Queue Length Stability in Trees under Slowly Convergent Traffic using Sequential Maximal Scheduling

Saswati Sarkar and Koushik Kar

***Abstract*— In this paper, we consider queue-length stability in wireless networks under a general class of arrival processes that only requires that the empirical average converges to the actual average polynomially fast. We present a scheduling policy, *sequential maximal scheduling*, and use novel proof techniques to show that it attains $\frac{2}{3}$ of the maximum stability region in tree-graphs under primary interference constraints, for all such arrival processes. For degree bounded networks, the computation time of the policy varies as the the logarithm of the network size. Our results are a significant improvement over previous results that attain only $\frac{1}{2}$ of the maximum throughput region even for graphs that have a simple path topology, in similar computation time under stronger (i.e., Markovian) assumptions on the arrival process.**

I. Introduction

Scheduling for maximum throughput is a key operational goal in any wireless network. Scheduling of links must be done such that no two "interfering" links are scheduled at the same time. Under random packet arrivals, the scheduling problem can be posed in a stochastic decision framework where the goal is to attain stability of queues over the largest possible set of arrival vectors. Queues are said to be stable, or rather queue-length-stable, if their expected lengths are finite in each slot. The set of arrival rate vectors for which the network is stabilized under some scheduling policy is referred to as the maximum throughput region. In a seminal work, Tassiulas *et al.* have characterized the maximum throughput region and also provided a scheduling strategy that attains this throughput region in any given wireless network [19]. Subsequently, several policies have been shown to attain (for the general and certain important special cases of the problem) either the maximum throughput region [1], [5], [6], [16], [17], [18] or a guaranteed fraction of it [2], [3], [10], [11], [20], while requiring lower computation time.

In this paper, we consider primary interference constraints which requires that a set of links can be simultaneously scheduled if and only if they constitute a matching. This interference model is also referred to as the node exclusive spectrum sharing model and arises when every node has a single transceiver and a unique frequency in its two-hop neighborhood. We focus on the special class of tree graph topologies which are very important from a practical perspective. For instance, in many applications, nodes organize themselves into a spanning tree and communication is confined to the tree edges only. These include various data gathering or data distribution applications where nodes either send data to, or collect data from, a single source node. We consider an arrival process which only requires that the empirical average converges to the actual average polynomially fast. This assumption is satisfied by a large number of arrival processes including Markovian, periodic, bounded-burstiness* arrival processes†. We consider a class of simple scheduling policies, which allows a link to contend if its neighboring links have equal or lower queue lengths, and links are scheduled among the contending links using "maximal scheduling". Maximal scheduling only ensures that if a link contends then either the link or one of its adjacent links is scheduled. We prove that this queue length based maximal scheduling policy attains $2/3$ of the maximum throughput region for tree graphs and primary interference model. Furthermore, the policy does not use any knowledge of the arrival rates, and requires each link to learn only the queue lengths and the scheduling decisions of its adjacent links. Under the reasonable assumption that control message exchanges have to satisfy primary interference constraints as well, the algorithm can be implemented in a fully distributed manner in $O(\Delta \log \Delta \log n)$ time, where n is the number of links and Δ is the maximum node degree in the network.

The main contributions of this paper with respect to existing research in this area are as follows. Firstly, we obtain throughput guarantees under the notion of queue length stability for a large class of "polynomially convergent" arrival processes which includes, but is not limited to, Markov processes. In most of the existing literature, the proofs, and hence the throughput guarantees, (a) rely on Lyapunov arguments and Foster's theorem [7] and (b) equivalence between the positive Harris recurrence and fluid stability of a queueing system [4], both of which apply only when the queue length process is Markovian, which in turn holds only when the arrival process is Markovian. Such assumptions on the arrival process do not often hold in reality, as recent Internet traffic analysis has shown. For non-Markovian arrival processes, throughput guarantees are known only under the notions of (a) rate stability which only requires that the input rates equal the output rates [2], [3], or (b) vanishing tail probability which requires that tail probabilities of queue lengths approach zero [13]‡. Note that several applications require finite expected delay, and therefore finite expected queue lengths, which rate stability does not guarantee. Thus, unlike existing results, our policy is able to guarantee queue length stability (and therefore finite expected delay) to a large class of realistic traffic models. In our work, throughput guarantees are obtained using non-standard proof techniques since the arrival process and therefore the queue length process is not Markovian in this system. Thus, both the policy and the proofs for the throughput guarantees are important contributions of this paper.

Secondly, for tree networks, our policy provides an excellent tradeoff between performance and complexity, which is better than those in the existing literature in different ways. While existing policies that attain maximum throughput in a similar setting require

S. Sarkar is with the Department of Electrical Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA. Email: swati@seas.upenn.edu. K. Kar is with the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, 12180, USA. Email: kark@rpi.edu. This is an extended version of a paper that appeared in the Annual Allerton Conference, Urbana-Champaign, 2006.

*An arrival process is said to have bounded-burstiness if the number of arrivals in any time interval of length t differs from ρt by at most a constant σ that does not depend on t; here ρ is the long-term arrival rate.

†Note that for Markov, periodic and bounded-burstiness arrival processes the empirical mean converges to the actual mean exponentially fast. The processes we consider may therefore have slower convergence.

‡The notion of stability Neely *et. al.* [13] consider requires that $\lim_{V\to\infty} g(V) = 0$ where $g(V) = \limsup_{t\to\infty}(1/t)\sum_{\tau=0}^{t-1}\mathbf{P}\left(Q_i(\tau) > V\right)$ where $Q_i(\tau)$ is the queue length at time τ for link i. But, this does not guarantee that $\sum_{V=1}^{\infty}\mathbf{P}\left(Q_i(\tau) > V\right)$ can be upper bounded by a quantity that does not depend on τ and thus it does not follow that the expected queue length in a link i can be upper bounded by a quantity that is independent of τ in any slot τ.

$O(n\Delta)$ time [1], [5], [6], [17], [18], our scheduling algorithm can be implemented $O(\Delta \log \Delta \log n)$ time. Therefore, our approach reduces the scheduling complexity significantly, at the cost of $1/3$ of the throughput region in the worst case. On the other hand, existing maximal scheduling based scheduling policies that require $O(\Delta)$ or $O(\Delta \log n)$ time [2], [3], [11], [20], [10], have been shown to attain at most 1/2 of the maximum throughput region. With respect to this class of work, therefore, we are able to improve the throughput guarantee significantly (from $1/2$ to $1/3$), with a modest increase in the computational complexity.

We now briefly review existing policies with provable throughput guarantees under the primary interference model. Tassiulas *et al.* [19], [18] have obtained policies that attain the maximum throughput region, which can be implemented in fully distributed manner using gossip based algorithms [6]. Distributed implementation of these policies however require $O(n)$ *communication rounds*, where each communication round involve message exchanges by nodes with their neighbors; the time complexity of these policies is therefore $O(n\Delta)$. Lin *et al.* [11] and Wu *et al.* [20] have shown that maximal scheduling is guaranteed to attain at least half of the maximum throughput region under the primary interference model. It has also been shown that the above performance guarantee is tight, i.e., in the worst case some maximal scheduling policies attain at most half the maximum throughput region even in simple networks like paths with only three links [2]. An arbitrary maximal scheduling policy cannot therefore attain a worst-case performance ratio better than $1/2$ even in the special case of trees. Maximal scheduling can be implemented in a distributed manner in $O(\log n)$ communication rounds, which translates to a time complexity of $O(\Delta \log n)$ in trees under primary interference constraints[§]. Lin *et al.* [10] proved that a random access scheme, where links access the medium with a probability that depends on their and their interferers' queue lengths, attains $1/3$ of the throughput region while requiring $O(1)$ communication rounds, or $O(\Delta)$ computation time in trees under primary interference constraints. Dimakis *et al.* [5] have shown that a greedy maximal weight scheduling attains the maximum throughput region in certain classes of networks; Brzezinski *et. al.* [1] have shown the above result for trees. The number of the communication rounds required by the above algorithm however depends on the diameter of the network, and the computation time is therefore $O(n\Delta)$ in the worst case. Therefore, our algorithm has a lower computation time than that in [1], [5], unless the diameter of the network is sufficiently small. Salonidis *et al.* [16] designed another policy that attains the maximum throughput region in trees; the policy however requires knowledge of the arrival rates in all links and therefore must be recomputed every time these rates change. The throughput guarantees obtained in all the above papers, except those in [2], [3], critically depend on the assumption that arrival processes are Markovian which we do not assume. As mentioned before, the throughput guarantees obtained in [2], [3] do not guarantee that expected queue lengths are finite which we ensure. Also, our policy attains a better throughput guarantee as compared to those in [2], [3], [11], [20], [10] and lower computation time as compared to those in [5], [19], [18], [6].

The paper is organized as follows. We describe the system model and the terminology in Section II. We present our policy and performance guarantee for (a) the special case that the network topology is a path in Section III, and (b) the case that the network topology is a tree in Section IV. We conclude in Section V.

[§]Several well-known distributed randomized algorithms for computing maximal schedules (e.g., [12]) need to exchange in each round at most one control message in each link. For attaining the above in trees under primary interference constraints, the nodes need to know their distances from the root which can be accomplished in a pre-processing step. The nodes with even-valued (odd-valued) distances are referred to as even (odd) nodes. Next, each round is divided in 2Δ sub-rounds, and the even (odd) nodes communicate the control messages to their children in the first (last) Δ sub-rounds. Clearly, the communications in the same sub-round do not interfere and can therefore be executed simultaneously; thus, each sub-round consumes constant time.

II. System Model

We consider the scheduling problem at the medium access control (MAC) layer of the network. We assume that time is slotted, and each packet takes exactly one slot for transmission. Therefore, a link transmission schedule must be computed at the beginning of every slot, and is used to transmit packets in that slot.

A wireless network topology can be modeled as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ and $\mathcal{L}$ respectively denote the sets of nodes and links. Each (undirected) link $(u, v) \in \mathcal{L}$ therefore denotes whether nodes u and v can hear each other's signals. The link set $\mathcal{L}$ depends on the transmission power levels of nodes and the local propagation conditions of the wireless channel. We assume that $\mathcal{G}$ is a *degree-bounded tree* with maximum degree-bound Δ. Without loss of generality, we will assume that $\mathcal{G}$ is connected; otherwise, our algorithm can be executed independently in each of the maximally connected subgraphs of $\mathcal{G}$. Let $n = |\mathcal{N}| = |\mathcal{L}| + 1$.

Each link is associated with a unique identifier (id). Let $\mathcal{L}_u$ be the set of links incident on node u. Two links are *adjacent* if and only if they have a node in common. By this definition, a link is always adjacent to itself. Let $\mathcal{N}_l$ be the set of links adjacent to l. For any path $\mathcal{P}$, let $\mathcal{C}_\mathcal{P}$ denote the set of links that are adjacent to the first and last link of $\mathcal{P}$ and are not part of $\mathcal{P}$.

Since we consider the primary interference model, two adjacent links "interfere" with each other and cannot be scheduled simultaneously, i.e., any two links (u, v) and (u', v') cannot be scheduled together if $u \in \{u', v'\}$ or $v \in \{u', v'\}$. Thus a valid schedule in any slot must correspond to a matching or a set of links none of which are adjacent to each other. Note that the primary interference model arises when the only transmission constraint is due to the single transceiver constraint at every node.

Each session represents a triplet (i, u, v) where i is the identifier associated with the session and u and v are source and destinations of the session. At the MAC layer, each session traverses only one link, but multiple sessions may traverse a link. Let $\hat{S}$ denote the set of sessions in the network.

Next we state our assumptions on the packet arrival process. Let $\hat{A}_i(t_1, t_2)$ denote the number of packets arriving at session i in interval $(t_1, t_2]$. We assume that $\hat{A}_i(t, t+1) \leq \hat{\sigma}_i \ \forall \ t$, session i, where $\hat{\sigma}_i$ is an integer for each i, and $\max_i \hat{\sigma}_i \geq 1$. Further, there exists a constant $\alpha > 1$ and a vector $\vec{\hat{\rho}} = (\hat{\rho}_1, \ldots, \hat{\rho}_{|\hat{S}|})$ such that the empirical average of the arrivals in the system in T slots converges to $\vec{\hat{\rho}}$ at a rate faster than $\frac{1}{T^\alpha}$. Mathematically, there exists $\hat{\chi}_\delta > 0$ such that for every i, $0 \leq t_3 < t_4$ and $\delta > 0$,

$$\mathbf{P}\left\{\left|\frac{\hat{A}_i(t_3, t_4)}{t_4 - t_3} - \hat{\rho}_i\right| \geq \delta\right\} < \frac{\hat{\chi}_\delta}{(t_4 - t_3)^\alpha}. \tag{1}$$

Clearly, $\hat{\chi}_\delta$ is a non-increasing function of δ. Note that (1) implies that the empirical average of the packet arrivals converges in probability to $\vec{\hat{\rho}}$ polynomially fast. Also, most commonly used arrival processes, e.g., bounded-burstiness, periodic, i.i.d., and Markovian arrival processes with finite state space, satisfy the above assumption.

Next we introduce a few definitions.

Definition 1: The network is said to be *stable* if there exists a finite real number B_0 such that for any $t > 0$, $\mathbf{E}Q_l(t) \leq B_0$, $\forall \ l \in \mathcal{L}$.

We consider a virtual-queue Q_l associated with link l that contains all packets waiting for transmission for all sessions that traverse l. All packets arriving in a session traversing l are routed to Q_l and

whenever l is scheduled the head of line packet in Q_l is transmitted. Note that the virtual queue in a link $l = (u, v)$ may contain packets of sessions traversing l in both directions $u \rightarrow v$ and $v \rightarrow u$. Let $Q_l(t)$ be the queue length at link l at the beginning of slot t (after the arrivals but before the transmissions in t). For simplicity, we assume that $Q_l(0) = 0$ for all $l \in \mathcal{L}$; our results can be generalized for any positive, but finite values of $\vec{Q}(0)$.

Let $A_l(t_1, t_2)$ denote the number of packets arriving in virtual queue Q_l, or more simply at link l, in interval $(t_1, t_2]$. Clearly, there exists integers σ_l such that $A_l(t, t+1) \leq \sigma_l \ \forall \ t, l \in \mathcal{L}$, and $\max_{l \in \mathcal{L}} \sigma_l \geq 1$. Also, there exists an arrival rate vector $\vec{\rho} = (\rho_1, \ldots, \rho_{|\mathcal{L}|})$ such that the empirical average of the arrivals in each link in T slots converges to $\vec{\rho}$ at a rate faster than $\frac{1}{T^\alpha}$. Mathematically, there exists $\chi_\delta > 0$ such that for every $l \in \mathcal{L}$, $0 \leq t_3 < t_4$ and $\delta > 0$,

$$\mathbf{P}\left\{\left|\frac{A_l(t_3, t_4)}{t_4 - t_3} - \rho_l\right| \geq \delta\right\} < \frac{\chi_\delta}{(t_4 - t_3)^\alpha}. \tag{2}$$

Again, χ_δ is a non-increasing function of δ. We refer to ρ_i as the arrival rate for link i.

Clearly, the network is stable if and only if the expected queue length at each link remains finite at all time.

Definition 2: The *throughput region* of a scheduling policy is the set of arrival rate vectors $\vec{\rho}$ satisfying (2) for which the network is stable under the policy.

Definition 3: An arrival rate vector $\vec{\rho}$ is said to be *feasible* if it is in the throughput region of some scheduling policy.

Definition 4: The *maximum throughput region* Λ^* is the set of all feasible arrival rate vectors.

If an arrival rate vector $\vec{\rho} \in \Lambda^*$, then (a) $\rho_l \geq 0 \ \forall \ l \in \mathcal{L}$ and (b) $\forall \ u \in \mathcal{N}, \ \sum_{l \in \mathcal{L}_u} \rho_l < 1$ [9].

Definition 5: A scheduling policy π is said to *guarantee a fraction ν of the maximum throughput region* if its throughput region, Λ_π, satisfies the following condition: for any $\vec{\rho} \in \Lambda^*$, $\nu\vec{\rho} \in \Lambda_\pi$.

Loosely speaking, if scheduling policy S guarantees a fraction ν of the maximum throughput region, then its throughput region is at least ν fraction of the maximum throughput region.

We seek to prove that the scheduling policies we propose guarantee $2/3$ of the maximum throughput region. We therefore need to show that for any arrival rate vector $\vec{\rho}$ such that

$$\forall \ u \in \mathcal{N}, \ \sum_{l \in \mathcal{L}_u} \rho_l \ < \ 2/3, \tag{3}$$

$\vec{\rho}$ is in the throughput region of our policies. We assume (3) henceforth.

$$\text{Let } \xi \ = \ 2/3 - \max_{u \in \mathcal{N}} \sum_{l \in \mathcal{L}_u} \rho_l.$$

Finally, we describe the maximal scheduling policy, which will be a key constituent in our scheduling policy presented later in the paper. A maximal scheduling policy schedules a subset S of links such that (i) every link in S has a packet to transmit, (ii) no link in S interferes with any other link in S, (iii) if a link l has a packet to transmit, then either l or a link adjacent to l, is included in S.

III. Scheduling Policy for a path

In this section, we consider a graph $\mathcal{G}$ that is a *simple* path, i.e., $\mathcal{L}$ corresponds to a sequence of links such that the consecutive links in the sequence are adjacent. In Section III-A, we describe our scheduling policy, which we call *Sequential Maximal Path Scheduling*, and in Section III-B we prove that this policy attains $2/3$ of the maximum throughput region.

Sequential Maximal Path Scheduling

Initial Step: Each link sets its status to to "un-decided" if it has a packet to transmit, and to "un-scheduled" otherwise.

Iterative Step:: For $k = 1$ to $k = 2$, execute Phase k, as given below:
Phase k : A link in the path contends if and only if (a) it is un-decided, (b) its adjacent links are un-scheduled or un-decided, (c) its queue length is not less than that of its adjacent links that satisfy conditions (a) and (b). A contending link sets its status to "scheduled" if its adjacent links do not contend or have higher id than it; links that are adjacent to scheduled links set their status to "un-scheduled".

Terminal Step: Compute a maximal schedule among the links that are un-decided and whose adjacent links are un-scheduled or un-decided. Set the status of the links selected in the maximal schedule to "scheduled", and the status of the links that are adjacent to scheduled links as "un-scheduled".

Fig. 1. Sequential Maximal Path Scheduling Algorithm

A. Sequential Maximal Scheduling in Paths

We describe the Sequential Maximal Path Scheduling policy in Figure 1.

Next we illustrate the Sequential Maximal Path Scheduling algorithm using the example shown in Figure 2. The path graph shown in the figure consists of 10 links whose queue-lengths are shown. Using our scheduling algorithm, only link 9 will be scheduled in Phase 1, link 7 will be scheduled in phase 2 and link 5 will be scheduled in phase 3. The terminal step will compute a maximal schedule amongst the links 1, 2, 3, which can be either links $\{1, 3\}$ or only link 2.

We now provide the intuition behind the design.

1) The iterative step of Sequential Maximal Path Scheduling policy provides higher priority to links whose queue lengths are higher than that of their adjacent links. This ensures that that a link can not be congested in isolation. Specifically, if links in a segment of $\mathcal{G}$ of length 5 or less have high queue lengths, then with a high probability, at least one link that is not in the segment but is adjacent to a link in the segment has high queue length as well (lemma 2, Section III-B.1). *Thus, if a link is congested, then with a high probability all links in a segment of length at least* 6 *are congested (Property 1).* The number 6, which is crucial in the rest of the proof, is attained because of multiple phases in the iterative step.
2) The terminal step of the policy ensures that the scheduling is maximal, which in turn guarantees that *the probability that in any slot t all links in any segment of $\mathcal{G}$ consisting of* 6 *links has high queue lengths is small* (lemma 3, Section III-B.1). If the above happens, then all these links must have packets to transmit for several slots until t. But, then, since the scheduling is maximal, at least 2 links are scheduled in the segment in each of the above slots. Now, the sum of the arrival rates in the links in any segment consisting of 6 links is less than 2 due to (3). Thus, the sum of the queue lengths of the links in such a segment must have been decreasing over all these slots, which implies that all links in the segment can not have large queue lengths in t.

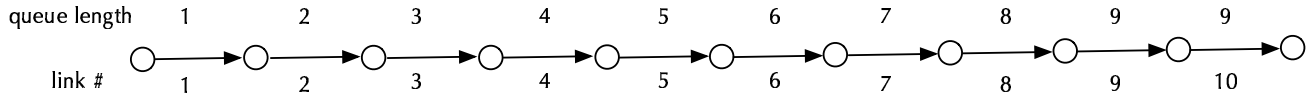

Fig. 2. Path Scheduling Example.

The above (italicized) assertions together imply our main result in this section, Theorem 1, that the queue length in any link becomes large only with a small probability.

Theorem 1: Let $\mathcal{G}$ be a simple path and (3) hold.

1) For each $t > 0$, $l \in \mathcal{L}$, $\mathbf{P}\{Q_l(t) \geq B\} \leq \tau B^{-\alpha}$, where $\tau = \max(1, 151\chi_{\xi/3}) \times (72 \times 4^5 \times 5^5 \max_{l\in\mathcal{L}} \sigma_l)^{\alpha}$.
2) For each $t \geq 0$, $\mathbf{E}(Q_l(t)) \leq \tau \sum_{i=1}^{\infty} i^{-\alpha}$.

Thus, for any $\vec{\rho}' \in \Lambda^*$, $(2/3)\vec{\rho}'$ is in the throughput region of our scheduling policy. In other words, our policy guarantees a $\frac{2}{3}$ fraction of the maximum throughput region. We prove Theorem 1 in Section III-B $-$ the proof proceeds as per steps 1 and 2 above.

We now analyze the time complexity of the policy. The iterative step in Sequential Maximal Path Scheduling can be computed in constant number of communication rounds. The expected number of communication rounds for the terminal step is $O(\log n)$ if maximal scheduling is computed using a distributed randomized algorithm like the one proposed in [12]. Since the graph topology is a path, each communication round takes constant time. This is because the iterative step, and the algorithm proposed in [12], can be executed by exchanging in each round at most one control message through each link in the path. For this purpose, each round can be divided in two sub-rounds, and the even (odd) numbered nodes in the path can transmit control messages to the odd (even) numbered nodes in the first (second) sub-round. Note that the nodes in the path can be numbered once at $t = 0$, i.e., during a pre-processing phase. Clearly, the communications in the same sub-round do not interfere and can therefore be executed simultaneously. Thus, each sub-round consumes constant time. Thus, the expected computation time for Sequential Maximal Path Scheduling is $O(\log n)$.

We now examine in more detail why the iterative step uses multiple phases. First, note that if this step did not have any phase (that is if this step were absent), the policy would be an ordinary maximal scheduling policy, which attains at most $1/2$ the throughput region even for paths of size 3 [2]. Thus, at least one phase is necessary for improving the throughput guarantee to $2/3$. However, our proofs indicate that only one phase does not guarantee Property 1 above which is key towards attaining the $2/3$ throughput guarantee. Nevertheless, we do not have a counterexample to establish that the policy does not attain the $2/3$ throughput guarantee in presence of only one phase. Also, it would be interesting to examine whether the throughput guarantee can be improved beyond $2/3$ by using a larger number of phases, especially since the policy can use $O(\log n)$ phases while still requiring $O(\log n)$ computation time. These intriguing questions constitute interesting topics for future research.

B. Proof of the $2/3$ throughput guarantee

We state and prove the supporting lemmas 2 and 3 in Section III-B.1 and prove Theorem 1 in Section III-B.2.

1) Supporting lemmas: We first state and prove lemma 1 which is used for proving lemma 2, and subsequently state and prove lemmas 2 and 3.

Lemma 1: Consider a path $\mathcal{P} \subseteq \mathcal{G}$ and an arbitrary slot t. Let $\mathcal{P}$ consist of links $l_1, \ldots, l_m$, and satisfy the following properties at t.

1) $Q_{l_i}(t) > 0 \ \ \forall\ i \in \{1, \ldots, m\}$ (*non-emptyness criterion*).
2) If $l \in \mathcal{C}_{\mathcal{P}}$ then $Q_l(t) < Q_{l_j}(t) \ \ \forall\ l_j \in \mathcal{N}_l \cap \{l_1, l_m\}$ (*isolation criterion*).

Consider the iterative step of the Sequential Maximal Tree Scheduling. If $m \in \{1, 2\}$, at least 1 link in $\mathcal{P}$ is scheduled during the first phase at t. If $m = 3$, either l_2 is scheduled during the first phase or two links in $\mathcal{P}$ are scheduled in the first two phases at t. If $m > 3$, at least 2 links in $\mathcal{P}$ are scheduled during the first two phases at t.

Proof: We first show that for any $m \geq 1$ at least 1 link in $\mathcal{P}$ is scheduled during the first phase at t. From the isolation and non-emptyness criteria, at least one link in $\mathcal{P}$ contends in the first phase at t, and the link with the greatest id among the contending links in $\mathcal{P}$ is scheduled. Thus, the first part of the lemma follows.

Now, let $m > 2$. The second and third parts of the lemma follows if at least 2 links in $\mathcal{P}$ are scheduled in the first phase. So, let exactly 1 link in $\mathcal{P}$ be scheduled in the first phase.

Let l_1 be scheduled in the first phase. Thus, $\{l_2, \ldots, l_m\}$ are not scheduled in the first phase and, l_2 does not prevent the contention of any link in the second phase. Consider a path $\mathcal{P}'$ consisting of links $l_3, \ldots, l_m$. Now, since l_{m-1} have not been scheduled in the first phase (since $m > 2$, $l_{m-1} \neq l_1$), from the isolation and non-emptyness criteria, at least one link in $\mathcal{P}'$ contends in the second phase. Using arguments similar to those in the first paragraph, we can show that at least one link in $\mathcal{P}'$ is scheduled in the second phase. Thus, the second and third part of the lemma follow.

The proof is similar if instead of l_1, l_m is scheduled in the first phase. Now, let l_i be scheduled in the first phase where $1 < i < m$. Let $m = 3$. Then $i = 2$. Thus, the second part of the lemma follows. Let $m > 3$. Now, either $i > 2$ or $i < m - 1$. Wlog, let $i > 2$. Thus, $\{l_1, \ldots, l_{i-2}\}$ are not scheduled in the first phase, and l_{i-1} does not prevent the contention of any link in the second phase. Consider a path $\mathcal{P}'$ that consists of links $\{l_1, \ldots, l_{i-2}\}$. Again, since l_2 have not been scheduled in the first phase (since $i > 2$), from the isolation and non-emptyness criteria, at least one link in $\mathcal{P}'$ contends in the second phase. Using arguments similar to those in the first paragraph, we can show that at least one link in $\mathcal{P}'$ is scheduled in the second phase. Thus, the third part of the lemma follows. ■

Note that the last two parts of lemma 1 do not hold if the iterative step of the Sequential Maximal Path Scheduling has only one phase. For example consider a slot t such that $Q_{l_1}(t) > Q_{l_2}(t) > \ldots > Q_{l_m}(t)$. Clearly, at most one link in $\mathcal{P}$, l_1 is scheduled at the end of the first phase in t, irrespective of whether the non-emptyness and the isolation criteria hold. If $m \geq 6$ and the non-emptyness criterion holds, then at least 2 links will be scheduled by Sequential Path Maximal Scheduling (since the scheduling is maximal), but these links need not be selected during the iterative step if the iterative step has one phase, and hence these links may be those with the minimum queue lengths in $\mathcal{P}$.

Lemma 2: Let κ and B be positive integers such that $B \geq 5\kappa + 1$. Consider a path $\mathcal{P} \subseteq \mathcal{G}$. Let $\mathcal{P}$ consist of links $l_1, \ldots, l_m$, where $1 \leq m \leq 5$. Consider an event $\mathcal{A}$ that occurs if and only if there exists a time t such that

1) $Q_{l_i}(t) \geq B - \kappa \ \ \forall\ i \in \{1, \ldots, m\}$ (*lower bound criterion*)
2) $Q_{l_i}(t') \leq B - 1 \ \ \forall\ i \in \{1, \ldots, m\}$ and $\forall\ t' \leq t$ (*upper bound criterion*), and

3) $Q_l(t') < B - 5\kappa \ \forall \ l \in \mathcal{C}_\mathcal{P}, \ \forall \ t' < t$ (*boundary condition*).

Then $\mathbf{P}(\mathcal{A}) \leq 5\chi_{\xi/3}(\frac{\max_{l\in\mathcal{L}}\sigma_l}{4\kappa})^\alpha$.

Proof: Let $\mathcal{A}$ occur. Since $Q_l(0) = 0$ for all $l \in \mathcal{L}$, there exists a slot $t_2 < t$ such that

$$Q_{l_i}(t') \geq B - 5\kappa \ \forall \ \ i \in \{1,\ldots,m\} \text{ and } t' \in [t_2, t], \quad (4)$$
$$\text{and } Q_{l_i}(t_2) = B - 5\kappa \text{ for some } i \in \{1,\ldots,m\}. \quad (5)$$

From the lower bound criteria and (5), $t - t_2 \geq \frac{4\kappa}{\max_{l\in\mathcal{L}}\sigma_l}$.

Let $\mathcal{B}_i$ be the event that $A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2) \geq \xi/3(t-t_2)$. From (2), $\mathbf{P}(\mathcal{B}_i) < \chi_{\xi/3}(\frac{\max_{l\in\mathcal{L}}\sigma_l}{4\kappa})^\alpha$. We will prove that if $\mathcal{A}$ occurs, then $\cup_{i=1}^m \mathcal{B}_i$ occurs. Thus, $\mathbf{P}(\mathcal{A}) \leq \sum_{i=1}^m \mathbf{P}(\mathcal{B}_i)$. The result follows.

From (5) and the lower and upper bound criteria,

$$\begin{aligned}
&\sum_{i=1}^m (Q_{l_i}(t) - Q_{l_i}(t_2)) \\
\geq \ & m(B-\kappa) - (m-1)(B-1) - (B-5\kappa), \\
= \ & 5\kappa - m\kappa + m - 1 \ \geq \ 0 \text{ (since } 1 \leq m \leq 5). \quad (6)
\end{aligned}$$

$$\sum_{i=1}^m (Q_{l_i}(t) - Q_{l_i}(t_2)) = \sum_{i=1}^m (A_{l_i}(t_2,t) - S_{l_i}(t_2,t)), \quad (7)$$

where $S_{l_i}(t_2,t)$ is the number of packets of link i scheduled in interval $[t_2,t)$. Thus, from (6) and (7),

$$\sum_{i=1}^m (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) \geq \sum_{i=1}^m (S_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)). \quad (8)$$

First, let $m \in \{1,2\}$. Now, from the boundary condition, (4) and lemma 1, at least 1 link in $\mathcal{P}$ is scheduled in each slot in $[t_2,t)$. Thus, from (8),

$$\begin{aligned}
\sum_{i=1}^m (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) \ &\geq \ (1 - \sum_{i=1}^m \rho_{l_i})(t-t_2) \\
&\geq \ \xi(t-t_2) \text{(from (3))}. \quad (9)
\end{aligned}$$

Thus, clearly $\mathcal{B}_i$ occurs for some i such that $1 \leq i \leq m$. The result follows.

Now, let $m \in \{4,5\}$. Then, from the boundary condition, (4) and lemma 1, at least 2 links in $\mathcal{P}$ are scheduled in each slot in $[t_2,t)$. Thus, from (8), $\sum_{i=1}^m (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) \geq (2 - \sum_{i=1}^m \rho_{l_i})(t-t_2)$. From (3), $\sum_{i=1}^m \rho_{l_i} \leq 3 \times (2/3 - \xi) = 2 - 3\xi$. Thus, $\sum_{i=1}^m (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) \geq 3\xi(t-t_2)$. Thus, $\mathcal{B}_i$ occurs for some i such that $1 \leq i \leq m$. Thus, the lemma holds for $m \in \{1,2,4,5\}$.

Now, let $m = 3$. Thus, $\mathcal{P}$ consists of l_1, l_2, l_3.

$$\begin{aligned}
&\sum_{i=1}^2 (Q_{l_i}(t) - Q_{l_i}(t_2)) + \sum_{i=2}^3 (Q_{l_i}(t) - Q_{l_i}(t_2)) \\
= \ & \sum_{i\in\{1,3\}} (Q_{l_i}(t) - Q_{l_i}(t_2)) + 2(Q_{l_2}(t) - Q_{l_2}(t_2)). \quad (10)
\end{aligned}$$

Now, from (5) and lower and upper bound criteria, $\forall \ i \in \{1,2,3\}$, $Q_{l_i}(t) - Q_{l_i}(t_2) \geq B - \kappa - (B-1) = 1 - \kappa$, and for some $i \in \{1,2,3\}$, $Q_{l_i}(t) - Q_{l_i}(t_2) \geq B - \kappa - (B - 5\kappa) = 4\kappa$. Thus,

$$\begin{aligned}
&\sum_{i\in\{1,3\}} (Q_{l_i}(t) - Q_{l_i}(t_2)) + 2(Q_{l_2}(t) - Q_{l_2}(t_2)) \\
&\geq 3(1-\kappa) + 4\kappa \ > \ 0 \text{ (since } \kappa \geq 0). \quad (11)
\end{aligned}$$

Now, from (10) and (11),

$$\sum_{i=1}^2 (Q_{l_i}(t) - Q_{l_i}(t_2)) + \sum_{i=2}^3 (Q_{l_i}(t) - Q_{l_i}(t_2)) \geq 0. \quad (12)$$

From the boundary condition, (4) and lemma 1, either l_2 or both l_1 and l_3 are scheduled in each slot in $[t_2,t)$. Thus,

$$S_{l_1}(t_2,t) + 2S_{l_2}(t_2,t) + S_{l_3}(t_2,t) \geq 2(t-t_2). \quad (13)$$

$$\begin{aligned}
&\sum_{i=1}^2 (Q_{l_i}(t) - Q_{l_i}(t_2)) + \sum_{i=2}^3 (Q_{l_i}(t) - Q_{l_i}(t_2)) \\
= \ & \sum_{i=1}^2 (A_{l_i}(t_2,t) - S_{l_i}(t_2,t)) \\
& + \sum_{i=2}^3 (A_{l_i}(t_2,t) - S_{l_i}(t_2,t)) \text{ (from (7))} \\
= \ & \sum_{i=1}^2 A_{l_i}(t_2,t) + \sum_{i=2}^3 A_{l_i}(t_2,t) - S_{l_1}(t_2,t) \\
& -2S_{l_2}(t_2,t) - S_{l_3}(t_2,t) \\
\leq \ & \sum_{i=1}^2 (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) + \sum_{i=2}^3 (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) \\
& - \left(2 - \sum_{i=1}^2 \rho_{l_i} - \sum_{i=1}^2 \rho_{l_i}\right)(t-t_2) \quad (14) \\
\leq \ & \sum_{i=1}^2 (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) + \sum_{i=2}^3 (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) \\
& - (2 - 4/3 + 2\xi)(t-t_2) \text{ (from (3))} \\
\leq \ & \sum_{i=1}^2 (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) \\
& + \sum_{i=2}^3 (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) - 2\xi(t-t_2).
\end{aligned}$$

Note that (14) above follows from (13). Thus, from (12), $\sum_{i=1}^2 (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) + \sum_{i=2}^3 (A_{l_i}(t_2,t) - \rho_{l_i}(t-t_2)) \geq 2\xi(t-t_2)$. Thus, again, $\mathcal{B}_i$ occurs for some i such that $1 \leq i \leq m$. The result follows. ■

Note that lemma 2 does not hold if the iterative step of the Sequential Maximal Path Scheduling has only one phase, as its proof uses lemma 1 which does not hold in this case.

Lemma 3: Consider an integer $B \geq 6$ and a path $\mathcal{P} \subseteq \mathcal{G}$ consisting of links $l_1,\ldots,l_m$ such that $m = \min(6, |\mathcal{L}|)$. Consider an event $\mathcal{A}$ that occurs if and only if there exists a slot t such that

$$\begin{aligned}
Q_{l_i}(t') \ &\leq \ B - 1 \ \forall \ i \in \{j,\ldots,j+m-1\} \text{ and } \forall \ t' \leq t, \\
Q_{l_i}(t) \ &\geq \ 5B/6 \ \forall \ i \in \{j,\ldots,j+m-1\}.
\end{aligned}$$

Then $\mathbf{P}(\mathcal{A}) \leq 6\chi_{\xi/2}(6\max_{l\in\mathcal{L}}\sigma_l/5B)^\alpha$.

Proof:

Consider the last slot t' before t such that $Q_{l_k}(t') = 0$ for some $k \in \{1,\ldots,6\}$. Since $Q_{l_k}(t) \geq 5B/6$, $t' \leq t - 5B/6\max_{l\in\mathcal{L}}\sigma_l$. Let $\mathcal{B}_i$ be the event that $A_{l_i}(t',t) - \rho_{l_i}(t-t') \geq \xi/2(t-t')$. From (2), $\mathbf{P}(\mathcal{B}_i) < \chi_{\xi/2}(\frac{6\max_{l\in\mathcal{L}}\sigma_l}{5B})^\alpha$. We will prove that if $\mathcal{A}$ occurs, then $\cup_{i=1}^m \mathcal{B}_i$ occurs. Thus, $\mathbf{P}(\mathcal{A}) \leq \sum_{i=1}^m \mathbf{P}(\mathcal{B}_i)$. The result follows.

$$\begin{aligned}
&\sum_{i=1}^m \left(Q_{l_i}(t) - Q_{l_i}(t')\right) \\
= \ & \left(Q_{l_k}(t) - Q_{l_k}(t')\right) + \sum_{\substack{1\leq i\leq m \\ i\neq k}} \left(Q_{l_i}(t) - Q_{l_i}(t')\right) \\
\geq \ & 5B/6 + \sum_{\substack{1\leq i\leq m \\ i\neq k}} (5B/6 - B + 1) \ \geq \ m. \quad (15)
\end{aligned}$$

$$\text{Also, } \sum_{i=1}^{m}\left(Q_{l_i}(t)-Q_{l_i}(t')\right)=\sum_{i=1}^{m}A_{l_i}(t',t)-\sum_{i=1}^{m}S_{l_i}(t',t). \quad (16)$$

Thus,

$$\begin{aligned}
&\sum_{i=1}^{m}\left(A_{l_i}(t',t)-\rho_{l_i}(t-t')\right)\\
\geq\ &\sum_{i=1}^{m}S_{l_i}(t',t)-\sum_{i=1}^{m}\rho_{l_i}(t-t')+m \text{ (from (15),(16))}\\
\geq\ &\sum_{i=1}^{m}S_{l_i}(t',t)-(\lceil m/2\rceil)\left((2/3)-\xi\right)(t-t')+m \text{ (from (3)).}
\end{aligned}$$

Next, $Q_{l_i}(t_1)>0 \ \ \forall\ i\in\{1,\ldots,m\}, \ \ \forall\ t_1\in(t',t]$. Thus, since the set of links scheduled at each slot constitutes a maximal scheduling among those that have positive queue lengths in the slot, (a) at least two links in $\mathcal{P}$ are scheduled in every slot in $(t',t]$, if $m\in\{4,5,6\}$ and (b) one link in $\mathcal{P}$ is scheduled in every slot in (t,t'). (For $m<6$, the above follows since $m=|\mathcal{L}|$, and hence $\mathcal{P}=\mathcal{G}$). Thus, if $m\geq 4$, $\sum_{i=1}^{m}S_{l_i}(t',t)\geq 2(t-t')-2$. Thus, if $m\geq 4$,

$$\sum_{i=1}^{m}\left(A_{l_i}(t',t)-\rho_{l_i}(t-t')\right)\geq (m/2)\xi(t-t').$$

Thus, for some $i\in\{1,\ldots,m\}$, $\mathcal{B}_i$ occurs. The result follows for $m\in\{4,5,6\}$. Next, let $m\in\{1,2\}$. Then, $\sum_{i=1}^{m}S_{l_i}(t',t)\geq(t-t')-2$. Thus,

$$\sum_{i=1}^{m}\left(A_{l_i}(t',t)-\rho_{l_i}(t-t')\right)\geq \xi(t-t')-1.$$

Thus, for some $i\in\{1,\ldots,m\}$, $\mathcal{B}_i$ occurs. The result follows for $m\in\{1,2\}$.

Now, let $m=3$. Similar to the proof for (15), we can prove that,

$$\begin{aligned}
&\sum_{i\in\{1,3\}}\left(Q_{l_i}(t)-Q_{l_i}(t')\right)+2\left(Q_{l_2}(t)-Q_{l_2}(t')\right)\\
\geq\ &5B/6+3(5B/6-B+1)\geq 3. \quad (17)\\
&\text{Also, } \sum_{i\in\{1,3\}}\left(Q_{l_i}(t)-Q_{l_i}(t')\right)+2\left(Q_{l_2}(t)-Q_{l_2}(t')\right)\\
=\ &\sum_{i\in\{1,3\}}A_{l_i}(t',t)-\sum_{i\in\{1,3\}}S_{l_i}(t',t)\\
&+2A_{l_2}(t',t)-2S_{l_2}(t',t). \quad (18)\\
&\text{Thus, } \sum_{i\in\{1,3\}}\left(A_{l_i}(t',t)-\rho_{l_i}(t-t')\right)\\
&+2\left(A_{l_2}(t',t)-\rho_{l_2}(t-t')\right)\\
\geq\ &\sum_{i\in\{1,3\}}S_{l_i}(t',t)+2S_{l_2}(t',t)-\sum_{i\in\{1,3\}}\rho_{l_i}(t-t')\\
&-2\rho_{l_2}(t-t')+3 \text{ (from (17),(18))}\\
\geq\ &\sum_{i\in\{1,3\}}S_{l_i}(t',t)+2S_{l_2}(t',t)\\
&-2\left((2/3)-\xi\right)(t-t')+3 \text{ (from (3)).}
\end{aligned}$$

Again, using similar arguments as before, either l_2 or both l_1 and l_3 are scheduled in every slot in $(t',t]$. Thus, $\sum_{i\in\{1,3\}}S_{l_i}(t',t)+2S_{l_2}(t',t)\geq 2(t-t')-2$. Thus,

$$\begin{aligned}
&\sum_{i\in\{1,3\}}\left(A_{l_i}(t',t)-\rho_{l_i}(t-t')\right)+2\left(A_{l_2}(t',t)-\rho_{l_2}(t-t')\right)\\
\geq\ &2\xi(t-t')-1.
\end{aligned}$$

Thus, for some $i\in\{1,\ldots,3\}$, $\mathcal{B}_i$ occurs. The result follows. ■

2) Proof for Theorem 1: We just prove the first part of the theorem, as the second part is immediate from the first. The result trivially holds for $B<\tau$. Let $B\geq\tau$. Consider the event $\mathcal{A}(B,t)$ that occurs if and only if $\max_{l\in\mathcal{G}}Q_l(t)\geq B$. Let $\mathcal{A}(B,t)$ occur. Then there exists a slot $t_0\leq t$ such that $Q_l(t')\leq B-1$ for all $l\in\mathcal{L}$ and $t'\leq t_0$ and $Q_l(t_0+1)\geq B$ for some $l\in\mathcal{L}$; let l' be one such l. Then, $Q_{l'}(t_0)\geq B-\max_{l\in\mathcal{L}}\sigma_l$. For $1\leq c\leq 6$, $1\leq d\leq c$, consider paths $\mathcal{P}_{c,d}\subseteq\mathcal{G}$ consisting of c links with the dth link being l', provided such a path exists. For example, such a path does not exist if l' is the last link of path $\mathcal{G}$, and $d<c$. For $1\leq c\leq 5, 1\leq d\leq c$, event $\mathcal{C}_{c,d}$ is said to occur if the event $\mathcal{A}$ described in lemma 2 occurs with $\mathcal{P}=\mathcal{P}_{c,d}$, $\kappa=4^5 5^{c-1}\lfloor\frac{B}{6\times 4^5\times 5^5}\rfloor$. Event $\mathcal{C}_{\min(6,|\mathcal{L}),d}$ is said to occur if $\mathcal{P}_{\min(6,|\mathcal{L}),d}$ exists and the event $\mathcal{A}$ described in lemma 3 occurs with $\mathcal{P}=\mathcal{P}_{\min(6,|\mathcal{L}),d}$. Clearly, when $\mathcal{A}(B,t)$ occurs, $\mathcal{C}_{c,d}$ occurs for some c,d, $1\leq c\leq 6, 1\leq d\leq c$. Thus, $\mathbf{P}\left(\mathcal{A}(B,t)\right)$ is upper bounded by the sum of the probabilities of the above events. Thus, the result follows from the upper bounds of the probabilities of these events provided in lemmas 2 and 3.

IV. Sequential Maximal Scheduling in Trees

We now describe how a throughput guarantee of $2/3$ can be attained through distributed scheduling in trees. We will first show that every tree can be decomposed into a collection of link disjoint paths that constitute a tree of paths of depth at most $O(\log n)$ (Section IV-B). We refer to this new tree as a *path tree*. In our scheduling algorithm, every path in this path tree executes a queue length based sequential maximal scheduling policy after waiting for a time interval in which its parent path in the path tree finishes its scheduling (with high probability) (Section IV-C). The sequential maximal scheduling policy that can be used in paths in the tree (*Sequential Maximal Tree Scheduling*) however needs to be slightly different from that when the entire graph is a path. This is because irrespective of its queue length, the first link in a path $\mathcal{H}$ can not be scheduled in a slot in which the last link of its parent path is scheduled $-$ such slots are referred to as *constrained slots* for $\mathcal{H}$. Nevertheless, we prove that the combination attains a $2/3$ throughput guarantee as before (Section IV-E).

A. Preliminaries

We now assume that $\mathcal{G}$ is a tree with maximum degree $\Delta\geq 1$.

Next we introduce some terminology and definitions that will be used in presenting our algorithm and its analysis. Let $\mathcal{H}_i$, $i=1,\ldots,k$, denote subsets of $\mathcal{L}$. If $\mathcal{H}_i=\{l_{1,i},\ldots,l_{m,i}\}$ is a path, then $l_{1,i}$ and $l_{m,i}$ are its *terminal* links. If there exist a link $l_1\in\mathcal{H}_i$ and a link $l_2\in\mathcal{H}_j$ such that l_1 and l_2 are adjacent, then $\mathcal{H}_i$ and $\mathcal{H}_j$ are *adjacent* and l_1 (l_2) is adjacent to $\mathcal{H}_j$ ($\mathcal{H}_i$); if l_1 is a terminal link in $\mathcal{H}_i$, then $\mathcal{H}_i$ is *terminal-adjacent* of $\mathcal{H}_j$.

The following property, which we refer to as the *tree-property*, holds since $\mathcal{G}$ is a tree. Let elements in $\{\mathcal{H}_1,\ldots,\mathcal{H}_k\}$ be pair-wise disjoint and pair-wise adjacent, and $\mathcal{B}=\{l: l\in\mathcal{H}_i \text{ for some } i, \text{ and } \mathcal{N}_l\cap\mathcal{H}_j\neq\phi \text{ for some } i\neq j\}$. Then all links in $\mathcal{B}$ intersect at one node in $\mathcal{G}$. Also, at most two links in any $\mathcal{H}_i$ can be adjacent to $\mathcal{H}_j$ where $j\neq i$.

Let $\{\mathcal{H}_k\}$ constitute a partition of $\mathcal{L}$ such that each set $\mathcal{H}_u$ in the partition is a path in $\mathcal{G}$, and corresponds to a node u in a tree $\mathcal{G}^P$ (with a designated root node) that satisfies the following properties. Consider two nodes u and v in $\mathcal{G}^P$ and the corresponding sets $\mathcal{H}_u$ and $\mathcal{H}_v$ in the partition.

P.1 If u is a parent (child) of v, (a) $\mathcal{H}_v$ ($\mathcal{H}_u$) is terminal-adjacent of $\mathcal{H}_u$ ($\mathcal{H}_v$) and (b) only one link in $\mathcal{H}_v$ ($\mathcal{H}_u$) is adjacent to $\mathcal{H}_u$ ($\mathcal{H}_v$).

P.2 If u and v are siblings, then either both $\mathcal{H}_u$ and $\mathcal{H}_v$ are terminal-adjacent of each other, or they are not adjacent.

P.3 If u is not a parent, child, sibling of v, then $\mathcal{H}_u$ and $\mathcal{H}_v$ are not adjacent.

To illustrate the above definitions and properties, consider the example tree network consisting of 11 links as shown in Figure 3(a). The tree has been partitioned into 6 (link-) disjoint paths, {H0, ..., H5}, where H0 = {1,2,3}, H1 = {4}, H2 = {5}, H3 = {6, 7}, H4 = {8, 10, 11}, and H5 = {9}. For path H0, 1 and 3 are the terminal links, while for path H3, both 6 and 7 are terminal links. H0 and H3 are not only adjacent, but also terminal-adjacent of each other; however, H0 is not adjacent to H5. To illustrate the tree property, consider the paths {H0, H2, H3 } which are pair-wise disjoint and pair-wise adjacent. In this case $\mathcal{B}$ = {1, 2, 5, 6}. Clearly, all links in $\mathcal{B}$ intersect at a single node, v_1. Also note that two links in H0 are adjacent to H2, H3, while only a single link in H2 (H3) is adjacent to H1, H3 (H1, H2).

In this example, it can be verified the path tree $\mathcal{G}^P$ shown in Figure 3(b) satisfies the properties P.1-P.3 stated above. For instance, since H3 is a child of H0 in $\mathcal{G}^P$, consistent with property P.1, in graph $\mathcal{G}$, H3 is terminal adjacent of H0, and only one link of H3 is adjacent to H0. To illustrate property P.2, consider siblings H1, H2, H3, H4 in graph $\mathcal{G}^P$, and note that H2, H3 are terminal-adjacent of each other, while H1 and H4 are not adjacent to any of the other sibling paths. Property P.3 ca be illustrate by considering H0 and H5.

Our algorithm requires a decomposition of the link set $\mathcal{L}$ into a tree $\mathcal{G}^P$ of paths that satisfy properties P.1-P.3 and have a depth of $O(\log n)$. We show next that this can always be done, and present an algorithm that achieves this in polynomial time.

B. Path Tree Construction

We first introduce some new terminology. The *size* of a node in $\mathcal{G}$ is the number of nodes in the subtree rooted at the node. The *root-component* of $\mathcal{G}$ is $\mathcal{G}$ itself.

We now describe the construction of the paths corresponding to nodes in $\mathcal{G}^P$. The path corresponding to the root of $\mathcal{G}^P$, which we denote as the *root-path* of $\mathcal{G}$, is the path $u_0, \ldots, u_k$ where u_0 is the root of $\mathcal{G}$, u_i is the node with the maximum size among the children of u_{i-1} in $\mathcal{G}$, and u_k is a leaf of $\mathcal{G}$. Once the root-path has been identified, all nodes in the root-path and the links originating from these nodes are removed from $\mathcal{G}$. Each component in the residual graph is referred to as the *child-component* of the *root-component* and the *root-component* is their *parent-component.* Note that a child-component may consist of a single node or may have multiple nodes and links. The root-path in each child-component with a single node is considered to be the node itself (i.e., this path is empty in the sense that it does not have any links). Once the root-path is identified in such a child-component, the child-component is removed from the graph. The root-path in each child-component with multiple nodes is determined similar to the root-path in $\mathcal{G}$, and this in turn leads to child-components of each child-component. The process terminates when the residual graph has no nodes.

We now describe how the paths obtained as above can be organized to constitute the path tree $\mathcal{G}^P$. The root-path $\mathcal{H}$ for the root-component (i.e. $\mathcal{G}$) corresponds to the root of $\mathcal{G}^P$. Subsequently, we consider the root-paths of the child-components of $\mathcal{G}$. Let $\mathcal{H}'$ be one such root-path. There exists a link l between an end-node of $\mathcal{H}$ and $\mathcal{H}'$. Let $\mathcal{H}'' = \mathcal{H}' \cup \{l\}$. Then in $\mathcal{G}^P$, $\mathcal{H}''$ corresponds to a child of the node corresponding to $\mathcal{H}$. Similarly, other children of the root of $\mathcal{G}^P$ are identified by considering root-paths of other child-components of $\mathcal{G}$. Subsequently, the paths corresponding to the nodes in the next level of $\mathcal{G}^P$ are identified similarly. Note that at the end of this procedure each identified path has at least one link; thus henceforth we no longer consider empty paths as in the above paragraph.

From the construction of $\mathcal{G}^P$, it is easy to verify that it satisfies P.1-P.3.

To illustrate the path tree construction for the graph shown in Figure 3(a), note that starting from the root node v_0, the first path identified (the root path of $\mathcal{G}$) is H0. (Note that v_1 (size = 7) is preferred over v_8 (size = 4), and v_2 (size = 3) is preferred over v_3 (size = 2).) When this root path is removed, the graph $\mathcal{G}$ decomposes into 4 child components - the subgraphs formed by the node sets $\{v_4\}, \{v_5\}, \{v_6, v_7\}$ and $\{v_8, v_9, v_{10}, v_{11}\}$. The first two components are single node sets, and result in the two single-link paths H1 and H2. The third is a two-link single-path component which results in path H3. The root path in the last child component is {10, 11} (v_{10} (size = 2) is preferred over v_9 (size = 1)), which when appended with link 8 (which connects this root path with H0, the root path of the parent component), results in the path H4. Thus H1, H2, H3 and H4 become children of H0 in the path tree $\mathcal{G}^P$. Removing the root path from the last child component leaves the single node component $\{v_9\}$, which results in path H5, a child of H4 in the path tree $\mathcal{G}^P$, as show in Figure 3(b). Thus, running our path tree construction algorithm on $\mathcal{G}$ shown in Figure 3(a) results in Figure 3(b).

Lemma 4: The depth of $\mathcal{G}^P$ is at most $\log n$.

Proof: Each node u in $\mathcal{G}^P$ corresponds to a path in $\mathcal{G}$, and each such path is the root-path of some component in $\mathcal{G}$, say $\mathcal{G}_u$; let the *counterpart* of u in $\mathcal{G}^P$ be the root-node of $\mathcal{G}_u$. Let the *weight* of a node in $\mathcal{G}^P$ be the size of its counterpart in $\mathcal{G}$. We will show that for any two nodes u, v in $\mathcal{G}^P$ such that v is a child of u in $\mathcal{G}^P$ the weight of v is less than half that of u. Thus, if the depth of $\mathcal{G}^P$ is $d(\mathcal{G}^P)$, then, $2^{d(\mathcal{G}^P)}$ times the weight of a leaf node in $\mathcal{G}^P$ is upper-bounded by the weight of the root node in $\mathcal{G}^P$. Note that the weight of the root and leaf nodes in $\mathcal{G}^P$ are n and 1 respectively. Thus, $d(\mathcal{G}^P) \leq \log(n)$.

Consider nodes u and v in $\mathcal{G}^P$ such that v is a child of u in $\mathcal{G}^P$. We now show that the weight of v is less than half that of u in $\mathcal{G}^P$. Let $\mathcal{H}_u$ be the path in $\mathcal{G}$ that corresponds to u. Let u_1 and v_1 be counterparts of u and v respectively in $\mathcal{G}$. Then (a) v_1 is in the sub-tree of $\mathcal{G}$ rooted at u_1, (b) v_1 is not in $\mathcal{H}_u$, (c) v_1 is the child in $\mathcal{G}$ of a node w in $\mathcal{H}_u$ in $\mathcal{G}$, (d) w is in the sub-tree of $\mathcal{G}$ rooted at u_1 and w has a child w_1 in $\mathcal{H}_u$. Clearly, w_1 and v_1 are siblings and the size of v_1 can not exceed the size of w_1 (otherwise $\mathcal{H}_u$ would have traversed v_1 instead of w_1). Since w_1 and v_1 are children of w, the size of w exceeds the sum of sizes of v_1 and w_1. Hence, the size of v_1 is less than half the size of w and hence less than half the size of u_1 since w is in the sub-tree in $\mathcal{G}$ rooted at u_1. Thus, the weight of v is less than half that of u in $\mathcal{G}^P$. ■

C. Scheduling Algorithm

Each path represented by the vertices in the path tree graph $\mathcal{G}^P$, the output of the path tree construction procedure described above, executes the Sequential Maximal Path Scheduling algorithm after waiting for a time interval that depends on the position of the vertex corresponding to the path in $\mathcal{G}^P$. We provide details of the algorithm below.

Let paths $\{\mathcal{H}_k\}$ be the output of the path tree construction algorithm. If u is the parent of v in $\mathcal{G}^P$ then the link in $\mathcal{H}_v$ that is adjacent to $\mathcal{H}_u$ is referred to as the *first* link in $\mathcal{H}_v$; note that this is a terminal link in $\mathcal{H}_v$. For example, in Figure 3, link 6 is the first link of path H3. Due to the tree properties P.1-P.3, there exists a partition on the children of each u in $\mathcal{G}^P$ such that $\mathcal{H}_u$ and the corresponding paths in each set in the partition intersect at a common

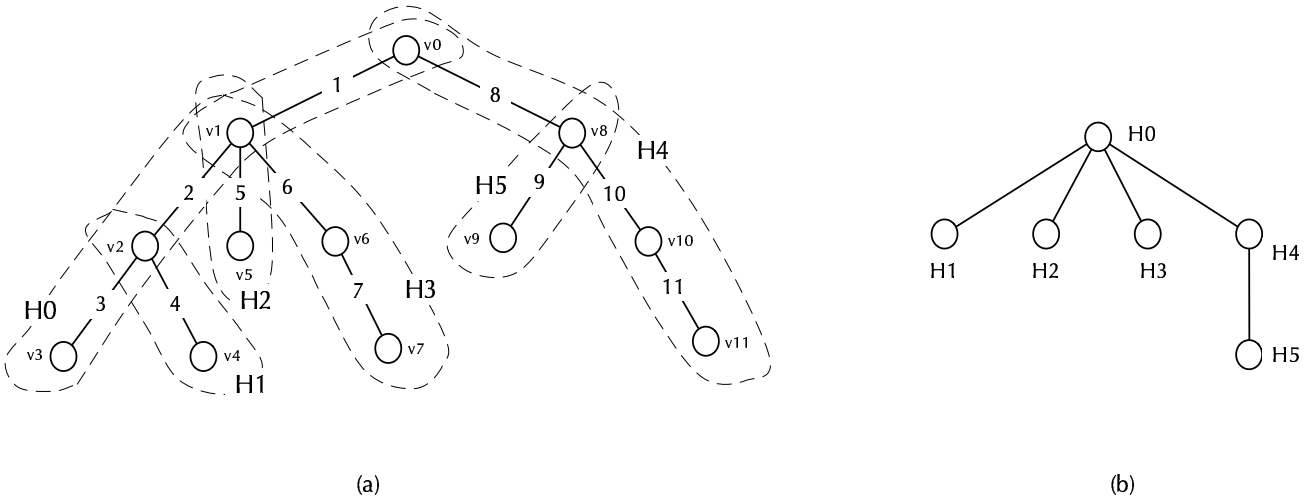

Fig. 3. Path Tree Construction: (a) The original tree $\mathcal{G}$; (b) The path tree $\mathcal{G}^P$.

node in $\mathcal{G}$, and the corresponding paths in different partitions are not adjacent. For the children of H0 in $\mathcal{G}^P$, {H1}, {H2, H3}, {H4}, represents such a partition. Given the degree bound, each partition consists of at most Δ nodes in $\mathcal{G}$, and all these nodes are siblings. The nodes in a partition are numbered in some chosen order. If two siblings v, w are in the same partition, and v has a higher number than w, then v (w) is an *older* (*younger*) sibling of w (v). Thus, a node in $\mathcal{G}^P$ can have at most $\Delta - 1$ older siblings.

Without loss of generality, assume that the $\mathcal{H}_i$s have been numbered in the sequence in which the corresponding nodes will be visited in a breadth first traversal of $\mathcal{G}^P$ starting from the root of $\mathcal{G}^P$; the breadth first traversal visits an older sibling before a younger sibling. Let p_i be the level (i.e., the distance from the root) of node i in $\mathcal{G}^P$ and r_i be the number of its older siblings. Let $\hat{p}$ be the maximum level of any node in $\mathcal{G}^P$. From Lemma 4, $\hat{p}$ is $O(\log n)$. In $\mathcal{G}^P$, p_i for H1, H2, H3 and H4 is 1, while p_i for H5 is 2. Moreover, if H2 is considered older than H3 in the partition {H2, H3}, then r_i for H1, H2, H3 and H4 are 0, 0, 1 and 0, respectively.

Recall that maximal scheduling is implemented using a distributed randomized algorithm like the one proposed in [12]. The algorithm operates in rounds, where each round requires communication by nodes with their neighbors in the same path of the path tree. [12], [14](chap. 8). Let $\gamma > 0$ be the probability that the second link in a path with only three links does not select itself at the end of its first round of the distributed maximal scheduling algorithm. Given that a link is un-decided at the beginning of a round in its maximal scheduling, it is un-decided with a probability of at most γ at the end of the round. For the algorithm in [12], it can be easily shown that $\gamma < \frac{13}{27}$.

At the beginning of every slot, all links that do not have any packets to transmit set their status to *un-scheduled*. All other links set their status to *un-decided* initially. As the scheduling algorithm progresses, these un-decided links change their status to *scheduled* or *un-scheduled*. Links in $\mathcal{H}_i$ start executing their scheduling phase after the paths that correspond to $\mathcal{H}_i$'s predecessors and their older siblings in $\mathcal{G}^P$, and the older siblings of $\mathcal{H}_i$ itself, have completed their scheduling (with high probability). Note that since a node in $\mathcal{G}^P$ can have at most Δ older siblings, the number of $\mathcal{H}_i$'s predecessors and their older siblings, plus the older siblings of $\mathcal{H}_i$ itself, is upper bounded by $p_i\Delta + r_i$. In our algorithm, links in $\mathcal{H}_i$ start executing the Sequential Maximal Tree Scheduling routine (Figure 4) after $(p_i\Delta + r_i)\left(T_1 + T_2\lceil \ln(36\Delta) / (-\ln(\gamma))\rceil\right)$ time, where T_1 represents an upper bound on the time required to execute the initial and iterative steps of the Sequential Maximal Tree Scheduling algorithm for a path, and T_2 represents an upper bound on the time required to complete *a single round* of the distributed maximal scheduling algorithm [12], [14](chap. 8). Note that this implies that a path starts its scheduling after its predecessors and their older siblings, and the path's own older siblings, have completed at least $\lceil \ln(36\Delta) / (-\ln(\gamma))\rceil$ rounds of the maximal scheduling algorithm. Thus when a path starts its scheduling, its predecessor paths and their older siblings, and its own older siblings, may not have completed their scheduling process (recall that maximal scheduling for a path takes $O(\log n)$ expected time). However, the constant $\lceil \ln(36\Delta) / (-\ln(\gamma))\rceil$ is chosen such that the probability of maximal scheduling completing within those many rounds is high enough for our stability result to hold.

We now point out the similarities and differences between Sequential Maximal Tree Scheduling and Sequential Maximal Path Scheduling. Consider an arbitrary path $\mathcal{H} \in \{\mathcal{H}_k\}$, where $\mathcal{H} = l_{\mathcal{H},1}, \ldots, l_{\mathcal{H},m}$, and $l_{\mathcal{H},1}$ is the first link in $\mathcal{H}$. A slot is a *constrained slot* for $\mathcal{H} \in \{\mathcal{H}_k\}$ if the first link of $\mathcal{H}$ sets its status to un-scheduled in the sequential constraint step, and is an *un-constrained slot* otherwise. In an un-constrained slot, since the start of its scheduling phase, the two scheduling procedures are identical. The above holds in a constrained slot as well except for $l_{\mathcal{H},1}$ which becomes un-scheduled in Sequential Maximal Tree Scheduling irrespective of its queue length. Note that in an un-constrained slot, the scheduling for $\mathcal{H}$ is oblivious of any link not in $\mathcal{H}$, and in a constrained slot, the scheduling for $\mathcal{H} \setminus \{l_{\mathcal{H},1}\}$ is oblivious of any link not in $\mathcal{H} \setminus \{l_{\mathcal{H},1}\}$. Finally, unlike that for paths, the overall scheduling for trees need not be maximal. This is because in a slot that is constrained for $\mathcal{H}$, it may turn out that the links that are (a) in the parent and older siblings of $\mathcal{H}$, (b) adjacent to the first link in $\mathcal{H}$, and (c) were undecided at the time $\mathcal{H}$ started its scheduling phase, may eventually not be scheduled in the slot. Nevertheless, in the next section, we prove that the $2/3$ throughput guarantee still holds for trees. This is attained by (a) exploiting the fact that the constrained slots for each path occur only at a rate which is upper-bounded by one minus the packet arrival rate in the first link of $\mathcal{H}$, and (b) using an additional phase in the the iterative step of Sequential Maximal Tree Scheduling. Thus, the iterative step now uses 3 phases, whereas the iterative step of Sequential Maximal Path Scheduling only uses 2 phases.

SEQUENTIAL MAXIMAL TREE SCHEDULING ($\mathcal{H}_i$)

INITIAL STEP: The first link, say $l_{\mathcal{H}_i 1}$, in $\mathcal{H}_i$, sets its status to "un-scheduled" if at least one link in $\mathcal{N}_{l_{\mathcal{H}_i 1}} \cap \mathcal{H}_j$, where j is a parent or an older sibling of i in $\mathcal{G}^P$, has been scheduled or is un-decided (*sequential constraint*). Each link sets its status to "un-scheduled" if it does not have a packet to transmit. If a link does not set its status to "un-scheduled," it sets its status to "un-decided."

ITERATIVE STEP: For $k = 1$ to $k = 3$, execute Phase k, as given below:
Phase k : A link in $\mathcal{H}_i$ contends if and only if (a) it is un-decided, (b) its adjacent links in $\mathcal{H}_i$ are un-scheduled or un-decided, (c) its queue length is not less than that of its adjacent links in $\mathcal{H}_i$ that satisfy conditions (a) and (b). A contending link sets its status to "scheduled" if its adjacent links do not contend or have higher id than it; links that are adjacent to scheduled links set their status to "un-scheduled".

TERMINAL STEP: Compute a maximal schedule among the links in $\mathcal{H}_i$ that are un-decided and whose adjacent links in $\mathcal{H}_i$ are un-scheduled or un-decided. Set the status of the links selected in the maximal schedule to "scheduled", and the status of the links that are adjacent to scheduled links as "un-scheduled".

Fig. 4. Sequential Maximal Tree Scheduling Algorithm for $\mathcal{H}_i$

Finally, we evaluate the time required for the schedule computation. Firstly, note that the first link $l_{\mathcal{H}_i 1}$ in $\mathcal{H}_j$ shares a node with links in $\mathcal{N}_{l_{\mathcal{H}_i 1}} \cap \mathcal{H}_j$ (where j is a parent or an older sibling of i in $\mathcal{G}^P$). Therefore, assuming that each end-node of a link keeps track of its scheduling status, the initial step in the Sequential Maximal Tree Scheduling algorithm takes constant time. Since the iterative step requires nodes to exchange a constant number of messages with their neighbors in the corresponding path, this implies that T_1, the maximum time required to execute the initial and iterative steps of the Sequential Maximal Tree Scheduling algorithm is a constant independent of Δ and n. Furthermore, since a single round of the maximal schedule computation (terminal step of the Sequential Maximal Tree Scheduling algorithm) only requires a constant number of message exchanges by nodes with their neighbors in the corresponding path, T_2 is also a constant independent of Δ and n. The path that starts its scheduling process last, starts after waiting for $O\left((\max_i p_i \Delta + r_i)\log \Delta\right)$ time, since T_1 and T_2 are constants independent of Δ and n. Once started, the scheduling process for a path takes $O(\log n)$ expected time to complete. Thus, since $r_i \leq \Delta - 1$, and $p_i \leq \hat{p}$ which is $O(\log n)$, the scheduling for the entire tree can be computed in $O(\Delta \log \Delta \log n + \Delta \log \Delta + \log n)$, or $O(\Delta \log \Delta \log n)$, expected time.

D. Discussion

The Sequential Maximal Tree Scheduling Algorithm is fully distributed, as long as we implement the maximal scheduling algorithm on each path in a distributed manner (using the algorithm in [12], for example).

Note that in the path construction algorithm, the root path in any component can be constructed in $O(n)$ communication rounds, where each communication round requires communication by nodes with their neighbors in the given tree network. Therefore, utilizing the fact that root path in all child components of a root component can be constructed in parallel, the entire path construction procedure takes $O(n \log n)$ communication rounds, or $O(\Delta n \log n)$ time. The path tree construction algorithm should be viewed as a "pre-processing" step, and needs to be re-run only when the network topology changes. Therefore, the complexity of the path tree construction does not contribute to the per-slot complexity of the scheduling algorithm.

As mentioned earlier, in our scheduling algorithm, when a path starts its scheduling process, it is possible (although with low probability) that its predecessor and older sibling paths have not completed their scheduling processes yet. However, note that the control message exchanges required during the scheduling process of any path does not interfere with that of its predecessor or older sibling paths, assuming primary interference constraints on control message exchanges. For example, in Figure 3, consider the message exchanges on path H4 after it begins its scheduling process (but before path H5 begins its scheduling). At this time, if link 1 (which belongs to the predecessor path H0) has already decided its scheduling status, then there is no message exchange across link 1, and therefore no interference in the message exchanges on the links in path H4. However, if link 1 is still undecided, then link 8 sets its status to un-scheduled and does not subsequently participate in the scheduling process; control messages are then exchanged only on links 10 and 11, which do not interfere with message exchanges on the link 1 or any other link on the predecessor or older sibling paths. When child path H5 (which consists of only link 9) starts its scheduling process, link 9 will schedule itself only if links 8 and 10 have already set their status to un-scheduled; therefore, there is no interference between control message exchanges on path H4 with those on path H5, even after H5 has started its scheduling process. This holds true in general, due to the fact that at any point in time during the scheduling process on any path, the set of undecided links in the path is node disjoint from all undecided links in other paths that have already begun their schedule computation process.

Finally, note that the framework we proposed involves decomposition of trees into paths and scheduling links in each path using a policy that attains a provable throughput guarantee ($2/3$) for paths. It is interesting to observe that this decomposition based approach retains the same throughput guarantee for trees as compared to that for paths. In general, if the throughput guarantee for path graphs can be improved further while using $O(\log n)$ time, then we can use this framework to obtain the same guarantees for trees while still requiring an overall computation time of $O(\Delta \log \Delta \log n)$.

E. Proof of the $2/3$ throughput guarantee for a tree

We now state the main result of this section, Theorem 2, which proves that Sequential Maximal Tree Scheduling policy attains a $2/3$ throughput guarantee when $\mathcal{G}$ is a tree.

Theorem 2: Let $\mathcal{G}$ be a tree and (3) hold.

1) For each $t > 0$, $l \in \mathcal{L}$, $\mathbf{P}\{Q_l(t) \geq B\} \leq \tau_{\hat{p},\Delta-1} B^{-\alpha}$, where $\tau_{\hat{p},\Delta-1}$ is obtained through the following recursions.

$$\gamma_{0,y} = 0 \;\forall\; 0 \leq y \leq \Delta - 1, \tag{19}$$

$$\begin{aligned}\tau_{x,y} &= \left(\max(1, 151\chi_{\min(\xi/3,1/6)} + 11\gamma_{x,y})\right) \\ &\quad \times (72 \times 4^5 \times 5^5 \max_{l\in\mathcal{L}} \sigma_l)^{\alpha}, \\ &\quad 0 \leq x \leq \hat{p},\ 0 \leq y \leq \Delta - 1,\end{aligned} \tag{20}$$

$$\begin{aligned}\gamma_{x,y} &= \Delta\left(\nu(\Delta,\alpha) + \chi_{(1/18\Delta)}\right) \\ &\quad + 18^{\alpha}\Delta^{\alpha+1} \times \max(\tau_{x-1,\Delta-1}, \max_{0\leq z\leq y-1} \tau_{x,z}), \\ &\quad 0 \leq x \leq \hat{p}, 0 \leq y \leq \Delta - 1.\end{aligned} \tag{21}$$

2) For each $t \geq 0$, $\mathbf{E}\left(Q_l(t)\right) \leq \tau_{\hat{p},\Delta-1} \sum_{i=1}^{\infty} i^{-\alpha}$.

Note that Theorem 2 is similar to Theorem 1; only the constants in the expressions for the probabilities $\mathbf{P}\left\{Q_l(t) \geq B\right\}$ and the expected queue lengths differ. We now describe the structure of the proof for Theorem 2 and point out the similarities and differences with the proof for Theorem 1.

Similar to the proof for the special case in which $\mathcal{G}$ is a path, we first prove that a link can not be congested in isolation. This proof has two major steps. Consider a path $\mathcal{H}$ in $\mathcal{G}$. The first step is to show that if links in a segment $\mathcal{P}$ of $\mathcal{H}$ of length 5 or less have high queue lengths and the segment does not include the first link of $\mathcal{H}$, then with a high probability, at least one link in $\mathcal{H}$ that is not in $\mathcal{P}$ but is adjacent to a link in $\mathcal{P}$ has high queue length as well (lemma 6, Section IV-E.1). This result is similar to lemma 2 proved earlier for a path. We next prove that if $\mathcal{H}$ is not constrained very often, and the first link in $\mathcal{H}$ has a high queue length, then with a high probability the second link has a high queue length as well; this holds for the second-third, third-fourth, fourth-fifth and fifth-sixth pairs as well (lemma 8, Section IV-E.1). This result holds only when the iterative step of the Sequential Tree Maximal Scheduling has three (or more) phases. The above results together imply that if a path is not constrained very often and a link in the path has a high queue length, then with a high probability all links in a segment of length at least 6 are congested.

We next prove that if a path is not constrained very often, the probability that all links in a segment of a path consisting of 6 links has high queue lengths is small (lemma 9, Section IV-E.1). This result is similar to lemma 3, but the proofs differ somewhat since the scheduling for a tree is not always maximal. The proof for lemma 9 again relies on the fact that the iterative step of the Sequential Tree Maximal Scheduling has three phases.

We next prove that a path is not constrained very often if the probability that the queue lengths in the links in its parent and older siblings is low (lemma 10, Section IV-E.1).

Our main result, that the queue length in a link becomes large only with a small probability (Theorem 2), is now obtained using the above results and an induction argument. Note that the root path in $\mathcal{G}$ is never constrained. Thus, using lemmas 6, 8, 9, and arguments similar to the proof for Theorem 1, the result follows for the root path. It therefore follows from lemma 10 that the eldest child of the root path is not constrained very often. Thus the result follows for this as well, and hence follows for the children and the younger siblings of this eldest path, and subsequently for all other paths in $\mathcal{G}$.

We state and prove the supporting lemmas, lemmas 5 to 10 in Section IV-E.1, and using these prove Theorem 2 in Section IV-E.2.

1) Supporting lemmas: We present a series of lemmas, lemmas 5 to 10, for an arbitrary path $\mathcal{H}$ in $\{\mathcal{H}_k\}$, where $\mathcal{H} = l_{\mathcal{H},1}, \ldots, l_{\mathcal{H},m}$, and $l_{\mathcal{H},1}$ is the first link in $\mathcal{H}$. Lemmas 6, 8, 9, 10 are the main lemmas which will be used in proving Theorem 2. Lemmas 5 and 7 provide intermediate results that are only used in proving the main lemmas: lemma 5 is used in proving lemma 6, and lemma 7 is used in proving lemmas 8 and 9.

We first introduce some terminology required in the proofs. Let $\Theta_{\mathcal{H}}(t_1, t_2)$ be the number of un-constrained slots in $[t_1, t_2)$ for $\mathcal{H} \in \{\mathcal{H}_k\}$. Then, $\mathcal{H}$ is said to satisfy the *constraint-lower-bound* if there exists a constant $\gamma_{\mathcal{H}}$ such that $\forall\ 0 < t_6 < t_7$,

$$\mathbf{P}\left\{\Theta_{\mathcal{H}}(t_6, t_7) \leq (\rho_{l_{\mathcal{H},1}} + 1/6)(t_7 - t_6)\right\} \leq \frac{\gamma_{\mathcal{H}}}{(t_7 - t_6)^{\alpha}}.$$

The constraint-lower-bound states that with a high probability the unconstrained slots in each path occur more frequently than the arrivals in the first link of the path. In Theorem 2, using induction, we prove that every path $\mathcal{H}$ satisfies the constraint-lower-bound, and subsequently prove the throughput guarantee using lemmas 6, 8, 9 - the last two of these lemmas hold only when the constraint-lower-bound holds.

Lemma 5: Consider a path $\mathcal{P} \subseteq \mathcal{H}$ where $\mathcal{H}$ is a path in $\{\mathcal{H}_k\}$ and an arbitrary slot t. Let either $\mathcal{P} \subseteq \mathcal{H} \setminus \{l_{\mathcal{H},1}\}$ or t be an un-constrained slot. Let $\mathcal{P}$ consist of links $l_1, \ldots, l_m$, and satisfy the following properties at t.

1) $Q_{l_i}(t) > 0 \ \ \forall\ i \in \{1, \ldots, m\}$ (*non-emptyness criterion*).
2) If $l \in \mathcal{C}_{\mathcal{P}}$ then $Q_l(t) < Q_{l_j}(t) \ \ \forall\ l_j \in \mathcal{N}_l \cap \{l_1, l_m\}$ (*isolation criterion*).

Consider the iterative step of the Sequential Maximal Tree Scheduling. If $m \in \{1, 2\}$, at least 1 link in $\mathcal{P}$ is scheduled during the first phase at t. If $m = 3$, either l_2 is scheduled during the first phase or two links in $\mathcal{P}$ are scheduled in the first two phases at t. If $m > 3$, at least 2 links in $\mathcal{P}$ are scheduled during the first two phases at t.

The statement and the proof for this lemma is similar to that for lemma 1 for the special case that $\mathcal{G}$ is a path. The only difference is that this lemma holds under additional conditions, that is, when (a) the slot is un-constrained or (b) the segment does not contain the first link of the path.

Lemma 6: Let κ and B be positive integers such that $B \geq 5\kappa + 1$. Consider a path $\mathcal{P} \subseteq \mathcal{H} \setminus \{l_{\mathcal{H},1}\}$ where $\mathcal{H}$ is a path in $\{\mathcal{H}_k\}$. Let $\mathcal{P}$ consist of links $l_1, \ldots, l_m$, where $1 \leq m \leq 5$. Consider an event $\mathcal{A}$ that occurs if and only if there exists a time t such that

1) $Q_{l_i}(t) \geq B - \kappa \ \ \forall\ i \in \{1, \ldots, m\}$ (*lower bound criterion*)
2) $Q_{l_i}(t') \leq B - 1 \ \ \forall\ i \in \{1, \ldots, m\}$ and $\forall\ t' \leq t$ (*upper bound criterion*), and
3) $Q_l(t') < B - 5\kappa\ \forall\ l \in \mathcal{C}_{\mathcal{P}} \cap \mathcal{H},\ \forall\ t' < t$ (*boundary condition*).

Then $\mathbf{P}(\mathcal{A}) \leq 5\chi_{\xi/3}(\frac{\max_{l \in \mathcal{L}} \sigma_l}{4\kappa})^{\alpha}$.

The above lemma is similar to that for lemma 2 for the special case that $\mathcal{G}$ is a path. The only difference is that this lemma applies only for segments that do not contain the first link of the path. This lemma can be proved using lemma 5 just as lemma 2 has been proved using lemma 1.

The following lemmas, lemmas 7 and 8 do not have counterparts in the special case that $\mathcal{G}$ is a path.

Lemma 7: Consider an arbitrary path $\mathcal{H} \in \{\mathcal{H}_k\}$. Consider two adjacent links $l_{\mathcal{H},i}, l_{\mathcal{H},i+1}$ in $\mathcal{H}$, where $1 \leq i \leq \min(4, |\mathcal{H}| - 1)$. Consider a slot t that satisfies: 1) $\min\left(Q_{l_{\mathcal{H},i}}(t), Q_{l_{\mathcal{H},i+1}}(t)\right) > 0$, and 2) if $i + 2 \leq |\mathcal{H}|$, $Q_{l_{\mathcal{H},i+1}}(t) > Q_{l_{\mathcal{H},i+2}}(t)$. Then either $l_{\mathcal{H},i}$ or $l_{\mathcal{H},i+1}$ is scheduled in t.

Proof: First, let $i = 1$. In a constrained slot, clearly, $l_{\mathcal{H},2}$ is scheduled at the end of the first phase. In an un-constrained slot, consider a path $\mathcal{P}$ consisting of links $l_{\mathcal{H},1}, l_{\mathcal{H},2}$. Clearly, $\mathcal{P}$ satisfies the conditions of lemma 5. The result follows from the case with $|\mathcal{H}| = 2$ in lemma 5.

Now, let $i > 1$. First, let $Q_k(t) = 0$ for some k such that $1 \leq k < i$. Let $j = \max\{k : k < i, Q_k(t) = 0\}$. Consider path $\mathcal{P}$ consisting of links $l_{\mathcal{H},j+1}, \ldots, l_{\mathcal{H},i}, l_{\mathcal{H},i+1}$. Now, $\mathcal{P}$ consists of $i - j + 1$ links where $2 \leq i - j + 1 \leq 4$. Since $j + 1 > 1$, $l_{\mathcal{H},1} \notin \mathcal{P}$. Also, $\mathcal{P}$ satisfies the conditions of lemma 5. Let $i - j + 1 = 2$. Then, $\mathcal{P}$ consists of $l_{\mathcal{H},i}, l_{\mathcal{H},i+1}$. The result follows from the case with $|\mathcal{H}| = 2$ in lemma 5. Let $i - j + 1 = 3$. Then, $\mathcal{P}$ consists of $l_{\mathcal{H},i-1}, l_{\mathcal{H},i}, l_{\mathcal{H},i+1}$. The result follows from the case with $|\mathcal{H}| = 3$ in lemma 5. Let $i - j + 1 = 4$. Then, $\mathcal{P}$ consists of $l_{\mathcal{H},i-2}, l_{\mathcal{H},i-1}, l_{\mathcal{H},i}, l_{\mathcal{H},i+1}$. From lemma 5 with $|\mathcal{H}| = 4$, at least 2 links in $\{l_{\mathcal{H},i-2}, \ldots, l_{\mathcal{H},i+1}\}$ are scheduled at the end of the first two phases. Since $l_{\mathcal{H},i-2}$ and $l_{\mathcal{H},i-1}$ can not be scheduled simultaneously, one of the scheduled links must be $l_{\mathcal{H},i}$ or $l_{\mathcal{H},i+1}$. The result follows.

Now, let $Q_k(t) > 0$ for all k, $1 \leq k < i$. In a constrained slot, consider a path $\mathcal{P}$ consisting of links $l_{\mathcal{H},2}, \ldots, l_{\mathcal{H},i}, l_{\mathcal{H},i+1}$. Now, $\mathcal{P}$ consists of i links where $2 \leq i \leq 4$. Also, $l_{\mathcal{H},1} \notin \mathcal{P}$. The

result follows using the same arguments as in the previous paragraph. Consider an un-constrained slot and a path $\mathcal{P}$ consisting of links $l_{\mathcal{H},1},\ldots,l_{\mathcal{H},i},l_{\mathcal{H},i+1}$. Let $i<4$. Now, $\mathcal{P}$ consists of $i+1$ links where $3\le i+1\le 4$. Again, $\mathcal{P}$ satisfies the conditions of lemma 5. The result follows using the same arguments as in the previous paragraph. Finally, let $i=4$. Now, $\mathcal{P}$ consists of 5 links: $l_{\mathcal{H},1},\ldots,l_{\mathcal{H},5}$. Let neither $l_{\mathcal{H},4}$ nor $l_{\mathcal{H},5}$ be scheduled at the end of the second phase. From lemma 5 for $|\mathcal{H}|=5$, at least 2 links in $\mathcal{P}$ are scheduled at the end of the second phase. Thus, $l_{\mathcal{H},1}$ and $l_{\mathcal{H},3}$ must be scheduled at the end of the second phase. Thus, $l_{\mathcal{H},4}$ does not contend in the third phase, $l_{\mathcal{H},5}$ contends in the third phase, and $l_{\mathcal{H},6}$ (if $|\mathcal{H}|\ge 6$) does not contend in the third phase (since $Q_{l_{\mathcal{H},5}}(t)>Q_{l_{\mathcal{H},6}}(t)$). Thus, $l_{\mathcal{H},5}$ is scheduled in the third phase. The result follows. ∎

Lemma 7 does not hold if the iterative step of the Sequential Maximal Tree Scheduling has two or fewer phases. Consider a path $\mathcal{H}$ in $\mathcal{G}$ with 6 links $l_1,\ldots,l_6$. Let $Q_{l_1}(t)>Q_{l_2}(t)\ldots Q_{l_6}(t)>0$. Thus, l_4,l_5 satisfy the conditions of the lemma. Let $\mathcal{H}$ not be constrained in slot t. Only l_1 and l_3 are scheduled at the end of the first two phases of the iterative step. If the iterative step has only two phases, then l_5 and l_6 subsequently contend using maximal scheduling, and let l_5 lose this contention. Thus, neither l_4 nor l_5 are scheduled.

Lemma 8: Consider an arbitrary path $\mathcal{H}\in\{\mathcal{H}_k\}$, where $\mathcal{H}=\{l_{\mathcal{H},1},\ldots,l_{\mathcal{H},m}.\}$ Let $\mathcal{H}$ satisfy the constraint-lower-bound. Let B and β be positive integers such that $\beta<B/4$. Consider a link $l_{\mathcal{H},i}$ in $\mathcal{H}$, $1\le i\le 5$, and an event $\mathcal{A}$ that occurs if and only if there exists a slot t such that

1) $Q_{l_{\mathcal{H},i}}(t)\ge B-\beta$,
2) $\max_{l\in\mathcal{N}_{l_{\mathcal{H},i}}\cap\mathcal{H}}Q_l(t')\le B-1$ for each $t'\in[0,t]$ and
3) if $i<m$, $Q_{l_{\mathcal{H},i+1}}(t')<B-4\beta\ \ \forall\ t'\in(0,t]$.

Then $\mathbf{P}(\mathcal{A})\le(2\chi_{1/6}+\gamma_{\mathcal{H}})(\max_{l\in\mathcal{L}}\sigma_l/\beta)^{\alpha}$.

Lemma 8 does not hold when the iterative step has one or two phases, as its proof uses lemma 7 which does not hold in this case.

Proof: Let $\mathcal{A}$ occur. Then there exists a slot $t_2\in(0,t)$ such that $Q_{l_{\mathcal{H},i}}(t_2)=B-4\beta$ and $Q_{l_{\mathcal{H},i}}(t')\ge B-4\beta$ for all $t'\in[t_2,t]$, and either $m=i$ (case (a)) or $Q_{l_{\mathcal{H},i+1}}(t')<B-4\beta$ for all $t'\in[t_2,t]$ (case (b)). Also, $t-t_2\ge 3\beta/\max_{l\in\mathcal{L}}\sigma_l$.

First, let $i=1$. In both cases (a) and (b), $l_{\mathcal{H},1}$ is scheduled in each un-constrained slot in $[t_2,t)$. Thus, $S_{l_{\mathcal{H},1}}(t_2,t)=\Theta_{\mathcal{H}}(t_2,t)$. Now, $Q_{l_{\mathcal{H},1}}(t)=Q_{l_{\mathcal{H},1}}(t_2)+A_{l_{\mathcal{H},1}}(t_2,t)-S_{l_{\mathcal{H},1}}(t_2,t)$. Thus, $A_{l_{\mathcal{H},1}}(t_2,t)\ge S_{l_{\mathcal{H},1}}(t_2,t)=\Theta_{\mathcal{H}}(t_2,t)$. This implies that either $A_{l_{\mathcal{H},1}}(t_2,t)\ge(\rho_{l_{\mathcal{H},1}}+1/6)(t-t_2)$ or $\Theta_{\mathcal{H}}(t_2,t)\le(\rho_{l_{\mathcal{H},1}}+1/6)(t-t_2)$. From (2), the probability of the first event is at most $\chi_{1/6}(t-t_2)^{-\alpha}$. From the constraint-lower-bound, the probability of the second event is at most $\gamma_{\mathcal{H}}(t-t_2)^{-\alpha}$. Thus, $\mathbf{P}(\mathcal{A})\le(\chi_{1/6}+\gamma_{\mathcal{H}})(t-t_2)^{-\alpha}$. The lemma follows for $i=1$ since $t-t_2\ge 3\beta/\max_{l\in\mathcal{L}}\sigma_l$.

Now, let $i>1$. Thus, there exists a slot $t_3\in(t_2,t)$ such that $Q_{l_{\mathcal{H},i}}(t_3)=B-2\beta$ and $Q_{l_{\mathcal{H},i}}(t')\ge B-2\beta$ for all $t'\in[t_3,t]$. Clearly, $t-t_3\ge\beta/\max_{l\in\mathcal{L}}\sigma_l$. Let $\mathcal{B}$ be the event that $Q_{l_{\mathcal{H},i-1}}(t')<B-2\beta$ for all $t'\in[t_3,t]$.

Let $\mathcal{A}\cap\mathcal{B}$ occur. In both cases (a) and (b), $l_{\mathcal{H},i}$ is scheduled in each slot in $[t_3,t]$. Thus, $S_{l_{\mathcal{H},i}}(t_3,t)=t-t_3$. Now, $Q_{l_{\mathcal{H},i}}(t)=Q_{l_{\mathcal{H},i}}(t_3)+A_{l_{\mathcal{H},i}}(t_3,t)-S_{l_{\mathcal{H},i}}(t_3,t)$. Thus, $A_{l_{\mathcal{H},i}}(t_3,t)\ge S_{l_{\mathcal{H},i}}(t_3,t)=t-t_3$. From (2) and (3), $\mathbf{P}\left\{A_{l_{\mathcal{H},i}}(t_3,t)\ge t-t_3\right\}<\chi_{1/3}(t-t_3)^{-\alpha}$. Thus, $\mathbf{P}(\mathcal{A}\cap\mathcal{B})<\chi_{1/3}(t-t_3)^{-\alpha}\le\chi_{1/3}(\max_{l\in\mathcal{L}}\sigma_l/\beta)^{\alpha}$.

Now, let $\mathcal{A}\cap\mathcal{B}^c$ occur. Thus, $Q_{l_{\mathcal{H},i-1}}(t')\ge B-2\beta$ for some $t'\in[t_3,t]$; let t_4 be one such t'. Now, $t_4\in[t_2,t]$, $Q_{l_{\mathcal{H},i-1}}(t_4)\ge B-2\beta$, $Q_{l_{\mathcal{H},i}}(t_4)\ge B-2\beta$ (since $t_4\in[t_3,t]$). Thus,

$$\sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}}Q_l(t_4)\ge 2B-4\beta. \tag{22}$$

From the definition of t_2, there also exists a slot $t_5\in[t_2,t_4]$ such that $Q_l(t')\ge B-4\beta$ for all $l'\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}$ and $t'\in[t_5,t_4]$ and $\min\left(Q_{l_{\mathcal{H},i-1}}(t_5),Q_{l_{\mathcal{H},i}}(t_5)\right)=B-4\beta$. Clearly, $t_4-t_5\ge 2\beta/\max_{l\in\mathcal{L}}\sigma_l$. Since $4\beta<B$, and $1\le i-1\le 4$, in both cases (a) and (b), from lemma 7, either $l_{\mathcal{H},i-1}$ or $l_{\mathcal{H},i}$ is served in each slot in $[t_5,t_4]$. Thus, $\sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}}S_l(t_5,t_4)=t_4-t_5$. Now,

$$\begin{aligned}
&\sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}}A_l(t_5,t_4)\\
=&\sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}}Q_l(t_4)-\sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}}Q_l(t_5)\\
&+\sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}}S_l(t_5,t_4) &(23)\\
\ge&\ (2B-4\beta)-(B-4\beta)-(B-1)+t_4-t_5 &(24)\\
\ge&\sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}}(\rho_l+1/6)(t_4-t_5)\text{ (from (3))}. &(25)
\end{aligned}$$

Note that (24) above follows from (22) and the fact $\sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}}S_l(t_5,t_4)=t_4-t_5$.

Thus, either $A_{l_{\mathcal{H},i-1}}(t_5,t_4)\ge(\rho_{l_{\mathcal{H},i-1}}+1/6)(t_4-t_5)$, or $A_{l_{\mathcal{H},i}}(t_5,t_4)\ge(\rho_{l_{\mathcal{H},i}}+1/6)(t_4-t_5)$. From (2), the probability of each event is less than $\chi_{1/6}(t_5-t_4)^{-\alpha}$, which is upper bounded by $\chi_{1/6}(\max_{l\in\mathcal{L}}\sigma_l/2\beta)^{\alpha}$. Thus, $\mathbf{P}(\mathcal{A}\cap\mathcal{B}^c)<2\chi_{1/6}(\max_{l\in\mathcal{L}}\sigma_l/2\beta)^{\alpha}$.

Since $\mathbf{P}(\mathcal{A})=\mathbf{P}(\mathcal{A}\cap\mathcal{B})+\mathbf{P}(\mathcal{A}\cap\mathcal{B}^c)$, for $i>1$, $\mathbf{P}(\mathcal{A})<3\chi_{1/6}(\max_{l\in\mathcal{L}}\sigma_l/\beta)^{\alpha}$. The result follows.

∎

Lemma 9: Let $\mathcal{H}$ be a path in $\{\mathcal{H}_k\}$ that satisfies the constraint-lower-bound. Consider an integer $B\ge 6$ and a path $\mathcal{P}\subseteq\mathcal{H}$ consisting of links $l_{\mathcal{H},j},\ldots,l_{\mathcal{H},j+m-1}$ such that $m=\min(6,|\mathcal{H}|)$. Consider an event $\mathcal{A}$ that occurs if and only if there exists a slot t such that

$$\begin{aligned}
Q_{l_{\mathcal{H},i}}(t')&\le B-1\ \forall\ i\in\{j,\ldots,j+m-1\}\text{ and }\forall\ t'\le t,\\
Q_{l_{\mathcal{H},i}}(t)&\ge 5B/6\ \forall\ i\in\{j,\ldots,j+m-1\}.
\end{aligned}$$

Then $\mathbf{P}(\mathcal{A})\le(6\chi_{\min(\xi/2,1/6)}+\gamma_{\mathcal{H}})(6\max_{l\in\mathcal{L}}\sigma_l/5B)^{\alpha}$.

Lemma 9 is similar to lemma 3 for the special case that $\mathcal{G}$ is a path. The only difference is that this lemma holds under additional conditions, that is, when the path $\mathcal{H}$ satisfies the constraint-lower-bound. The proofs differ when $m<6$.

Proof: When $m=6$, in every slot in which every link in $\mathcal{P}$ has a packet to transmit, at least 2 links in $\mathcal{P}$ are scheduled for service. This clearly holds when either the slot is un-constrained or $l_{\mathcal{H},1}\notin\mathcal{P}$. If $\mathcal{P}$ consists of $l_{\mathcal{H},1}$ and the slot is constrained, at least 2 links are scheduled among $l_{\mathcal{H},2}\ldots l_{\mathcal{H},6}$. Using the above, the proof in this case follows using the same arguments as in the proof for lemma 3 in the case that $m=6$.

Now, let $m<6$. Then $m=|\mathcal{H}|$. Thus, $\mathcal{P}=\mathcal{H}$.

Let $m>1$. Thus, $Q_l(t)\ge 5B/6$ for all $l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}$. Thus,

$$\sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}}Q_l(t)\ge 5B/3. \tag{26}$$

Also, there exists a slot $t_1<t$ such that $Q_{l_{\mathcal{H},i}}(t')>0$ for all $l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}$ and $t'\in(t_1,t]$, and $\min_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}}Q_{l_{\mathcal{H},i}}(t_1)=0$. Clearly, $t-t_1\ge 5B/6\max_{l\in\mathcal{L}}\sigma_l$. From lemma 7, either $l_{\mathcal{H},m-1}$ or $l_{\mathcal{H},m}$ is served in each slot in $(t_1,t]$. Thus, $\sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}}S_l(t_1,t)\ge t-t_1-1$.

Now,

$$\begin{aligned}
&\sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} A_l(t_1,t)\\
=&\sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} Q_l(t) - \sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} Q_l(t_1)\\
&+\sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} S_l(t_1,t) \qquad (27)\\
\geq\ & 5B/3-(B-1)+(t-t_1)-1 \qquad (28)\\
\geq\ &\sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} (\rho_l+1/6)(t-t_1)\text{(from (3))}. \qquad (29)
\end{aligned}$$

Note that (28) above follows from (26) and from the fact $\sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} S_l(t_1,t)\geq t-t_1-1$.

Thus, either $A_{l_{\mathcal{H},m-1}}(t_1,t)\geq(\rho_{l_{\mathcal{H},m-1}}+1/6)(t-t_1)$, or $A_{l_{\mathcal{H},m}}(t_1,t)\geq(\rho_{l_{\mathcal{H},m}}+1/6)(t-t_1)$. From (2), the probability of each event is less than $\chi_{1/6}(t-t_1)^{-\alpha}$, which is upper bounded by $\chi_{1/6}(6\max_{l\in\mathcal{L}}\sigma_l/5B)^{\alpha}$. Thus, $\mathbf{P}(\mathcal{A})<2\chi_{1/6}(6\max_{l\in\mathcal{L}}\sigma_l/5B)^{\alpha}$.

Let $m=1$. Thus, $\mathcal{P}$ and $\mathcal{H}$ consist of only one link $l_{\mathcal{H},1}$. Thus, $Q_{l_{\mathcal{H},1}}(t)\geq 5B/6$. Thus, there exists a slot $t_1<t$ such that $Q_{l_{\mathcal{H},1}}(t')>0$ for all $t'\in(t_1,t]$, and $Q_{l_{\mathcal{H},1}}(t_1)=0$. Again, $t-t_1\geq 5B/6\max_{l\in\mathcal{L}}\sigma_l$. Clearly, $l_{\mathcal{H},1}$ is scheduled in each un-constrained slot in $(t_1,t]$. Thus, $S_{l_{\mathcal{H},1}}(t_1,t)\geq\Theta_{\mathcal{H}}(t_1,t)-1$. Now, $A_{l_{\mathcal{H},1}}(t_1,t)=Q_{l_{\mathcal{H},1}}(t)-Q_{l_{\mathcal{H},1}}(t_1)+S_{l_{\mathcal{H},1}}(t_1,t)$. Thus, $A_{l_{\mathcal{H},1}}(t_1,t)\geq 5B/6+\Theta_{\mathcal{H}}(t_1,t)-1\geq\Theta_{\mathcal{H}}(t_1,t)$ (since $B\geq 2$)). This implies that either $A_{l_{\mathcal{H},1}}(t_1,t)\geq(\rho_{l_{\mathcal{H},1}}+1/6)(t-t_1)$ or $\Theta_{\mathcal{H}}(t_1,t)\leq(\rho_{l_{\mathcal{H},1}}+1/6)(t-t_1)$. From (2), the probability of the first event is at most $\chi_{1/6}(t-t_1)^{-\alpha}$. From the constraint-lower-bound, the probability of the second event is at most $\gamma_{\mathcal{H}}(t-t_1)^{-\alpha}$. Thus, $\mathbf{P}(\mathcal{A})\leq(\chi_{1/6}+\gamma_{\mathcal{H}})(t-t_1)^{-\alpha}$. The lemma follows for $m=1$ since $t-t_1\geq 5B/6\max_{l\in\mathcal{L}}\sigma_l$. ∎

Consider an arbitrary path $\mathcal{H}\in\{\mathcal{H}_i\}$ and the corresponding node u in $\mathcal{G}^P$. Let $\mathcal{F}_{\mathcal{H}}=\{v: v \text{ is either the parent or an older sibling of } u \text{ in } \mathcal{G}^P\}$.

Lemma 10: Consider an arbitrary path $\mathcal{H}\in\{\mathcal{H}_j\}$. Let for each $t>0$, $l\in\cup_{i\in\mathcal{F}_{\mathcal{H}}}\mathcal{H}_i$, $\mathbf{P}\{Q_l(t)\geq B\}\leq\mu_{\mathcal{H}_i}B^{-\alpha}$. Then $\mathcal{H}$ satisfies the constraint-lower-bound with $\gamma_{\mathcal{H}}=\Delta\chi_{(1/18\Delta)}+\Delta\nu(\Delta,\alpha)+18^{\alpha}\Delta^{\alpha+1}\max_{i\in\mathcal{F}_{\mathcal{H}}}\mu_{\mathcal{H}_i}$, where $\nu(\Delta,\alpha)$ is a constant whose value depends on Δ,α.

Lemma 10 does not have a counterpart for the special case that $\mathcal{G}$ is a path.

Proof: Consider $i\in\mathcal{F}_{\mathcal{H}}$. For each $0<t_1<t_2$, let $U_l(t_1,t_2)$ be the number of slots in $[t_1,t_2)$ in which link l in $\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}}$ is undecided just before the start of the scheduling phase of $\mathcal{H}$. Let $W=\lceil\ln(36\Delta)/(-\ln(\gamma))\rceil$. Each link in $\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}}$ executes maximal scheduling for at least W rounds before $\mathcal{H}$ starts its scheduling phase, and it is undecided at the end of W rounds with a probability of at most γ^W, which is less than $1/(36\Delta)$. Thus, $U_l(t_1,t_2)$ is stochastically lesser than the sum of t_2-t_1 independent Bernoulli random variables each of which is 1 w.p. $1/(36\Delta)$ and 0 otherwise. Thus, from Bernstein's inequality (p. 32, [8]),

$$\begin{aligned}
&\mathbf{P}\left\{U_l(t_1,t_2)\geq\frac{t_2-t_1}{18\Delta}\right\}\\
\leq\ & e^{-\frac{t_2-t_1}{4\times 36^2\times\Delta^2}}\\
\leq\ & \nu(\Delta,\alpha)(t_2-t_1)^{-\alpha}\ \forall\ l\in\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}}, \qquad (30)
\end{aligned}$$

where, $\nu(\Delta,\alpha)$ is a constant whose value depends on Δ,α.

Clearly, $\Theta_{\mathcal{H}}(t_1,t_2)\geq(t_2-t_1)-\sum_{l\in\cup_{i\in\mathcal{F}_{\mathcal{H}}}(\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}})}(S_l(t_1,t_2)+U_l(t_1,t_2))$. Let $\Theta_{\mathcal{H}}(t_1,t_2)\leq(\rho_{l_{\mathcal{H},1}}+1/6)(t_2-t_1)$. Then,

$$\begin{aligned}
&\sum_{l\in\cup_{i\in\mathcal{F}_{\mathcal{H}}}(\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}})}(S_l(t_1,t_2)+U_l(t_1,t_2))\\
\geq\ & (5/6-\rho_{l_{\mathcal{H},1}})(t_2-t_1)\\
\geq\ & (1/6+\sum_{l\in\cup_{i\in\mathcal{F}_{\mathcal{H}}}(\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}})}\rho_l)(t_2-t_1). \qquad (31)
\end{aligned}$$

The last inequality follows from (3) since all links in $\cup_{i\in\mathcal{F}_{\mathcal{H}}}(\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}})$ intersect at the same node in $\mathcal{G}$

Now, $S_l(t_1,t_2)\leq Q_l(t_1)+A_l(t_1,t_2)$. Thus, from (31), and since $|\cup_{i\in\mathcal{F}_{\mathcal{H}}}(\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}})|\leq\Delta$,

$$\begin{aligned}
&\sum_{l\in\cup_{i\in\mathcal{F}_{\mathcal{H}}}(\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}})}(A_l(t_1,t_2)+Q_l(t_1)+U_l(t_1,t_2))\\
\geq\ &\sum_{l\in\cup_{i\in\mathcal{F}_{\mathcal{H}}}(\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}})}(1/(6\Delta)+\rho_l)(t_2-t_1).
\end{aligned}$$

Thus, either $A_l(t_1,t_2)\geq(1/18\Delta+\rho_l)(t_2-t_1)$ or $Q_l(t_1)\geq(t_2-t_1)/18\Delta$ or $U_l(t_1,t_2)\geq(t_2-t_1)/18\Delta$ for some $l\in\cup_{i\in\mathcal{F}_{\mathcal{H}}}(\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}})$. From assumption, the probability that $Q_l(t_1)\geq(t_2-t_1)/18\Delta$ is at most $\mu_{\mathcal{H}_i}(\frac{t_2-t_1}{18\Delta})^{-\alpha}$ if $l\in H_i$ and $i\in\mathcal{F}_{\mathcal{H}}$. The result follows from (2) and (30). ∎

2) Main Result: Theorem 2 is proved using an induction argument, and the proof for the base case is similar to the proof for Theorem 1.

Proof: We first prove the first part of the theorem. We will prove that for any $t>0$, for all $l\in\mathcal{H}_j$,

$$\mathbf{P}\{Q_l(t)\geq B\}\leq\tau_{p_j,r_j}B^{-\alpha}, \qquad (32)$$

where τ_{p_j,r_j} is defined through the recursions in the statement of the theorem. The result follows since $\tau_{x,y}$ increases with increase in x,y and $p_i\leq\hat{p}$ and $r_i\leq\Delta-1$ for all i.

We prove using induction on the level of j, p_j and the number of older siblings of j, r_j.

First consider $p_j=0$. Since for all x, $\tau_{0,x}\geq\left(12\times4^5\times5^5\max_{l\in\mathcal{L}}\sigma_l\right)^{\alpha}$, (32) trivially holds for $B<12\times4^5\times5^5\max_{l\in\mathcal{L}}\sigma_l$. Let $B\geq12\times4^5\times5^5\max_{l\in\mathcal{L}}\sigma_l$. Now, j is the root of $\mathcal{G}^P$ and hence does not have any sibling. Thus, $r_j=0$. Thus, every slot is an un-constrained slot for $\mathcal{H}_j$. Hence, from (3), $\mathcal{H}_j$ satisfies the constraint-lower-bound with $\gamma_{\mathcal{H}_j}=\gamma_{0,0}=0$. Consider the event $\mathcal{A}(B,t)$ that occurs if and only if $\max_{l\in\mathcal{H}_i}Q_l(t)\geq B$. Let $\mathcal{A}(B,t)$ occur. Then there exists a slot $t_0\leq t$ such that $Q_l(t')\leq B-1$ for all $l\in\mathcal{H}_j$ and $t'\leq t_0$ and $Q_l(t_0+1)\geq B$ for some $l\in\mathcal{H}_j$; let $l_{\mathcal{H}_j,k}$ be one such l. Then, $Q_{l_{\mathcal{H}_j,k}}(t_0)\geq B-\max_{l\in\mathcal{L}}\sigma_l$. For $0\leq q\leq5-k$, if $|\mathcal{H}_j|>k+q$, event $\mathcal{B}_q$ is said to occur if the event $\mathcal{A}$ described in lemma 8 occurs with $i=k+q$ and $\beta=4^q\lfloor B/(6\times4^5\times5^5)\rfloor$. If $|\mathcal{H}_j|\geq6$, for $1\leq c\leq6$, $1\leq d\leq c$, consider paths $\mathcal{P}_{c,d}\subseteq\mathcal{H}_j$ consisting of c links with the dth link being $l_{\mathcal{H}_j,\max(k,6)}$. If $c\leq5$ $l_{\mathcal{H}_1,1}\notin\mathcal{P}_{c,d}$. For $1\leq c\leq5, 1\leq d\leq c$, event $\mathcal{C}_{c,d}$ is said to occur if the event $\mathcal{A}$ described in lemma 6 occurs with $\mathcal{P}=\mathcal{P}_{c,d}$, $\kappa=4^5 5^{c-1}\lfloor\frac{B}{6\times4^5\times5^5}\rfloor$. Event $\mathcal{C}_{6,d}$ is said to occur if the event $\mathcal{A}$ described in lemma 9 occurs with $\mathcal{P}=\mathcal{P}_{6,d}$.

Clearly, when $\mathcal{A}(B,t)$ occurs, $\mathcal{B}_q$ or $\mathcal{C}_{c,d}$ occurs for some c,d,q, $0\leq q\leq5-j, 1\leq c\leq6, 1\leq d\leq c$. Thus, $\mathbf{P}(\mathcal{A}(B,t))$ is upper bounded by the sum of the probabilities of the events $\mathcal{B}_q$, $\mathcal{C}_{c,d}$ for $0\leq q\leq5-j, 1\leq c\leq6, 1\leq d\leq c$. Thus (32) follows from the upper bounds of the probabilities of these events provided in lemmas 6, 8, 9.

We now consider the induction case. Now, let (32) hold for all i such that $p_i\leq h$. We will prove the hypothesis for i such that $p_i=h+1$. The proof is the same as that for the base case once we can show that $\mathcal{H}_i$ satisfies the constraint-lower-bound with $\gamma_{\mathcal{H}_i}=\gamma_{h+1,r_i}$. First consider $\mathcal{H}_i$ such that $p_i=h+1$ and $r_i=0$. Thus,

i does not have an older sibling in $\mathcal{G}^P$. Since i's parent's level is h, i's parent satisfies (32). Now, lemma 10 shows that $\mathcal{H}_i$ satisfies the constraint-lower-bound with $\gamma_{\mathcal{H}_i} = \gamma_{h+1,r_i}$. Now, using the same proof as that for the base case, we can show that (32) holds for i. Now, let (32) hold for all i such that $p_i = h+1$ and $r_i \leq a$. Let $p_i = h+1$ and $r_i = a+1$. Now, i's parent and older siblings satisfy (32). Again, lemma 10 shows that $\mathcal{H}_i$ satisfies the constraint-lower-bound with $\gamma_{\mathcal{H}_i} = \gamma_{h+1,a+1}$. Thus, as before, (32) holds for i.

Thus, the first part of the theorem holds. The second part is immediate from the first. ■

V. Conclusion

In this paper, we provide a policy that attains queue-length stability under mild assumptions on the arrival process. This policy approximates the maximum throughput region within a factor of $2/3$ in tree topologies under primary interference constraints, can be implemented in a fully distributed manner, and requires $O(\Delta \log \Delta \log n)$ computation time. The computation time of our policy is comparable (within a $\log \Delta$ factor) to that of existing maximal scheduling based policies that can only attain up to $1/2$ of the maximum throughput region. It would be interesting to investigate whether, without significantly increasing the computation time, the approximation ratio can be improved and the results can be extended for cyclic graphs and other interference models for the same class of polynomially convergent arrival processes. In a companion paper, we show that when the arrival process is i.i.d., the stability region can be approximated arbitrary closely for a large class of networks and interference models with a computation time that depends only on the approximation factor and the maximum node degree in the network [15]. The results in the two papers complement each other.

VI. Acknowledgement

We would like to thank Professor Sudipto Guha at University of Pennsylvania for several discussions on Lemma 4 which helped us to substantially simplify its proof. This research was supported in part by NCR 0238340, CNS-0435141, CNS 0435306, CNS-0448316, ECCS 0621782, CNS 0721308

References

[1] A. Brzezinski, G. Zussman, and E. Modiano. Distributed throughput maximization in wireless mesh networks - a partitioning approach. In *Proceedings of ACM MOBICOM*, Los Angeles, CA, September 2006.

[2] P. Chaporkar, K. Kar, and S. Sarkar. Throughput guarantees through maximal scheduling in multihop wireless networks. In *Proceedings of 43d Annual Allerton Conference on Communication, Control and Computing*, Allerton, Monticello, Illinois, September 28-30 2005.

[3] J. Dai and B. Prabhakar. The throughput of data switches with and without speedup. In *Proceedings of INFOCOM*, pages 556–564, Tel Aviv, Israel, Mar 2000.

[4] J. G. Dai. On the positive Harris recurrence for multiclass queueing networks: A unified approach via fluid models. *The Annals of Applied Probability*, 5:49–77, 1995.

[5] A. Dimakis and J. Walrand. Sufficient conditions for stability of longest queue first scheduling: second order properties using fluid limits. *Advances of applied Probability*, 38(2):505–521, June 2006.

[6] D. Shah E. Modiano and G. Zussman. Maximizing throughput in wireless networks via gossiping. In *Proc. ACM SIGMETRICS / IFIP Performance'06*, June 2006.

[7] G. Fayolle, V. A. Malyshev, and M. V. Menshikov. *Topics in the Constructive Theory of Countable Markov Chains*. Cambridge University Press, 1995.

[8] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, Great Clarendon Street, Oxford, U.K, 3 edition, 2001.

[9] B. Hajek and G. Sasaki. Link scheduling in polynomial time. *IEEE Transactions on Information Theory*, 34(5):910–917, Sep 1988.

[10] X. Lin and S. Rasool. Constant-time distributed scheduling policies for ad hoc wireless networks. In *Proceedings of IEEE CDC-ECC'05*, San Diego, CA, Dec 2006.

[11] X. Lin and N. Shroff. The impact of imperfect scheduling on cross-layer rate control in multihop wireless networks. In *Proceedings of INFOCOM*, Miami, FL, Mar 2005.

[12] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1055, 1986.

[13] M. J. Neely, E. Modiano, and C. E. Rohrs. Dynamic power allocation and routing for time varying wireless networks. *IEEE Journal on Selected Areas in Communications, Special Issue on Wireless Ad-Hoc Networks*, 23(1):89–103, Jan 2005.

[14] D. Peleg. *Distributed Computing: A Locality-sensitive Approach*. Society of Industrial and Applied Mathematics, Philadelphia, PA, 2000.

[15] S. Ray and S. Sarkar. Arbitrary throughput versus complexity tradeoffs in wireless networks using graph partitioning. In *Proceedings of Information Theory and Applications Second Workshop*, University of California at San Diego, 2007.

[16] T. Salonidis and L. Tassiulas. Distributed dynamic scheduling for end-to-end rate guarantees in wireless ad hoc networks. In *Proceedings of ACM MOBIHOC*, 2005.

[17] D. Shah, P. Giaccone, and B. Prabhakar. An efficient randomized algorithm for input-queued switch scheduling. *IEEE Micro*, 22(1):19–25, Jan-Feb 2002.

[18] L. Tassiulas. Linear complexity algorithms for maximum throughput in radio networks and input queued switches. In *Proceedings of INFOCOM*, pages 533–539, 1998.

[19] L. Tassiulas and A. Ephremidis. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, Dec 1992.

[20] X. Wu and R. Srikant. Regulated maximal matching: a distributed scheduling algorithm for multihop wireless networks with node-exclusive spectrum sharing. In *Proceedings of IEEE CDC-ECC'05*, Seville, Spain, Dec 2005.

Saswati Sarkar Saswati Sarkar (S'98, M'00) received Master of Engineering in Electrical Communication Engineering from the Indian Institute of Science, Bangalore in 1996 and Phd in Electrical and Computer Engineering from University of Maryland, College Park in 2000. She is currently an Associate Professor in the department of Electrical and Systems Engineering in University of Pennsylvania. Her research interests are in resource allocation and performance analysis in communication networks. She received the Motorola gold medal for the best masters student in the division of electrical sciences at the Indian Institute of Science and a National Science Foundation (NSF) Faculty Early Career Development Award in 2003. She was an associate editor of IEEE Transaction on Wireless Communications from 2001 to 2006.

Koushik Kar Koushik Kar received the Ph.D. and M.S. degrees in Electrical and Computer Engineering from the University of Maryland, College Park, in 2002 and 1999, respectively. He received the B.Tech degree in Electrical Engineering from Indian Institute of Technology, Kanpur, in 1997. Since 2002, he has been an assistant professor in the Electrical, Computer and Systems Engineering department at Rensselaer Polytechnic Institute, Troy, NY. His research interests include performance optimization questions in ad-hoc and sensor networks, traffic engineering, congestion control and multicasting. Dr. Kar received the National Science Foundation (NSF) Faculty Early Career Development Award from the National Science Foundation in 2005.