011] \) and \( \zeta = [110] \)), while not compromising the MSM property. This is, of course consistent with the fact that it lies in the \( \Pi^{(3)} \) class where ties are always broken in favor of inner queues, and
- \( \pi_{OQ}^{(3)} \) prioritizes the two outer queues.

To begin with, we show that \( \pi_{TD}^{(3)} \) and \( \pi_{BU}^{(3)} \) are T.O. Both these policies will later be used as building blocks to construct T.O. policies for larger systems and are therefore very important to our study.

**Theorem 4**: \( \pi_{TD}^{(3)} \) and \( \pi_{BU}^{(3)} \) are both throughput-optimal.

**Proof Sketch**: The proof of Thm. 4 uses the fact that under \( \pi_{TD}^{(3)} \), Queues 1 and 2 form a priority queueing system and are stable. We then show that Queue 3 is served “sufficiently often” to ensure stability. \( \pi_{BU}^{(3)} \) simply swaps the priorities of Queues 1 and 3 and its proof proceeds mutatis mutandis. The complete proof is available in Sec. XI-E.

The rest of this section is organized as follows. In Sec. IV-A we study the stability and delay performance of \( \pi_{IQ}^{(3)} \). We then analyze the stability of \( \pi_{OQ}^{(3)} \) in Sec. IV-B and show that it is, in fact, unstable. We then study a non-MSM policy \( \tilde{\pi}_{IQ}^{(3)} \), and show it to be T.O. This policy is used as a building block in policies for larger path graphs systems later. Finally, in Sec. IV-D we study a version of the “Flow-in-the-Middle” problem, an interesting phenomenon commonly observed in contention access networks, show that path graph networks also exhibit this behavior and analyze its consequences.

**A. Analysis of \( \tilde{\pi}_{IQ}^{(3)} \)**

This policy can be restated as follows.

**At time** \( t \):

1. If \( Q_1(t) > 0 \) and \( Q_3(t) > 0 \), then \( S(t) = [1, 0, 1] \).
2. Else, if \( Q_2(t) > 0 \), then \( S(t) = [0, 1, 0] \).
3. Else \( S(t) = [1, 0, 1] \).

We begin analyzing the policy by proving that it is Throughput-Optimal.

**Theorem 5**: \( \tilde{\pi}_{IQ}^{(3)} \) is throughput-optimal.

**Proof Sketch**: The proof of this result involves showing that \( \tilde{\pi}_{IQ}^{(3)} \) satisfies (P) in Lem. 2 and is therefore throughput-optimal. The proof is available in Sec. XI-F.

We next turn to the delay performance of the policy \( \tilde{\pi}_{IQ}^{(3)} \). Thm. 4.2 in [15] defines a projection operator \( L : \Pi^{(N)} \rightarrow \Gamma^{(N)} \) that takes any policy \( \pi \in \Pi^{(N)} \) and produces an MSM policy, \( L(\pi) \). It is then shown that the sum queue length with this MSM policy \( L(\pi) \) is stochastically smaller than with \( \pi \). Specifically, if \( Q^\pi(t) \) denotes the backlog induced by some policy \( \pi \), then Thm. 4.2 in [15] shows that when the systems upon which \( \pi \) and \( L(\pi) \) act are started out in the same initial state and the arrivals have the same statistics, then

\[
\sum_{i=1}^{N} Q_i^L(t) \leq \sum_{i=1}^{N} Q_i^\pi(t), \quad \forall t \geq 0,
\]

where “st” denotes stochastic ordering. Notice that in the above stochastic ordering relation is required to hold for any arrival rate vector in the system’s capacity region. Extending this gives rise to the concept of a Uniformly Delay Optimal Policy (in the literature, this is also referred to as Sample-Path Optimality [24], [25]):

**Definition 6**: For a path graph interference network with \( N \) queues, a policy \( \pi^* \in \Pi^{(N)} \) is said to be Uniformly Delay Optimal if, given any policy \( \pi \in \Pi^{(N)} \), when the systems upon which \( \pi \) and \( \pi^* \) act are started out in the same initial state and with the same arrivals statistics and for every arrival rate \( \lambda \in \Lambda_N \),

\[
\sum_{i=1}^{N} Q_i^{\pi^*}(t) \leq \sum_{i=1}^{N} Q_i^\pi(t), \quad \forall t \geq 0.
\]

Remark 8: Another important consequence of the operator \( L \) satisfying (9) is that whenever a given policy \( \pi \) is throughput-optimal, so is \( L(\pi) \). Please refer [23, Sec. V] for details. Directly analyzing the stability and delay properties of the policies we propose in the sequel is very difficult. We therefore develop indirect methods to analyze them by first analyzing nonMSM policies whose behavior can be understood easily, but that do not show desirable delay properties and study the proposed policies as modifications (such as projection) of these simpler policies, with the modifications giving rise to better delay performance.

**B. Analysis of \( \pi_{OQ}^{(3)} \)**

This policy prioritizes the outer queues and can be restated as follows.

**At time** \( t \):

1. If \( \text{either } Q_1(t) > 0 \text{ or } Q_3(t) > 0 \), then \( S(t) = [1, 0, 1] \).
2. Else \( S(t) = [0, 1, 0] \).
It turns out, analogous to the observation by McKeown et al [26] that this MSM policy is, in fact, not throughput-optimal.

**Proposition 9 (MSM But Not Throughput-Optimal):** $\pi_{OQ}(3)$ is not throughput-optimal.

The proof of this result involves constructing an arrival rate vector for which the offered service rate to one of the queues is strictly smaller than the arrival rate. It is available in Sec. XI-H of the Appendix. Once again, this proof technique is important and we will repeatedly use it in the sequel.

This completes the characterization of $\Pi_{M}^{(3)}$.

**C. Policies Outside $\Pi_{M}^{(3)}$**

We now propose and analyze a policy that we denote $\pi_{IQ}(3)$, and show the rather surprising result that it is throughput-optimal despite not being MSM. This stability comes from the fact the policy prioritizes the inner queue. However, since it is not MSM, its delay performance is not very good (see the simulation results in Sec. IX). This policy will become important shortly as a fundamental building block while constructing policies for larger systems using a novel Policy Splicing technique.

At time $t$

1) If $Q_{2}(t) > 0$, then $S(t) = [0, 1, 0]$.
2) Else $S(t) = [1, 0, 1]$.

Since $\zeta(t) = [1, 1, 1] \rightarrow [0, 1, 0]$, this policy is not MSM. However, we have

**Proposition 10 (A Non-MSM But Throughput-Optimal Policy):** $\pi_{IQ}(3)$ is throughput-optimal.

**PROOF.** The key tool behind the proof of this result is the throughput-optimality Lemma 2. It is easily checked that $\pi_{IQ}(3)$ satisfies ($P$) in every slot and thus, by Lemma 2, is throughput-optimal. □

**D. A Randomized Policy: The Flow-in-the-Middle Problem**

The “Flow-in-the-Middle” problem, or FIM for short, is a practical problem faced by all networks that employ CSMA at the MAC layer. It refers to the situation in which, the flow (or link) within the interference range of two adjacent flows, i.e., in the middle (Queue 2, in our model), can remain starved of service for long periods of time in the presence of uncoordinated transmissions. This problem has been studied in detail both analytically and experimentally in asynchronous continuous-time systems in the literature [27]-[32]. In this section, we aim to model such a scenario, albeit in slotted time, and understand whether such a phenomenon can occur in the network under study, which naturally leads to the central link (or flow) being starved for extended periods of time. Recall the occupancy vector is defined as $\zeta(t) := [\zeta_{1}(t), \zeta_{2}(t)]$, where $\zeta_{1}(t)$ is the occupancy of Queue 1.

Consider the policy $\rho_{r}^{(3)}$ indexed by a randomization parameter $\gamma \in [0, 1]$ as follows.

At time $t$:

- If $\zeta(t) = [1, 1, 1]$ or $[0, 0, 1]$, then $S(t) = [1, 0, 1]$.
- Else, if $\zeta(t) = [1, 1, 0]$ or $[0, 1, 1]$, then:
  1) $S(t) = [1, 0, 1]$ w.p. $1 - \gamma$ and
  2) $S(t) = [0, 1, 0]$ w.p. $\gamma$.
- Else, $S(t) = \zeta(t)$.

Clearly, this policy is a randomization between the two 3-queue MSM policies $\pi_{IQ}^{(3)}$ and $\pi_{OQ}^{(3)}$. A comparison of the definitions of $\rho_{r}^{(3)}$, $\pi_{IQ}^{(3)}$ and $\pi_{OQ}^{(3)}$ clearly shows that $\rho_{r}^{(3)}$ essentially chooses $\pi_{IQ}^{(3)}$ w.p. $\gamma$ and $\pi_{OQ}^{(3)}$ w.p. $1 - \gamma$. By this we mean that in every time slot, $\pi_{IQ}^{(3)}$ and $\pi_{OQ}^{(3)}$ choose their actions and $\rho_{r}^{(3)}$ then selects the former w.p. $\gamma$ and the latter w.p. $1 - \gamma$. In [23, Sec. VII] we analyze this policy and establish two important results: (i) $\rho_{r}^{(3)}$ is unstable for $\gamma \in [0, 0.5]$. We also conjecture that $\rho_{r}^{(3)}$ is unstable for all $\gamma \in [0, 1]$ but are unable to prove this as of now. Simulation results (Sec. IX) seem to indicate this as well, and (ii) Consider the set of arrival rates

$$\Lambda_{\gamma}^{(3)} := \{ \lambda \in \mathbb{R}_{+}^{2} | \lambda_{1} + \lambda_{2} < \gamma, \lambda_{2} + \lambda_{3} < \gamma \}.$$  \hspace{1cm} (12)

For every $\gamma \in (0, 1]$, $\rho_{r}^{(3)}$ stabilizes all rate vectors in $\Lambda_{\gamma}^{(3)}$. It follows that the stability region of $\rho_{r}^{(3)}$ is $\Lambda_{3} = \gamma \uparrow 1$.

**V. PATH-GRAPH MODELS WITH $N > 3$: POLICY SPICING FOR THROUGHPUT OPTIMAL QNB SCHEDULING**

The previous section introduced scheduling policies that rely only on the empty-nonempty status of queues and examined the behavior of such policies on a small network. We will use the knowledge gained therein to now propose such policies for larger systems while still confining ourselves to path-graph interference networks. We introduce a “policy splicing” technique to construct MSM policies for large systems by splicing together MSM policies for smaller systems.

**A. Top-Down and Bottom-up Scheduling on N Queues**

Recall the “Top-Down” and “Bottom-Up” policies, $\pi_{TD}(N)$ and $\pi_{BU}(N)$, discussed in Sec. IV. For a general path-graph network with $N$ queues, the “Top-Down” policy, $\pi_{TD}(N)$, which maps an occupancy vector $\zeta(t)$ to an activation vector $s(t)$, is defined as follows. Before defining the policy, we assume the presence of two virtual queues, Queue 0 and Queue $N + 1$, with $Q_{0}(t) = Q_{N+1}(t) = s_{0}(t) = s_{N+1}(t) = 0$, $\forall t \geq 0$. This is just to facilitate compact writing of the policy. These virtual queues do not play any actual role in the system. Recall that if $Q(t) = [Q_{1}(t), \cdots, Q_{N}(t)]$ is the queue length vector at time $t$, then the occupancy vector at time $t$ is defined by $\zeta(t) = [\mathbb{I}_{Q(t)>0}, \cdots, \mathbb{I}_{Q_{N}(t)>0}]$. The $\pi_{TD}(N)$ policy: At time $t$.

- For $j=1:N$
  1) If $\zeta_{j}(t) = 1$ and $s_{j-1}(t) = 0$, then $s_{j}(t) = 1$.
  2) Else if $\zeta_{j}(t) = 1$ and $s_{j-1}(t) = 1$, then $s_{j}(t) = 0$.
  3) Else $s_{j}(t) = 0$.

It is easy to see that this produces $\pi_{TD}(N)$ for $N = 3$, and $\pi_{BU}(N)$ is defined similarly. The following important property follows from the definition.

**Proposition 11:** $\pi_{TD}(N)$ and $\pi_{BU}(N)$ are MSM for all $N \in \mathbb{N}$. **PROOF.** Refer Sec. XI-I in the supplementary material. □

Before we venture into proving the throughput-optimality of $\pi_{TD}(N)$ and $\pi_{BU}(N)$, we use these two policies to describe the policy splicing process (see Fig. 3).

**B. Splicing TD and BU Policies**

Consider a system of $2N-1$ queues, $N \geq 1$. In Algorithm 1, we splice the TD and BU policies and construct a
scheduling policy\(^6\) \(\pi_{SP}^{(2N-1)}\) on this system. Note, we assume the presence of the two \textit{virtual queues} Queue 0 and Queue \(2N\) here as well. Before proceeding to analyze this policy, we first need to make sure it really is a well-defined policy, i.e., it provides a valid activation vector for each of the \(2^{2N-1}\) possible occupancy vectors.

\textbf{Lemma 12:} \(\pi_{SP}^{(2N-1)}\), as defined above, is well-defined.

\textbf{Proof:} See Sec. XI-K in the supplementary material.\(\blacksquare\)

A quick comparison with the definitions of \(\pi_{TD}^{(N)}\) and \(\pi_{BU}^{(N)}\) shows that \(\pi_{SP}^{(N)}\) \((2N-1)\) induces the former two policies on the subsets \(\{N, N+1, \cdots, 2N-1\}\) and \(\{1, 2, \cdots, N\}\) respectively. The following result follows from the definition of the splicing process. Recall from Sec. II-B that the capacity region of a path-graph interference network consisting of \(N\) queues is defined by

\[\Lambda_N := \left\{ \lambda \in \mathbb{R}^N_+ | \lambda_i + \lambda_{i+1} < 1, \ 1 \leq i \leq N-1 \right\}, \quad N \in \mathbb{N},\]

\textit{Theorem 13:} For every \(N \in \mathbb{N}\), such that \(\pi_{TD}^{(N)}\) and \(\pi_{BU}^{(N)}\) are throughput-optimal over \(\Lambda_N\), \(\pi_{SP}^{(2N-1)}\) is throughput-optimal over \(\Lambda_{2N-1}\).

\textbf{Proof:} See Sec. XI-J in the supplementary material.\(\blacksquare\)

\textbf{Remark 14:} We will discuss the stability of our TD and BU policies in Sec. V-D. It is important to note that although \(\pi_{TD}^{(N)}\) and \(\pi_{BU}^{(N)}\) are MSM, \(\pi_{SP}^{(2N-1)}\) is not. For example, consider an occupancy vector such that \(\zeta_{N-1} = 1\) and \(\zeta_j = 0, \forall j \in \{1, \cdots, N-2\} \cup \{N+2, \cdots, 2N-1\}\), i.e., the central queue and both adjacent queues are nonempty and all other queues are empty. Any MSM policy would produce the activation vector with \(S_{N-1}(t) = 1\) and \(S_j(t) = 0, \forall j \notin \{N-1, N+1\}\), whereas \(\pi_{SP}^{(2N-1)}\) produces the activation vector with \(S_N(t) = 1\) and \(S_j(t) = 0, \forall j \neq N\) thereby scheduling one less queue for transmission.

\textbf{C. Mapping \(\pi_{SP}^{(N)}\) to an MSM Policy}

To reduce delay, one needs to extract an MSM policy from this spliced policy. Informally, we project the policy onto the space \(\Pi_{M}^{(N)}\) of MSM policies using the projection operator \(L(\cdot)\), described in Sec. IV-A and defined in [23, Sec. V.C]. Thereafter, we use some observations based on Condition 2 in Sec. III-A to improve the delay performance of this projected MSM policy to finally obtain a policy in \(\Pi^{(N)}\). Figures 3 and 4 give a pictorial description of the entire process.

\(6\)The subscript “SP” refers to the fact that this is a Spliced Policy.
$2N - k$ queues and append $k$ virtual queues that start out empty and receive no arrivals in any time slot and run $\pi_{SP}^{(2N-1)}$ on them. So, the focus of the remainder of this section is proving the throughput-optimality of the Top-Down and Bottom-Up priority policies.

To summarise, the general policy splicing process involves the following steps (see Figures 3 and 4).

**General Splicing Procedure:**

1. Proposing Top-Down (TD) and Bottom-Up (BU) policies for the $N$-queue system,
2. Splicing them together to produce a non-MSM policy for the $(2N-1)$-queue system,
3. Projecting the spliced policy to get an MSM policy,
4. Modifying this policy to break ties in favor of inner queues to get a policy with better delay performance than its MSM predecessor. For example, with $N = 5$ queues and $\zeta(t) = [1, 1, 1, 1, 0]$, policies that choose $S(t) = [1, 0, 1, 0, 0]$ and $S(t) = [0, 1, 0, 1, 0]$ are both MSM, but only the latter breaks ties in favor of inner queues.\(^7\)

We have already proposed and analyzed two priority policies for 3-queue path networks that we named $\pi_{TD}^{(3)}$ and $\pi_{BU}^{(3)}$. As a quick illustration of the above procedure, in (23, Sec. VI.A) we explicitly show the entire procedure of how low delay policies for $2 \times 3 - 1 = 5$-queue systems can be constructed from these two 3-queue policies. Due to space constraints, we are unable to present the details in this paper.

**D. Policies for $N = 7, 8$ and $9$ Through Policy Splicing**

The Top-Down and Bottom-Up policies $\pi_{TD}^{(4)}$ and $\pi_{BU}^{(4)}$ will, as already discussed, be used to develop stabilizing policies for systems with $N = 2 \times 4 - 1 = 7$ queues. An equivalent way to define the Top-Down policy $\pi_{TD}^{(4)}$ is as below.

**At time $t$:**

1. If $Q_1(t) > 0$,
   a) If $Q_2(t) > 0, s(t) = (1, 0, 1, 0)$.
   b) Else, $s(t) = (1, 0, 0, 1)$.
2. Else, $Q_2(t) > 0, S(t) = (0, 1, 0, 1)$.
3. Else,
   a) If $Q_3(t) > 0, S(t) = (1, 0, 1, 0)$.
   b) Else, $S(t) = (1, 0, 0, 1)$.

**Proposition 16:** $\pi_{TD}^{(4)}$ is throughput optimal.

**Proof:** Notice that as far as the subsystem $[Q_1(t), Q_2(t), Q_3(t)]$ is concerned, this policy reduces to $\pi_{TD}^{(3)}$. That is, $\pi_{TD}^{(4)}$ restricted to the first three queues is $\pi_{TD}^{(3)}$. So, that subsystem is strongly stable.

The remainder of the proof of the proposition can be found in Sec. XI-L in the supplementary material.

It is easy to see that the Bottom-Up policy, $\pi_{BU}^{(4)}$, defined in a symmetric manner, giving highest priority to Queue 4 and lowest to Queue 1, is also T.O. We then define $\pi_{SP}^{(7)}$ as described in Sec. 1. Clearly, since $\pi_{SP}^{(7)}$ restricted to Queues 1, 2, 3, 4 is just $\pi_{BU}^{(4)}$ and restricted to Queues 4, 5, 6, 7 is $\pi_{TD}^{(4)}$, using the fact that both $\pi_{TD}^{(4)}$ and $\pi_{BU}^{(4)}$ are throughput-optimal and Thm. 13, we conclude that $\pi_{SP}^{(7)}$ is throughput-optimal as well. However, since $\pi_{SP}^{(7)}$ is not MSM, some modifications are required to improve delay performance. This simply requires executing Steps 3 and 4 in the general splicing procedure V-C.

In a similar manner we will now show that the top-down and bottom-up policies for the 5 queue system ($\pi_{TD}^{(5)}$ and $\pi_{BU}^{(5)}$) are both throughput-optimal, which will immediately yield a stabilizing policy ($\pi_{SP}^{(9)}$) for the 9-queue system.

**Proposition 17:** $\pi_{TD}^{(5)}$ is throughput-optimal.

**Proof:** This analysis closely follows our analysis of $\pi_{TD}^{(5)}$. With $\pi_{TD}^{(5)}$, we only need to prove that Queue 5 receives “enough” service, since this policy restricted to the first 4 queues is just $\pi_{TD}^{(4)}$ which, as we have just shown, is throughput-optimal. The remainder of the proof of the proposition can be found in Sec. XI-N.

The analysis of the bottom-up policy ($\pi_{BU}^{(5)}$) proceeds in a symmetric fashion. This means that the 9-queue policy $\pi_{SP}^{(9)}$, that induces $\pi_{TD}^{(5)}$ on queues 1, 2, 3, 4, and 5 and $\pi_{TD}^{(5)}$ on queues 5, 6, 7, 8 and 9, is throughput-optimal.

To summarize, in this section, we first proposed a general procedure to generate MSM Top-Down and Bottom-Up priority policies for a system with any $N \in \mathbb{N}$ queues. We then showed how these policies can be combined to construct policies for larger systems and provided a sufficient condition for such a spliced policy to be throughput-optimal. We then spliced the TD and BU policies for $N = 4$ and $N = 5$ queue systems and constructed stable QNB-MSM policies for systems with up to $N = 9$ queues. We now move on to analyzing the queueing delay performance of these policies.

**VI. PATH-GRAPH CONFLICT GRAPHS WITH $N > 3$: DELAY WITH QNB POLICIES**

We now turn our attention to the vital aspect of delay. We have already proved, in Thm. 7, that for the system with $N = 3$ queues, there exists a (unique) uniformly delay-optimal policy, that we named $\pi_{IQ}^{(3)}$. The natural question to ask in this context is if one can find a delay optimal queue nonemptiness-based policy for larger systems as well. In this section, we will answer this question in the negative.

**Theorem 18:** For all $N \geq 4$, there does not exist any policy in $\Pi_{M}^{(N)}$ that is uniformly delay optimal over all of $\Lambda_4$.

The proof of this theorem has been omitted due to space constraints, but we provide a short sketch below. The main idea is to show that $\Pi_{M}^{(N)}$ does not contain any queue length agnostic policy that is uniformly delay optimal over the entire throughput capacity region $\Lambda_4$ (the complete procedure is detailed in [23, Sec. IX]). We begin by proving, in Prop. 19 in [23], that the class of inner-queue prioritising MSM policies, $\Pi^{(4)}$, does not contain any uniformly delay optimal policy. $\Pi^{(4)}$ contains four policies which we show to be throughput optimal by first showing them to be the result of a splicing procedure involving one of the two priority policies $\pi_{TD}^{(3)}$ or $\pi_{BU}^{(3)}$, and $\pi_{IQ}^{(3)}$ (defined in Sec. IV-C). We then prove a theorem similar to Thm. 13 to establish throughput optimality. However, we then show that none of these policies performs uniformly better (or at least as good as) the others over all of $\Lambda_4$.

Next, in Prop. 20 therein we show that policies in $\Pi_{M}^{(4)}$ show better delay performance than those in $\Pi_{M}^{(N)}$. We already know from (9) that the delay of any policy in $\Pi_{M}^{(N)}$ can be

\(^7\)For further details, see Lem. 19 in the Appendix

\(^8\)Strictly speaking, from the definition of $\pi_{TD}^{(N)}$ above, $s_4(t)$ should be set to 1 iff $Q_4(t) = 1$. But setting $s_4(t) = 1$ when $Q_4(t) = 0$ doesn’t violate interference constraints since it means that Queue 4 is empty.
improved by projecting it onto $\Pi_1^{(N)}$. Thus, Prop. 20 in [23], along with (9), shows that delay optimal policies, when they exist, must necessarily lie in $\Pi_1^{(N)}$. This observation, along with the nonexistence of delay optimal policies in $\Pi_2^{(4)}$, are used to prove the claim in Thm. 18.

VII. CLUSTER-OF-CLIQUES INTERFERENCE NETWORKS: THROUGHPUT OPTIMAL SCHEDULING

We will now show that some of the scheduling policies developed for path-interference graph networks extend in a natural manner to policies for the SoC and the LAoC networks.

**Notation:** We denote policies designed for Star-of-Cliques networks by “$\theta$” and include an “$^\theta$” in the superscript to emphasize this. On the other hand, “$\varphi$” with and “$^\varphi$” in the superscript specifies an LAoC network policy. We will begin with centralized scheduling in SoC networks.

A. Scheduling in Star-of-Cliques Networks

Consider the following policy that we denote $\varphi_{IC}^{(3)}$, which is motivated by the 3-node path graph policy $\pi_{IC}^{(3)}$ which we discussed in Sec. IV-A. Recall that we defined $\mathcal{N}$ to be the total number of queues in the network. In keeping with the objective of developing queue nonemptiness-based policies, in every slot, $\varphi_{IC}^{(3)}$ maps the occupancy vector $\zeta(t) \in \{0,1\}^N$ to an activation vector $s(t) \in \{0,1\}^N$. We define $\varphi_{IC}^{(3)}$ as follows.

At each time $t$:
1. If $\sum_{m=2}^{N} \sum_{j \in C_m} \zeta_j(t) > 0$ serve any nonempty queue in every clique $\{C_m, m \geq 2\}$ having nonempty queues,
2. Else, if $\sum_{j \in C_1} \zeta_1(t) > 0$, serve any nonempty queue in $C_1$.
3. Else, serve one nonempty queue (if it exists) in each of $\{C_m, m \geq 2\}$.

In words, the above policy states that at time $t$,
- if every peripheral clique has at least one non empty queue, then serve one non empty queue in each of these cliques.
- else, if the inner clique has a non empty queue, serve one non empty queue in that clique.
- else, serve one non empty queue in every peripheral clique that has a non empty queue.

**Proposition 19:** $\varphi_{IC}^{(3)}$ is throughput-optimal.

**Proof:** The main idea behind the proof of this proposition is to prove a more general version of Property $\mathcal{P}$ (which we defined in Lem. 2 for path-graph networks) and use and use the total per-clique backlog as inputs to a new Lyapunov function to prove strong stability. See XI-O for details.

Towards the end of our discussion on queue nonemptiness-based scheduling for path-graph networks with $N = 3$ queues (see Sec. IV-C), we defined a non-MSM policy $\pi_{IC}^{(3)}$. Extending this to the SoC network model gives us a second queue nonemptiness-based policy $\varphi_{IC}^{(3)}$, which we define as follows.

At time $t$:
1. If $\sum_{j \in C_1} \zeta_1(t) > 0$, serve any nonempty queue in $C_1$.
2. Else, serve one nonempty queue (if it exists) in each of $\{C_m, m \geq 2\}$.

**Proposition 20:** $\varphi_{IC}^{(3)}$ is throughput-optimal.

**Proof:** Once again, the proof of this result rests on proving the new version of $\mathcal{P}$ for this policy, followed by Lyapunov analysis. The proof is available in Sec. XI-T.

Remark 21: We end this section with some remarks about implementation and delay performance. From the point of view of implementation, the latter, $\varphi_{IC}^{(3)}$ is actually easier to implement than $\varphi_{IC}^{(1)}$. We discuss this in detail in Sec. VIII which is completely dedicated to implementation issues. However, $\varphi_{IC}^{(3)}$ has its own advantages. With respect to the packet delay, recall that we had used a stochastic ordering argument to prove the delay optimality of Policy $\varphi_{IC}^{(1)}$ and later used a similar technique to show the absence of uniformly delay-optimal queue nonemptiness-based policies for path-graph networks. Along similar lines, we compare the delays induced by $\varphi_{IC}^{(3)}$ and $\varphi_{IC}^{(1)}$ below.

1) **Comparison of Delay With $\varphi_{IC}^{(1)}$ and $\varphi_{IC}^{(3)}$:**

**Proposition 22 (Stochastic Dominance of System Backlog):** Let the system backlog at time $t \geq 0$ with $\varphi_{IC}^{(1)}$ and $\varphi_{IC}^{(3)}$ be denoted by $Q_{IC}^{(1)}(t)$, and $Q_{IC}^{(3)}(t)$ respectively. Then, with $Q_{IC}^{(3)}(0) \succeq Q_{IC}^{(1)}(0)$, and arrivals to corresponding queues having the same statistics in both systems,

$$\sum_{m=1}^{N} \sum_{j \in C_m} Q_{IC}^{(3)}(t) \leq \sum_{m=1}^{N} \sum_{j \in C_m} Q_{IC}^{(1)}(t), \quad \forall t \geq 0. \quad (14)$$

**Proof Sketch:** This proof proceeds along the same lines as the proof of delay optimality of Policy $\varphi_{IC}^{(1)}$ that we presented in Sec. XI-G. It can be found in Sec. XI-R in the supplementary material.

We now begin our study of scheduling in LAoC networks, and return to policies for Star-of-Cliques networks once again when we shift our focus to decentralized implementation.

B. Scheduling in Linear-Arrays-of-Cliques

The technique we use to propose scheduling policies for LAoC networks is the policy splicing technique we developed in Sec. V. The proofs therein cannot be directly used to assess the stability of policies designed for LAoC networks since the proofs are designed for Bernoulli arrival processes to queues and require some more work to be extended to handle scheduling over cliques: however, one could argue that a clique can, in essence, be treated as a queue with an arrival process that is the sum of the processes to Queues $Q_{11,1}, Q_{11,2}$ and $Q_{1,3}$ therein. The resulting arrival process to the queue would then be a batch arrival process with arbitrary batch size (there can be any number of queues in a clique), and simple extensions of the proofs supplied hitherto can be shown to suffice.

As before, we begin with Top-Down and Bottom-Up policies for the 3-clique L AoC and splice them to construct policies for the L AoC’s with 4 and 5 cliques. Note, once again, that we place no restrictions on the number of queues within any clique.

1) **Scheduling Policies for Systems With $N = 3$ Cliques:**

The policy $\varphi_{TD}^{(3)}$ is described as follows.

At time $t$:
- If $\sum_{j=1}^{N_1} \zeta_{1,j}(t) > 0$ schedule any non-empty queue in $C_1$.
- If $\sum_{j=1}^{N_3} \zeta_{3,j}(t) > 0$ schedule any non-empty queue in $C_3$.
- Else, if $\sum_{j=1}^{N_2} \zeta_{2,j}(t) > 0$ schedule any non-empty queue in $C_2$.
• Else schedule any non-empty queue in $C_3$.

In other words, if in slot $t$ there is a non-empty queue in $C_1$, then $\theta_{TD}^{(3L)}$ serves one non-empty queue in $C_1$ and $C_3$.

• if $C_1$ is empty but $C_2$ has a non-empty queue in it, then $\theta_{TD}^{(3L)}$ serves that queue.

• if $C_1$ and $C_2$ are both empty, then $\theta_{TD}^{(3L)}$ serves any non-empty queue in $C_3$.

Proposition 23: $\theta_{TD}^{(3L)}$ is throughput-optimal.

Proof. This proof uses the ideas involved in proving the throughput-optimality of $\pi_{TD}$ and simply extends them to incorporate batch arrivals. The proof is available in Sec. XI-S.

A similar proof shows that $\theta_{BU}^{(3L)}$ is also throughput-optimal.

2) Scheduling Policies for Systems With $N = 4$ and 5 Cliques: Now, by splicing together $\theta_{TD}^{(3L)}$ and $\theta_{BU}^{(3L)}$, one can construct stable policies for the system with 5 cliques and hence, systems with 4 cliques. The spliced policy, $\theta_{SP}^{(5L)}$ is defined as

At time $t$:

1) If $\sum_{j=1}^{N_3} \zeta_{3,j}(t) > 0$ then schedule a nonempty queue in $C_3$.

   a) If $\sum_{j=1}^{N_3} \zeta_{1,j}(t) + \sum_{j=1}^{N_3} \zeta_{2,j}(t) > 0$ then schedule any nonempty queue each in $C_1$ and $C_5$.

2) Else if $\sum_{j=1}^{N_3} \zeta_{2,j}(t) \times \sum_{j=1}^{N_3} \zeta_{1,j}(t) > 0$ then schedule a nonempty queue each in $C_2$ and $C_4$.

3) Else if $\sum_{j=1}^{N_3} \zeta_{1,j}(t) \times \sum_{j=1}^{N_3} \zeta_{2,j}(t) > 0$ then schedule a nonempty queue each in $C_1$ and $C_5$.

4) Else if $\sum_{j=1}^{N_3} \zeta_{1,j}(t) \times \sum_{j=1}^{N_3} \zeta_{2,j}(t) > 0$ then schedule a nonempty queue each in $C_1$ and $C_4$.

5) Else schedule any nonempty queue each in $C_1$ and $C_5$.

Proposition 24: The policy $\theta_{SP}^{(5L)}$ is throughput-optimal.

Proof. See Sec. XI-P.

To summarize, in this section, we studied scheduling in the Star-of-Cliques and Linear-Array-of-Cliques models that could occur in IoT-type sensor network applications. Having characterized the capacity region of such networks, we proposed and analyzed multiple scheduling policies. However, as mentioned before, these policies depend on being able to find a nonempty queue in every slot in which the system is not empty. While disseminating occupancy information across the network is certainly not as expensive as sharing queue length information (required by, say, the MaxWeight family of scheduling algorithms), it raises the question of the existence of policies can work with even less information. The following section describes some preliminary attempts at achieving this goal.

VIII. SOME REMARKS ON DECENTRALIZED IMPLEMENTATION

In this section, we discuss several ways in which the policies developed and analyzed hitherto can be made amenable to decentralized implementation. QNB policies, by definition, only require the empty-non empty statuses of queues. While this vector, $\zeta(t)$ by itself can be disseminated across the network using $N$ bits\(^9\) in many cases this status can be inferred using a simple hypothesis test and requires no information exchange between queues. To accomplish this decentralized state inference, we take the help of what are known as miniblots \([16], [33]\) which we describe in detail, below. Note that while we focus on SoC networks in this section, we also address decentralized scheduling in LAoC networks in \([23, Sec. XI.D]\).

Transmission Sensing: We assume that all nodes transmit at the same fixed power, and that the maximum internode distance is such that every node in clique $C_j$, $j \geq 2$ can sense the power from a transmitting node in (the central) clique $C_1$ and vice versa, as dictated by the interference constraints.\(^{10}\) Suppose a node has been scheduled to transmit in a slot. Then, whether or not the node actually transmits can be determined by the other nodes by averaging the received power over a small time interval (akin to the “Clear Channel Assessment” or CCA mechanism \([34]\)). For reliable assessment, the interval will need to be of a certain length, and the distance between the nodes will need to be limited. As before, we refer to this activity-sensing interval a miniblot (see \([16], [33]\) and Fig. 5). Decentralized methods of implementing both $\phi_{IC}^{(S)}$ and $\phi_{IC}^{(F)}$ follow from the miniblot structure. Define the occupancy of Queue $k$ in Clique $j$ by $i_{j,k}(t) := I(I_{j,k}(t) > 0)$, and let $I_j(t) := \sum_{k \in C_j} i_{j,k}(t)$. If Clique $j$ has any nonempty nodes at the beginning of time slot $t$, we will discuss implementing $\phi_{IC}^{(S)}$ in detail, and do the same for $\phi_{IC}^{(F)}$ in \([23, Sec. XI.C]\).

A. Decentralized Implementation of $\phi_{IC}^{(S)}$

At time $t$,

1) If $I_j(t) > 0$, then one nonempty node from clique $C_1$ is allowed to transmit (see Fig. 5). Nodes in the other cliques sense this transmission in the first miniblot and refrain from transmitting during that slot.

2) If no power is sensed in the first miniblot, it means $I_j(t) = 0$, and each of the other cliques choose one nonempty queue (if any) for transmission during that slot.

This, of course, assumes that one is somehow able to identify a nonempty queue, if one exists, in each clique. So this implementation is, by itself, centralized within a clique and decentralized across cliques. We now propose methods to determine which (nonempty) queue within a clique actually gets to transmit in either of the two steps above.

One method is for the nodes in a clique to periodically share occupancy information which could be accomplished by having a sink node in every clique. The sink node of each

\(^9\)This could potentially even be reduced to $\log_2(N)$ bits using network coding techniques – this is an important direction for future work.

\(^{10}\)Obviously, for all $2 \leq j, k \leq N$, nodes in $C_j$ cannot sense transmissions in $C_k$ and vice versa.
clique periodically aggregates occupancy information from its nodes and uses it to schedule nonempty queues in some order. We discuss this towards the end of this section, and in Sec. VIII-B we propose and analyze a version of $\phi_{IC}^{(S)}$ that requires no explicit information exchange between queues.

**B. $\phi_{IC}^{(S)}$ Without Occupancy Information: Towards Fully Decentralized Policies**

First consider a clique, say $C_1$, in isolation. This is, by itself, a fully connected interference graph. Suppose the nodes in $C_1$ could determine the backlog of a node in $C_1$ each time it transmitted a packet. Then, at the beginning of slot $t$, the information common to all nodes in $C_1$ would consist of the number of slots $V_i(t)$ since node $i$ last transmitted and its backlog $Q_i(t - V_i(t))$ at that instant. With this partial information structure and equal arrival packet rates to the queues, we have already shown in [16], that exhaustively serving a nonempty queue minimizes mean delay. With exhaustive service, $Q_i(t - V_i(t))$ is always 0, which obviates the need to transmit queue lengths. When the queue under service, called the **incumbent** in the sequel, becomes empty we have already shown that scheduling node arg max$_{j \in C_1}$ $V_j(t)$ is throughput-optimal, and under certain conditions, also mean delay optimal. Motivated by this, we define another partial information version $\phi_{IC}^{(S)}$, of $\phi_{IC}$ below **under the assumption** that the inner clique $C_1$ has exactly one node, i.e., $|C_1| = 1$. We refer to this queue as **Queue c** (see Fig. 6).

**Proof Sketch.** The proof uses a Lyapunov drift argument and invokes the Foster-Lyapunov theorem to prove that the system backlog process $Q(t) := [Q_1(t), \ldots, Q_N(t)], t \geq 0$ is positive recurrent. The details of the proof can be found in the supplementary material in Sec. XI-U.

**Remark 26:** In [23, Sec. XI.B] we discuss how to implement $\phi_{IC}^{(S)}$ in a decentralized manner. As mentioned before, another commonly used technique to reduce control traffic for scheduling is to disseminate state information periodically. In [23, Sec. XI.A.D] we develop a family $\{\phi_{IC}^{(S)}(T), T \geq 1\}$ of such policies that take scheduling decisions based control information obtained only every $T$ time slots. We show that the family is throughput-optimal and also that they display some interesting delay properties with respect to the dissemination period, $T$.

**IX. Simulation Results**

In this section we numerically compare the performance of the various policies we have proposed and analyzed in the preceding sections. To begin with, we simulate the mean delay performances of the policies for the path graph network with $N = 3$, discussed in Sec. IV and compare them against the MaxWeight scheduling policy. To recapitulate, $\pi_{TD}^{(3)}$ and $\pi_{BU}^{(3)}$ are the Top-Down and Bottom-Up policies respectively, $\tilde{\pi}_{IQ}^{(3)}$ is the delay optimal policy defined in IV-A and $\pi_{IQ}^{(3)}$ is the throughput-optimal non-MSM policy defined in IV-C. In every slot $t \geq 0$, MaxWeight simply serves Queues 1 and 3 if $Q_1(t) > Q_3(t) > Q_2(t)$ and Queue 2 otherwise. Obviously, this policy requires more state-information than any of the others. We simulate these policies when the arrival processes to the three queues are independent Bernoulli processes of rates $s \times [0.25, 0.74, 0.25], s \in [0, 1]$, i.e., the inner queue has a high arrival rate, and $s \times [0.74, 0.25, 0.74]$, i.e., the outer two queues have the high arrival rates. The results are shown in Figures 7a and 7b. As claimed in Thm. 7, $\tilde{\pi}_{IQ}^{(3)}$ performs best, showing in mean delay of up to 30% less than MaxWeight near $s = 1$ (in fact, the reduction in delay becomes more pronounced as $s$ approaches 1) and 38% less than $\pi_{TD}^{(3)}$. Notice that in both plots MaxWeight does not perform as well as $\tilde{\pi}_{IQ}^{(3)}$ showing that it does not prioritize the middle queue frequently enough.

Moving on, although we have proved our stability results with Bernoulli arrival processes, we now provide a simulation study which suggests that these results seem to hold for more general arrival processes. Also note that the stochastic ordering proof of Thm. 7, i.e., the delay optimality of policy $\tilde{\pi}_{IQ}^{(3)}$, being a sample path optimality argument, does not take into account the fact that the arrival processes to the queues are Bernoulli. It is, hence, equally valid for other types of arrival processes as...
However, the plots also suggest that the inner bound $\Lambda_{\alpha}$ is not large enough. As before, our policies on the 3-queue path-interference graph with non-Bernoulli arrivals, $\lambda_i \in \{0,5,0.74,0.25\}$, are given in Table II due to space constraints. The table shows that our proposed policies do outperform MaxWeight in all cases. Note that the arrival rate vectors raise to their $\alpha$th powers, with $\alpha > 0$. This policy has been observed to show smaller sum queue lengths (than MW) with smaller $\alpha$ [36].

The result for the SoC networks is, in particular, quite interesting, since one expects that situations may arise in which the performance of MaxWeight, $\rho^{(3)}$, is essentially $\equiv \rho^{(3)}$, and for every $t \geq 1$, $E_{A_j}(t) = \xi_{0,1} + \xi_{1,1} = \lambda_j$. Both plots (Figures 9a and 9b) bear out the fact that even with non-Bernoulli arrivals, $\rho^{(3)}$ shows the best delay performance beating the closest competitor by at least 34%. Again, in both plots we see that MaxWeight performs about just as well as the Top-Down and Bottom-Up policies, suggesting that it does not prioritize the inner queue “enough.” We now move on to the performance of the randomized policy $\theta^{(3}_D$, indexed by the randomization parameter $\gamma \in [0,1]$. In Sec. IV-D we derived an inner bound on the set of arrival rates that the policy can stabilize for a given $\gamma$, its stability region, and showed that $\Lambda_{\gamma} \not\subseteq \Lambda_3$ as $\gamma \uparrow 1$. The plot in Fig. 10, simulated with $\gamma = 0.5$, 0.55, and 0.6 help corroborate our analysis. However, the plots also suggest that the inner bound $\Lambda_{\gamma}^{(3)}$ is actually not very tight. Further study is required to establish better bounds on this region.

Moving on to larger path graphs, recall that in Sec. V-C we proposed a policy-splicing procedure to derive low delay QNB-MSM scheduling policies for path graphs with arbitrary number of queues. We demonstrate the performance of these policies in Table II, where we compare our proposed policies with the benchmark MaxWeight ($MW$) and a third policy that is based on a popular scheduler called the “MaxWeight-$\alpha$” scheduler. This last policy, that we denote by $L(MW_{\alpha})$, is an MSM policy, obtained by using the operator $L$ (see Sec. IV-A) to project a modification of MaxWeight ($MW$) called MaxWeight-$\alpha$ onto $T^{(5)}_{\alpha}$. The $MW_{\alpha}$ policy, studied in [35] and [17], is essentially $MW$ with all queue lengths raised to their $\alpha$th powers, with $\alpha > 0$. This policy has been observed to show smaller sum queue lengths (than $MW$) with smaller $\alpha$ [36].

The table shows that our proposed policies do outperform MaxWeight in all cases. Again, the arrival rate vectors have not been shown in Table II due to space constraints. We have reported the vectors in Sec. XI-V of the Appendix. Recall that the analysis of throughput optimality was limited to $N = 9$ queue systems. In Row 3, we perform the splicing procedure (Sec. V-C) to produce a QNB-MSM policy for a system with $N = 15$ queues and show that it outperforms both the benchmark policies. Finally, Row 1 of the table shows an arrival rate vector for which our proposed queue length-agnostic policy does worse and $L(MW_{\alpha})$ shows the smallest sum queue length. In light of Thm. 18, this should not be entirely surprising. Moreover, the loss in performance is small. We move on to simulations of the policies proposed for the second class of conflict graphs discussed in this article, namely, Cluster-of-cliques graphs. The first, shown in Fig. 11, is a Star-of-Cliques (SoC) networks comprising 4 cliques and a total of 6 queues. The second network is the LAoC network shown in Fig. 2a. It consists of 4 cliques and a total of 9 queues. Table III shows the result of simulating $\phi$, $\theta^{(5}_D$, defined in Sec. VII-B) and $MW$ on these networks. We see that the proposed policies consistently perform better than the benchmarks.

The result for the SoC networks is, in particular, quite interesting, since one expects that situations may arise

\[ \text{Fig. 7. Simulation results for the path-graph network with } N = 3 \text{ for Bernoulli packet arrival processes. The mean delay performances of all deterministic policies discussed in Sec. IV are shown in Figures (a) and (b), and compared with the MaxWeight scheduling policy [10].} \]

\[ \text{Fig. 8. The transition probability diagram of the Markovian arrival process.} \]

\[ 1 - p \quad 0 \quad p \quad 1 - q \]

\[ \text{Fig. 9a) Delay performance of the policies } \hat{\pi}_{IQ}, \pi_{IQ}^{(3)}, \text{ MaxWeight, } \pi_{TD}^{(3)} \text{ and } \pi_{TD}^{(3)} \text{ along the trajectory } \lambda(s) = s \times [0.25, 0.74, 0.25], \ s \in [0, 1], \text{ in the capacity region } \Lambda_3. \]

\[ \text{Fig. 9b) Delay performance of the policies } \hat{\pi}_{IQ}, \pi_{IQ}^{(3)}, \text{ MaxWeight, } \pi_{TD}^{(3)} \text{ and } \pi_{TD}^{(3)} \text{ along the trajectory } \lambda(s) = s \times [0.74, 0.25, 0.74], \ s \in [0, 1], \text{ in the capacity region } \Lambda_3. \]

\[ \text{Fig. 10. The transition probability diagram of the Markovian arrival process.} \]

\[ \text{If, in slot } t, \text{ the arrival process was in State } 0, \text{ i.e., } A(t) = 0, \text{ then in slot } t + 1, \ A(t + 1) = 1 \text{ with probability } p \text{ and } A(t + 1) = 0 \text{ with probability } 1 - p. \]

\[ \text{well. Our simulations bear out this fact. Once again, we simulate our policies on the 3-queue path-interference graph with Markovian arrival processes as described below. The arrivals to every queue form a two-state stationary discrete-time Markov chain (DTMC), i.e., } \{A_i(t), \ t \geq 1\} \text{ forms a DTMC. As before, } A_i(t) = 1 \text{ refers to the arrival of a packet into Queue } i \text{ and } A_i(t) = 0 \text{ refers to no arrivals.} \]

\[ \text{Fig. 8 shows the transition probability diagram of a generic two-state DTMC. For Queue } j, \text{ the stationary probability of the arrival being in State } i, \ i \in \{0,1\} \text{ is given by } \xi_{i,j}; \text{ obviously, } \xi_{0,j} + \xi_{1,j} = 1, \forall j \in \{1,2,3\}. \supset \text{Suppose the transition probabilities of the process for Queue } j \text{ are given by } P(A_j(t+1) = 1|A_j(t) = 0) = p_j, \text{ and } P(A_j(t+1) = 0|A_j(t) = 1) = q_j. \text{ From basic Markov chain theory, we know that } \xi_{1,j} = \frac{p_j}{p_j + q_j}, \text{ and for every } t \geq 1, \text{ } E_{A_j}(t) = \xi_{0,j} + \xi_{1,j} = \lambda_j. \]

\[ \text{Both plots (Figures 9a and 9b) bear out the fact that even with non-Bernoulli arrivals, } \pi_{IQ}^{(3)} \text{ shows the best delay performance beating the closest competitor by at least 34%. Again, in both plots we see that MaxWeight performs about just as well as the Top-Down and Bottom-Up policies, suggesting that it does not prioritize the inner queue “enough.” We now move on to the performance of the randomized policy } \theta^{(3}_D, \text{ indexed by the randomization parameter } \gamma \in [0,1]. \text{ In Sec. IV-D we derived an inner bound on the set of arrival rates that the policy can stabilize for a given } \gamma, \text{ its stability region, and showed that } \Lambda_{\gamma} \not\subseteq \Lambda_3 \text{ as } \gamma \uparrow 1. \text{ The plot in Fig. 10, simulated with } \gamma = 0.5, 0.55, \text{ and } 0.6 \text{ help corroborate our analysis. However, the plots also suggest that the inner bound } \Lambda_{\gamma}^{(3)} \text{ is actually not very tight. Further study is required to establish better bounds on this region.} \]

\[ \text{See Sec. II-A for details.} \]
Fig. 9. Simulation results for the path-graph network with $N = 3$ for Markovian packet arrival processes. For all plots, and every $i \in \{1, 2, 3\}$ the transition probabilities of the arrival process (see Fig. 8) are chosen as follows $p_i = 0.10$, and $q_i = (1 - \frac{1}{\lambda_i}) p_i$.

Fig. 10. Illustrating the loss of stability due to the "Flow-in-the-Middle" problem discussed in Sec. IV-D. We compare the delay performance of the policy $\rho_1^{(s)}$ along the trajectory $\lambda(s) = s \in [0.74, 0.25, 0.25]$, $s \in [0, 1]$, in the capacity region $\Lambda_3$. For every $s \leq 1$, the arrival rate vector lies within the interior of $\Lambda_3$ and is, hence, stabilizable. The policies can be seen to render the system unstable much before the system load parameter $s$ hits 1.

Fig. 11. The Star-of-Cliques (SoC) network used to study the performance of $\hat{\phi}$. Simulation results are reported in Table III.

wherein only two of the three peripheral cliques and $C_1$ are nonempty. In such a case, $\hat{\phi}$ would serve $C_1$, giving up the chance to serve both the peripheral nonempty cliques simultaneously and remove 2 packets from the system in a single slot, which is what $MW$ might have attempted, if the queues therein were large enough. If, for example, in some slot $t$, $C_2$ is empty, while $Q_{1,1}(t) = 1, Q_{3,1}(t) = 5$ and $Q_{4,1}(t) = 2$, $\hat{\phi}$ still serves only $Q_{1,1}$ (1 packet transmitted) while MaxWeight serves both $Q_{1,1}$ and $Q_{4,1}$ (2 packets transmitted). Why $\hat{\phi}$ still performs better requires more investigation and will be a focus of our future work.

X. CONCLUSION AND FUTURE WORK

In this paper, we began by studying the scheduling of transmissions over a class of noncollocated interference networks that we called "path-graph interference networks." We provided sufficient conditions for queue nonemptiness based (QNB) policies to be throughput-optimal over these networks. We then provided a complete characterization of the class of maximum size matching-QNB (MSM-QNB) policies on path-graphs with 3 queues and showed that this class contains stable, delay-optimal and even unstable policies. We then saw how priority policies for smaller path-graphs can be combined to construct QNB policies for larger networks. Next, we showed that policies so constructed are not MSM, but can be made MSM using a projection operator. We also showed how the delay properties of these MSM policies can be further improved by using certain observations of the nature of scheduling policies in $\hat{\Pi}^{(N)}$. We then showed that there cannot exist QNB policies that are uniformly delay optimal over the entire capacity region, for any path graph network with $N \geq 4$ links.

Motivated by wireless networks commonly used for IoT-type applications, we introduced a new class of interference networks, called the "Cluster-of-Cliques" networks and studied two subclasses, namely, the Star-of-Cliques and the Linear-Arrays-of-Cliques networks. We then constructed QNB scheduling policies for both these classes, studied their stability and delay properties, and also developed a T.O. protocol that requires no explicit exchange of even occupancy information. Our simulation results showed that the QNB...
policies we have developed perform better than existing scheduling policies that require complete knowledge of the system backlog in every slot.

In short, MaxWeight and policies based on it (such as MaxWeight-α) have been known to suffer from two major implementation issues, namely (i) disseminating queue length information across the network (or reporting it to some centralized scheduling entity), and (ii) finding the maximum weight independent set (MWIS), which for general conflict graphs, is famously an NP-hard problem. However, in the context of the current article, the latter problem is simplified. In fact, there exist dynamic programming approaches to solve the MWIS problem in linear time for path graphs. The outstanding issue in computing schedules, therefore, is one of information dissemination. Our work provides rigorous theoretical evidence that suggests that once the MWIS problem is simplified, detailed queue length information is (almost) irrelevant.

Future work will include extending these throughput-optimality results to non-Bernoulli arrival processes, and proving the throughput-optimality of $\pi^{TD}$ and $\pi^{MU}$ for general $N$-queue path graph networks. Finally, we would also like to explore such reduced state information based scheduling policies for more general conflict graphs and the existence of graphs that do not permit stable QNB scheduling policies.

**REFERENCES**

[1] M. Honrubia. (2017). *Industrial IoT is Booming Thanks to a Drop in Sensor Prices.* [Online]. Available: https://www.ennomotive.com/

[2] E. Dahlman, S. Parkvall, and J. Skold, 4G: LTE/LTE-Advanced for Mobile Broadband. New York, NY, USA: Academic, 2013.

[3] M. Alasti, B. Neekzad, J. Hui, and R. Vannithamby, “Quality of service in WiMAX and LTE networks [topics in wireless communications],” *IEEE Commun. Mag.*, vol. 48, no. 5, pp. 104–111, May 2010.

[4] M. Lohstroh, H. Kim, J. C. Eidson, C. Jerad, B. Osyk, and E. A. Lee, “On enabling technologies for the Internet of important things,” *IEEE Access*, vol. 7, pp. 27244–27256, 2019.

[5] Z. Kong, D. H.-K. Tsang, B. Bensaou, and D. Gao, “Performance analysis of IEEE 802.11e contention-based channel access,” *IEEE J. Sel. Areas Commun.*, vol. 22, no. 10, pp. 2095–2106, Dec. 2004.

[6] H. Wu, X. Wang, Q. Zhang, and X. Shen, “IEEE 802.11 e enhanced distributed channel access (EDCA) throughput analysis,” in *Proc. IEEE Int. Conf. Commun.*, vol. 1, June 2006, pp. 223–228.

[7] H. Zhu and I. Chlamtac, “Performance analysis for IEEE 802.11e EDCA service differentiation,” *IEEE Trans. Wireless Commun.*, vol. 4, no. 4, pp. 1779–1788, Jul. 2005.

[8] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, “6TiSCH: Deterministic IP-enabled industrial Internet (of things),” *IEEE Commun. Mag.*, vol. 52, no. 12, pp. 36–41, Dec. 2014.

[9] D. Dujovne, L. Griceo, M. Palattella, and N. Accettura, “6tisch on-the-fly schedulingdraft-dujovne-6tisch-on-the-fly-04,” Internet Eng. Task Force, Fremont, CA, USA, IETF 6TISCH WG, Tech. Rep., 2015.

[10] L. Tassiulas and A. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,” *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992-49, 1994.

[11] S. Sanghavi, D. Shah, and A. S. Willsky, “Message passing for maximum weighted independent set,” *IEEE Trans. Inf. Theory*, vol. 55, no. 11, pp. 4822–4834, Nov. 2009.

[12] E. Modiano, D. Shah, and G. Zussman, “Maximizing throughput in wireless networks via gossiping,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 34, no. 1, pp. 27–38, Jun. 2006.

[13] L. Jiang and J. Walrand, “Approaching throughput-optimality in distributed CSMA scheduling algorithms with collisions,” *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 816–829, Jun. 2011.

15 The bottleneck in the stability proofs appears to be combinatorial in nature. Specifically, several of the $2^N$ realizations of $\Theta(1)$ needed checking to establish stability. The proof for general $N$ will, therefore, require a suitable counting argument that can reduce the number of realizations that need to be checked.
Aditya Gopalan (Member, IEEE) received the B.Tech. and M.Tech. degrees in electrical engineering from the Indian Institute of Technology Madras and the Ph.D. degree in electrical engineering from The University of Texas at Austin. He was an Andrew and Erna Viterbi Post-Doctoral Fellow at the Technion-Israel Institute of Technology. He is currently an Assistant Professor and INSPIRE Faculty Fellow at the Indian Institute of Science, Electrical Communication Engineering. His research interests include machine learning and statistical inference, control, and algorithms for resource allocation in communication networks.

Anurag Kumar (Fellow, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT) Kanpur and the Ph.D. degree in electrical engineering from Cornell University. He was with Bell Labs, Holmdel, NJ, USA, for over six years. Since then, he has been on the faculty of the ECE Department, Indian Institute of Science (IISc), Bengaluru, where he is currently the Director of the Institute. His area of research is communication networking, and he has recently focused primarily on wireless networking. He is a Fellow of the Indian National Science Academy (INSA), the Indian National Academy of Engineering (INAE), and the Indian Academy of Sciences (IASc). He was an Associate Editor of the IEEE TRANSACTIONS ON NETWORKING and the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS.