Achieving 2/3 Throughput Approximation with Sequential Maximal Scheduling under Primary Interference Constraints

Saswati Sarkar and Koushik Kar

***Abstract*— In this paper, we present a scheduling policy, *sequential maximal tree scheduling* that attains $\frac{2}{3}$ of the maximum throughput region in tree-graphs under primary interference constraints. The computation time of the policy varies as the square of the logarithm of the network size. Our results are a significant improvement over previous results which can attain only $\frac{1}{2}$ of the maximum throughput region, even for graphs that have a simple path topology, in similar computation time.**

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. Using this framework, Tassiulas *et al.* have characterized the maximum attainable throughput region and also provided a scheduling strategy that attains this throughput region in any given wireless network [10]. The computation time for this policy is however exponential in the size of the network. Later, Tassiulas [9] and Shah *et al.* [8] provided randomized scheduling schemes that attain the maximum achievable throughput region, which can be implemented in fully distributed manner using gossip based algorithms [2]. The computation times of the above policies are linear in the size of the network.

Recent research has focused on attaining provable throughput guarantees using fully distributed scheduling policies whose computation times are logarithmic in the size of the network. A class of simple distributed scheduling policies, referred to as *maximal scheduling* policies satisfy the above criteria. Maximal scheduling only ensures that if a transmitter u has a packet to transmit to a receiver v, either (u, v) or a transmitter-receiver pair that can not simultaneously transmit with (u, v) is scheduled for transmission; the scheduling is otherwise arbitrary. We have earlier shown that for arbitrary interference models the fraction of the throughput region guaranteed by maximal scheduling is the reciprocal of the maximum "interference degree" of links in the network, where interference degree of any link refers to the maximum number of links that interfere (i.e., cannot be scheduled simultaneously) with the given link, but do not interfere with each other [1]. Furthermore, the above throughput guarantee is also tight, that is, there exists maximal scheduling policies whose throughput region is at most the reciprocal of the maximum interference degree in the network [1].

In this paper, we focus on a specific interference model, the node-exclusive spectrum sharing model, where the only scheduling restrictions are due to *primary* interference constraints, i.e., a node cannot communicate with multiple nodes simultaneously. Thus, a set of links can be simultaneously scheduled if and only if they constitute a matching. This specific interference model holds only when every node has a unique frequency in its two-hop neighborhood. Lin *et al.* [4] and Wu *et al.* [11] have shown that maximal scheduling is guaranteed to attain at least half of the maximum throughput region under the node-exclusive spectrum sharing model. We have 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 [1]. An arbitrary maximal scheduling policy cannot therefore attain a worst-case performance ratio better than $\frac{1}{2}$ even in the special case of tree-graphs. Salonidis *et al.* [7] provides a policy that attains the maximum throughput region in the special case of tree graphs; the policy however needs the arrival rates in all links and therefore must be recomputed every time these rates change.

In this paper, we present a queue length based maximal scheduling policy that attains $2/3$ of the maximum throughput region for tree graphs under node-exclusive spectrum sharing model. The policy does not use any knowledge of the arrival rates and requires a computation time which is logarithmic in the size of the network. The policy therefore attains a substantially higher performance guarantee in tree graphs as compared to arbitrary maximal scheduling policies while retaining their computational simplicity. Although the applicability of our results to general graphs remain open, it is worth noting that the class of graphs that we consider - tree graphs - 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.

The paper is organized as follows. We describe the system model and the terminology in Section II. Our algorithm in presented in Section III, and its throughput guarantees are analyzed in Section IV.

This work was supported by the National Science Foundation under grants NCR-0238340, CNS-0435306, CNS-0448316 and CNS-0435141.

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.

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 directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ and $\mathcal{L}$ respectively denote the sets of nodes and (directed) links. Let $n = |\mathcal{L}|$. A link exists from a node u to another node v if and only if v can receive u's signals. The link set $\mathcal{L}$ depends on the transmission power levels of nodes and the propagation conditions in different directions. 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 $\mathcal{G}$.

Each link is associated with a unique identifier (id). 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. Since we consider the node-exclusive spectrum sharing (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.

Next we state our assumptions on the packet arrival process. Let $A_l(t_1, t_2)$ denote the number of packets arriving at link l in interval $(t_1, t_2]$. We assume that $A_l(t, t+1) \leq \sigma_l \ \forall \ t, l \in \mathcal{L}$, where σ_l is an integer for each $l \in \mathcal{L}$, and $\max_{l \in \mathcal{L}} \sigma_l \geq 1$. Further, there exists a constant $\alpha > 1$ and an arrival rate vector $\vec{\rho} = (\rho_1, \ldots, \rho_{|\mathcal{L}|})$ such that the empirical average of the arrivals in the system 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 $\delta > 0$,

$$\mathbf{P}\left\{\left|\frac{A_l(t_3, t_4)}{t_4 - t_3} - \rho_l\right| \geq \delta\right\} < \frac{\chi_\delta}{(t_4 - t_3)^\alpha}. \qquad (1)$$

Clearly, χ_δ is a non-increasing function of δ. Note that a large class of arrival processes, e.g., periodic, i.i.d., and Markovian arrival processes with finite state space, satisfy the above assumption. We refer to ρ_i as the arrival rate for link i.

Next we introduce a few definitions.

Definition 1: The network is said to be *stable* 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 (1) 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.

Sequential Maximal Path Scheduling

Iterative Step:: For $k = 1$ to $k = 3$, 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"; links that are adjacent to scheduled links set their status to "un-scheduled".

Fig. 1. Sequential Maximal Path Scheduling Algorithm

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.

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

In this section, we describe our scheduling policy, and derive bounds on the fraction of the maximum throughput region attained by it. For ease of exposition, we will first present the algorithm for paths, and then show how it can be extended to trees.

A. Sequential Maximal Scheduling in Paths

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. Our algorithm, which we call *sequential maximal scheduling*, consists of several phases as described in Figure 1. In our algorithm, all links that do not have any packets to transmit set their status to *un-scheduled*. All other links, i.e., links that have packets to transmit, initially set their status set to *un-decided*.

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.

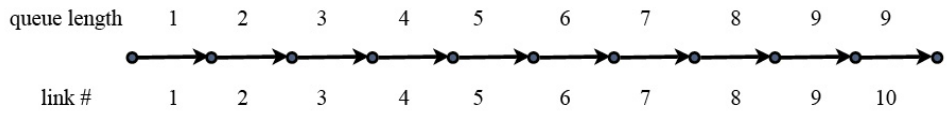

Fig. 2. Path Scheduling Example

The Sequential Maximal Path Scheduling algorithm attains a guaranteed fraction $\frac{2}{3}$ of the maximum throughput region. This result follows as a special case of a more general result proved later in the paper for tree graphs. The iterative step in Sequential Maximal Path Scheduling can be computed in constant time. In a degree-bounded graph, the expected computation time for the terminal step is $O(\log n)$ if maximal scheduling is computed using a distributed randomized algorithm like the one proposed in [5]. Thus, the expected computation time for Sequential Maximal Path Scheduling is $O(\log n)$.

We now describe the intuition behind the design of the Sequential Maximal Path Scheduling Policy and its performance guarantee. First, observe that if in any slot any segment of a path consists of 6 links all of which have packets to transmit, then any maximal scheduling policy schedules at least 2 links in the segment in the slot irrespective of the scheduling in the rest of the path. Furthermore, any scheduling policy can schedule at most 3 links in the segment in any slot. This suggests that any maximal scheduling policy should attain a throughput guarantee of at least $2/3$. It has however been shown that in any graph there exists one maximal scheduling policy whose throughput guarantee is exactly the reciprocal of the interference degree in the graph [1]. The interference degree in a path is 2 [1]. Thus, in a path there exists one maximal scheduling policy which attains a throughput guarantee of $1/2$. A close examination of the counter example establishing the above throughput guarantee [1] will resolve the apparent contradiction in the above observations, and also motivates the design of the Sequential Maximal Path Scheduling policy.

Consider a path with 3 links, l_1, l_2, l_3. Let l_2 be adjacent to both l_1 and l_3. Consider a maximal scheduling policy that serves l_2 only when neither l_1 nor l_3 has a packet to transmit. Consider an arrival rate vector of $(1/2, \rho, 1/2)$ and an arrival pattern in which l_1 (l_3) generates packets in the odd (even) slots. Note that l_1 (l_3) is scheduled in every odd (even) slot and l_3 is never scheduled. Thus, the system is unstable for any positive ρ. If we scale up the arrival rates by a factor $2-4\rho$, which is nearly 2 for small ρ, then the resulting rate vector $(1-2\rho, 2\rho-4\rho^2, 1-2\rho)$ which can be stabilized by the optimum policy since the sum of the rates of l_1 (l_3) and l_2 is less than unity. This example can be modified slightly to show the the above maximal scheduling policy cannot attain a throughput guarantee better than $1/2$ in this path [1].

The first observation in the paragraph before the example does not apply in the example as all links never simultaneously have packets to transmit $-$ in most slots either l_1 or l_3 does not have packets to transmit while the queue length in l_2 continually increases. This suggests that we need to devise a scheduling policy which ensures that (a) the scheduling is maximal and (b) if a link has a large queue length, then there exists a segment of at least 6 links including the link that have large queue lengths as well. Hence, the links in this segment have packets to transmit for a large number of slots. Thus, owing to the scheduling of a maximal set of links, at least 2 links are scheduled in this segment in each slot for a large interval. Thus, the policy provides a service rate of at least $2/3$ of that provided by any policy to the links in this segment, and since the arrival rates of links are at most $2/3$ of a feasible arrival rate vector, a $2/3$ throughput guarantee is attained.

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 accomplishes the above goal (that is if a link has a large queue length, then there exists a segment of at least 6 links including the link that have large queue lengths as well). The terminal step ensures that the scheduling is maximal. Note that each phase in the iterative step provides a higher priority to links with high queue lengths - it may therefore seem that one phase is sufficient to accomplish he desired goal. Multiple phases are however necessary for attainng the desired performance guarantee, and the necessity will become evident in course of the proof.

B. 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 $O(\log n)$. We refer to this new tree as a *path tree.* Now, if every path conducts Sequential Maximal Path Scheduling after waiting for a time interval in which its parent path in the path tree finishes its scheduling (with high probability), then the overall scheduling attains a throughput guarantee of $2/3$ and also terminates in $O(\log^2 n)$ time. The Sequential Maximal Scheduling that can be used in paths in a 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 $\mathcal{H}$. The performance guarantee however does not change because 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 $\mathcal{H}$.

1) Preliminaries: We now assume that $\mathcal{G}$ is a tree with maximum degree $\Delta \geq 1$. Since $\mathcal{G}$ is a tree, any path in $\mathcal{G}$ must be a simple path (i.e., a path with no cycles). Hence, the terminology *path* in this section will refer to a simple path.

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 $\mathcal{T}$ (with a designated root node) that satisfies the following properties. Consider two nodes u and v in $\mathcal{T}$ 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.

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

2) Path Tree Construction: From the tree graph $\mathcal{G}$, we construct a *path graph*, $\mathcal{G}^P = (\mathcal{V}^P, \mathcal{E}^P)$, where each vertex in $\mathcal{G}^P$ represents a path in $\mathcal{G}$. The path graph $\mathcal{G}^P$ is initialized as follows. Each path is set to correspond to a link in $\mathcal{G}$, i.e., each vertex in the path graph $\mathcal{G}^P$ is represents a single-link path in $\mathcal{G}$. We designate any node in $\mathcal{G}$ as the root, and traverse $\mathcal{G}$ using breadth first traversal. If two links l and l' belonging to $\mathcal{G}$ are such that either l is a parent of l' in the breadth first traversal of $\mathcal{G}$, or vice versa, then we draw an edge between the corresponding vertices in $\mathcal{G}^P$. We then add a dummy node r to $\mathcal{G}^P$, designate it as the root vertex of $\mathcal{G}^P$, and add edges between r and the vertices in $\mathcal{G}^P$ that correspond to the links attached to the root node of $\mathcal{G}$. Therefore, $|\mathcal{V}^P| = |\mathcal{L}| + 1 = |\mathcal{N}|$. It is easy to see that graph $\mathcal{G}^P$ is a tree, and is in fact isomorphic to the original tree graph $\mathcal{G}$. Thus, since the depth (maximum distance of a leaf node from the root) in $\mathcal{G}$ can be $O(n)$, the depth in $\mathcal{G}^P$ can be $O(n)$ as well, where $n = |\mathcal{N}|$.

PATH TREE COMPRESSION ALGORITHM

I. INITIALIZATION:

1) For all leaf vertices u in $\mathcal{G}^P$, set *subtree_balanced*(v) = TRUE; otherwise, set *subtree_balanced*(u) = FALSE.

II. ITERATIVE PATH TREE COMPRESSION:

While *subtree_balanced*(r) = FALSE, do the following:
For each node u in $\mathcal{G}^P$ such that *subtree_balanced*(u) = FALSE, run SUBTREE_BALANCE(u).

Fig. 3. Path Tree Compression Algorithm

SUBTREE_BALANCE(u)

IF (*subtree_balanced*(v) = TRUE for all children vertices v of u in $\mathcal{G}^P$), THEN do the following:

1) Compute d_u, the depth of the subtree rooted at u.
2) Compute $\mathcal{V}_u$, the set of children such that the subtree rooted at each of them has depth $d_u - 1$.
3) *If* ($|\mathcal{V}_u| = 1$), *then* do the following:
   Let v be the node such that $d_v = d_u - 1$. Then merge vertex v with vertex u (or in other words, shrink the edge connecting u ands v in graph $\mathcal{G}^P$). Thus the vertex now represented by u corresponds to the union of the paths earlier represented by nodes u and v, i.e., the path represented by v, concatenated with the single-link path earlier represented by u;
   Else Do nothing.
4) Set *subtree_balanced*(u) = TRUE;

ELSE Do nothing.

Fig. 4. The Subtree_Balance sub-routime used in the Path Tree Compression Algorithm.

We will next run a compression algorithm on this path tree $\mathcal{G}^P$ that will reduce the depth of $\mathcal{G}^P$ to $O(\log n)$. The algorithm works by merging paths appropriately (so as to create longer paths), starting at the bottom of the path tree $\mathcal{G}^P$. The algorithm is described in Figure 3.

Note that although the complexity of the path tree compression algorithm is $O(n^2)$ (where $n = |\mathcal{N}|$), it can be easily reduced to $O(n)$ be executing the SUBTREE_BALANCE procedures in a proper sequence, starting from the leaf nodes and moving up to the root.

Next we illustrate the Path Tree Compression Algorithm using an example, as shown in Figure 5.

Before we proceed further, we introduce a few definitions. A tree is a *binary balanced tree* rooted at r' if the shortest distance of any leaf vertex on the tree from r' is the same, and is equal to the depth of the tree. Note that a binary balanced tree must satisfy $d' = \log_2(n' + 1) - 1$, where d' is the depth of the tree, and n' is the number of vertices in the tree. A tree $\mathcal{T}$ is said to be a *subtree balanced tree* rooted at r' if it the following conditions hold: (i) Tree $\mathcal{T}$ contains a binary balanced tree $\mathcal{T}'$ rooted at r' as a subtree, (ii) If $d_{\mathcal{T}}$ and $d_{\mathcal{T}'}$ denote the depths of trees $\mathcal{T}$ and $\mathcal{T}'$ respectively, then $d_{\mathcal{T}} = d_{\mathcal{T}'}$.

Now we prove the following result.

Lemma 1: The PATH TREE COMPRESSION ALGORITHM terminates and the resulting graph $\mathcal{G}^P$ is a subtree balanced tree rooted at r.

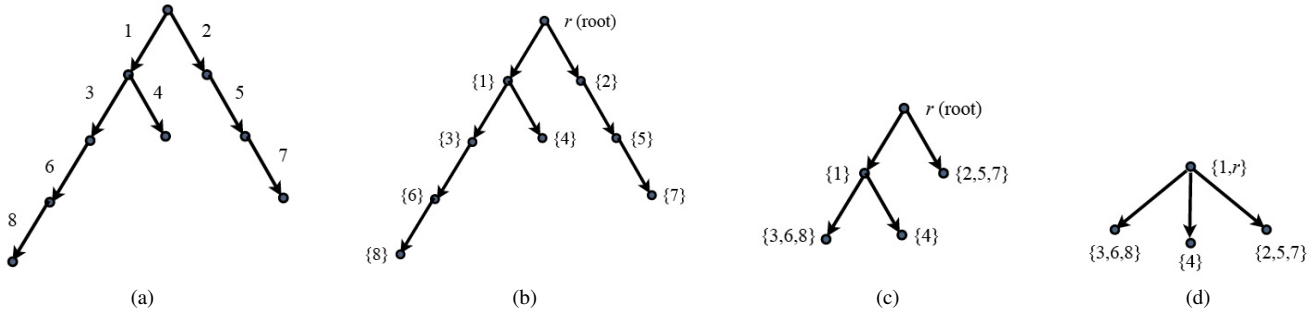

Fig. 5. Panel (a) shows the tree graph $\mathcal{G}$. Panel (b) show the initial path graph (tree) $\mathcal{G}^P$. Each node in the graph corresponds to a single-link path, as indicated across each node. The root r is a dummy node which is additionally inserted. Note that this graph is isomorphic to $\mathcal{G}$. Panel (c) shows the path graph (tree) $\mathcal{G}^P$ after the paths corresponding to links {3,6,8} and {2,5,7} gets compressed, and is each represented by a single node in $\mathcal{G}^P$. Panel (d) shows the path graph (tree) $\mathcal{G}^P$ after the paths corresponding to links {3,6,8} and {2,5,7} gets compressed, and each is represented by a single node in $\mathcal{G}^P$.

Proof: We first argue that the $\mathcal{G}^P$ always remains a tree during the execution of the PATH TREE COMPRESSION ALGORITHM. Recall that $\mathcal{G}^P$ is initially a tree. It is also easy to see that each vertex merging step in the Subtree_Balance procedure will preserve the tree structure, since it is equivalent to shrinking an edge in the tree. Therefore, when the algorithm terminates, $\mathcal{G}^P$ is a tree.

Next we show that when *subtree_balanced*(u) is set to TRUE, the subtree rooted at u in $\mathcal{G}^P$ is a subtree balanced tree, based on the definition stated above. We prove this by induction. Let us assume that the statement holds for each child vertex v of u. Now note that when *subtree_balanced*(u) is set to TRUE, *subtree_balanced*(v) must be TRUE for each child vertex v. Therefore, by our induction assumption, the subtree rooted at any child vertex v must be a subtree balanced tree at that stage. Now consider two possibilities: (i) $|\mathcal{N}_u| = 1$, and (ii) $|\mathcal{N}_u| \geq 2$.

In case (i), vertex v is merged with u. Therefore, after this merging, there exists a binary balanced tree rooted at u (the same binary balanced tree that was earlier rooted at v), of depth d_v. Since $|\mathcal{N}_u| = 1$, all other subtrees rooted at u of depth must have depth no more than $d_v - 1 + 1 = d_v$. Therefore, the binary balanced tree rooted at u also has the maximum depth amongst all subtrees rooted at u. Therefore, the subtree rooted at u is subtree balanced tree.

In case (ii), there are at least two children vertices v and v' of u that have a binary balanced subtree rooted at those vertices with a depth of $d_u - 1$; let these binary balanced subtrees be $\mathcal{T}$ and $\mathcal{T}'$, respectively. Now note that u, along with $\mathcal{T}$ and $\mathcal{T}'$, constitute a binary balanced tree of depth d_u. Since no merging is done in this case, the subtree rooted at u is indeed a subtree balanced tree.

To argue the base case, note that the leaf vertices are subtree balanced trees (leaf vertices, which have no children, are binary balanced trees, trivially). This completes the induction argument, and proves that on termination of the PATH TREE COMPRESSION ALGORITHM, $\mathcal{G}^P$ is a subtree balanced tree rooted at r.

We complete the proof by arguing that the PATH TREE COMPRESSION ALGORITHM terminates. For any vertex u, let $\delta(u)$ denote the shortest distance of the vertex from root r in the initial tree $\mathcal{G}^P$; let D_0 be the depth of the initial tree $\mathcal{G}^P$. Then, it is straightforward to see that at the end of the kth iteration of the while loop, all vertices u for which $\delta(u) = D_0 - k$, must have *subtree_balanced*(u) set to TRUE. Therefore, the algorithm must terminate after at most $D_0 \leq |\mathcal{N}|$ iterations of the while loop. This completes the proof. ■

Corollary 1: When the PATH TREE COMPRESSION ALGORITHM terminates, $\mathcal{G}^P$ is a tree with depth $O(\log n)$, and satisfies properties P.1-P.3.

Proof: From Lemma 1, when the PATH TREE COMPRESSION ALGORITHM terminates, $\mathcal{G}^P$ is a tree and contains a binary balanced tree rooted at r, whose depth is the same as that of $\mathcal{G}^P$. Let this binary balanced tree be $\mathcal{T}'$, and the number of vertices in $\mathcal{T}'$ be denoted by n'. Note that n' can be no greater than the number of vertices in $\mathcal{G}^P$, which is upper bounded by the number of nodes in $\mathcal{G}$. Thus, if d' denotes the depth of $\mathcal{G}^P$, then $d' = \log_2(n'+1) - 1 \leq \log_2(|\mathcal{N}|+1) - 1$, which is $O(\log n)$. From the construction of $\mathcal{G}^P$, it is easy to verify that it satisfies P.1-P.3 initially, as well as after every step of the iterative PATH TREE COMPRESSION ALGORITHM. The result follows. ■

3) Scheduling Algorithm: Each path represented by the vertices in the path tree graph $\mathcal{G}^P$, the output of the PATH TREE COMPRESSION ALGORITHM 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 full details of the algorithm below.

Let paths $\{\mathcal{H}_k\}$ be the output of the PATH TREE COMPRESSION ALGORITHM. If u is the parent of v in $\mathcal{G}$ 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 $\mathcal{H}_v$. Due to the tree properties $P.1$ to $P.3$, 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 node in $\mathcal{G}$, and the corresponding paths in different partitions are not adjacent. Given the degree bound, each partition consists of at most Δ nodes in T, 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 $\mathcal{G}^P$; the breadth first traversal visits an older sibling before a younger sibling. Let p_i be the level of node i in the breadth first traversal tree $\mathcal{T}$ 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 Corollary 1, $\hat{p}$ is $O(\log n)$.

Recall that maximal scheduling is implemented using a distributed randomized algorithm like the one proposed in [5]. The algorithm operates in rounds, and the time required to complete each round is constant in degree-bounded graphs [5], [6]. 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 maximal scheduling. 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 [5], it can be easily shown that $\gamma < \frac{3}{16}$.

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. Links in $\mathcal{H}_i$ start executing their scheduling phase, the Sequential Maximal Tree Scheduling routine (Figure 6), after $(p_i + r_i)\left(\lfloor \ln(36\Delta) / (-\ln(\gamma)) \rfloor + 1\right)$ time units where each unit corresponds to the time required to complete one round of the maximal scheduling policy. As the scheduling algorithm progresses, these un-decided links change their status to *scheduled* or *un-scheduled*.

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 which 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 the path 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.

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*).

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"; links that are adjacent to scheduled links set their status to "un-scheduled".

Fig. 6. Sequential Maximal Tree Scheduling Algorithm for $\mathcal{H}_i$

Finally, we evaluate the time required for the schedule computation. Since $r_i \leq d-1$, and $p_i \leq \hat{p}$ which is $O(\log n)$ and the expected time required for computing a maximal schedule is $O(\log n)$, the scheduling for the entire tree can be computed in $O(\log^2 n)$ expected time.

C. 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 [5], for example). Note that it is not necessary for each path to wait for each of its predecessor paths in the path tree graph to finish its scheduling before starting to execute its own scheduling phase. Each path only waits for a time that is large enough for its predecessor paths to finish their scheduling phase with a high probability.

Nodes exchange messages only with their neighbors, and the overall per-node message complexity is logarithmic in the size of the network. The path compression 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.

We have assumed throughout the paper that the maximum degree of any node is upper-bounded by a constant. In our algorithm, each node needs to periodically exchange information with all of its neighbors; the computation time for this message exchange will typically depend on the number of neighbors a node has. This is true even for the simple maximal scheduling policy considered in [4], [1]: the overall computation time is logarithmic only when message exchanges between a node and all of its neighbors have constant time complexity. However, if this can somehow be achieved, i.e., local message exchanges with all nodes in the neighborhood can be done in constant time irrespective of

node degree, then our Sequential Maximal Tree Scheduling Algorithm can be generalized to guarantee the same throughput (i.e., 2/3 of the maximum throughput region) within $O(\log^2 n)$ time even for trees that are not degree-bounded.

Finally, note that the framework we proposed involves decomposition of trees in paths and scheduling links in each path using a scheduling 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 logarithmic computation time, then we can use this framework to obtain the same guarantees for trees while requiring an overall computation time of $O(\log^2 n)$.

IV. Proof for $2/3$ throughput guarantee

We assume that $\mathcal{G}$ is a tree. If an arrival rate vector $\vec{\rho} \in \Lambda^*$, then (a) $\rho_l \geq 0 \ \forall \ l \in \mathcal{L}$ and (b) $\forall \ u \in \mathcal{N}, \ \sum_{l \in \mathcal{L} \cap \{(u,v),(v,u)\}} \rho_l \leq 1$ [3]. We consider an arrival rate vector $\vec{\rho}$ such that

$$\forall \ u \in \mathcal{N}, \quad \sum_{l \in \mathcal{L} \cap \{(u,v),(v,u), v \in \mathcal{N}\}} \rho_l \ < \ 2/3. \qquad (2)$$

$$\text{Let } \xi \ = \ 2/3 - \max_{u \in \mathcal{N}} \sum_{l \in \mathcal{L} \cap \{(u,v),(v,u), v \in \mathcal{N}\}} \rho_l.$$

The L.H.S. of (2) is the total packet arrival rate over all links that originate or terminate at node 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). We show that there exists a constant B_0 such that for any $t > 0$, $\mathbf{E}Q_l(t) \leq B_0$, $\forall \ l \in \mathcal{L}$ (Theorem 1). Thus, our scheduling policy guarantees a $\frac{2}{3}$ fraction of the maximum throughput region.

We subsequently state and prove supporting lemmas, lemmas 2 to 7 (subsection IV-A). We prove the main result, Theorem 1, using these lemmas (subsection IV-B).

A. Supporting Lemmas

We present a series of lemmas, lemmas 2 to 7 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 3,5,6,7 will be used in proving Theorem 1. Lemma 2 will be used in proving lemma 3, and lemma 4 will be used in proving lemmas 5 and 6.

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 IV-B, using induction, we prove that every path $\mathcal{H}$ satisfies the constraint-lower-bound, and subsequently prove the throughput guarantee using lemmas 3,5,6 - the last two of these lemmas hold only when the constraint-lower-bound holds.

For simplicity, we assume that $Q_l(0) = 0$ for all $l \in \mathcal{L}$; the proofs can be generalized for any positive, but finite values of $\vec{Q}(0)$. 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 $\mathcal{P}$.

Lemma 2: 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.

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 part of the above lemma does not hold if the iterative step of the Sequential Maximal Tree Scheduling has only one phase.

Lemma 3: Let κ and B be positive integers such that $B \geq 5\kappa+1$. Consider a path $\mathcal{P} \subseteq \mathcal{H} \backslash \{l_{\mathcal{H},1}\}$ 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) $Q_{l_i}(t) \geq B - \kappa \quad \forall\ i \in \{1, \ldots, m\}$ (*lower bound criterion*)
2) $Q_{l_i}(t') \leq B - 1 \quad \forall\ i \in \{1, \ldots, m\}$ and $\forall\ t' \leq t$ (*upper bound criterion*), and
3) either $Q_l(t') < B - 5\kappa \ \forall\ l \in \mathcal{C}_{\mathcal{P}} \cap \mathcal{H}$, or $\min_{i \in \{1,\ldots,m\}} Q_{l_i}(t') < B - 5\kappa \ \forall\ t' < t$ (*boundary condition*).

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

Lemma 3 shows that if links in a segment of a path of length 5 or less have high queue lengths, then with a high probability, at least one link that is not in the path but is adjacent to a link in the path has high queue length as well. Lemma 3 applies only for segments that do not contain the first link of the path.

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

$$Q_{l_i}(t') \geq B - 5\kappa \ \forall\ \ i \in \{1, \ldots, m\} \text{ and } t' \in [t_2, t], \quad (3)$$

$$\text{and } Q_{l_i}(t_2) = B - 5\kappa \text{ for some } i \in \{1, \ldots, m\}. \quad (4)$$

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

$$Q_l(t') < B - 5\kappa \ \forall\ l \in \mathcal{C}_{\mathcal{P}},\ \forall\ t' \in [t_2, t). \quad (5)$$

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 (1), $\mathbf{P}(\mathcal{B}_i) < \chi_{\xi/3}(\frac{\max_{l \in \mathcal{L}} \sigma_l}{4\kappa})^{\alpha}$. 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 (4) and the lower and upper bound criteria,

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

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

where $S_{l_i}(t_2, t)$ denotes 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 (5), (3) and lemma 2, at least 1 link in $\mathcal{P}$ is scheduled in each slot in $[t_2, t)$. Thus, from (8),

$$\begin{aligned} \sum_{i=1}^{m} (A_{l_i}(t_2, t) - \rho_{l_i}(t - t_2)) &\geq (1 - \sum_{i=1}^{m} \rho_{l_i})(t - t_2) \\ &\geq \xi(t - t_2) \text{(from (2))}. \quad (9) \end{aligned}$$

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 (5), (3) and lemma 2, 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 (2), $\sum_{i=1}^{m} \rho_{l_i} \leq 3 \times (2/3 - \xi) = 2 - 3\xi$. 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 (4) 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,

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

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 (5), (3) and lemma 2, either l_2 or both l_1 and l_3 are scheduled in each slot in $[t_2, t)$. Thus,

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

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

$$\leq \sum_{i=1}^{2}\left(A_{l_i}(t_2,t)-\rho_{l_i}(t-t_2)\right)+\sum_{i=2}^{3}\left(A_{l_i}(t_2,t) -\rho_{l_i}(t-t_2)\right)-(2-4/3+2\xi)\,(t-t_2) \text{ (from (2))}$$

$$\leq \sum_{i=1}^{2}\left(A_{l_i}(t_2,t)-\rho_{l_i}(t-t_2)\right)+\sum_{i=2}^{3}\left(A_{l_i}(t_2,t) -\rho_{l_i}(t-t_2)\right)-2\xi(t-t_2).$$

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

Lemma 4: 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 the following criteria:

1) $\min\left(Q_{l_{\mathcal{H},i}}(t), Q_{l_{\mathcal{H},i+1}}(t)\right)>0$ and
2) if $i+2\leq|\mathcal{H}|$, $Q_{l_{\mathcal{H},i+1}}(t)>Q_{l_{\mathcal{H},i+2}}(t)$.

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 2. The result follows from the case with $|\mathcal{H}|=2$ in lemma 2.

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 2. 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 2. 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 2. 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 2 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 $3\leq i+1\leq 4$. Again, $\mathcal{P}$ satisfies the conditions of lemma 2. 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 2 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 $|\mathcal{H}|\geq 6$) 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 4 does not hold if the iterative step of the Sequential Maximal Tree Scheduling has two or fewer phases.

Lemma 5: 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 $\mathcal{H}$, $1\leq i\leq 5$, and an event $\mathcal{A}$ that occurs if and only if there exists a slot t such that

1) $Q_{l_{\mathcal{H},i}}(t)\geq B-\beta$,
2) $\max_{l\in\mathcal{N}_{l_{\mathcal{H},i}}\cap\mathcal{H}} Q_l(t')\leq B-1$ 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 $\mathbf{P}(\mathcal{A})\leq(2\chi_{1/6}+\gamma_{\mathcal{H}})(\max_{l\in\mathcal{L}}\sigma_l/\beta)^{\alpha}$.

Lemma 5 shows that if in a path, a link l which is at most 5 hops away from the first link in the path has a high queue length, then with a high probability, another link in the path, say l', which is adjacent to l, but farther off from the first link has high queue length as well. Thus, if the first link in a path is congested, then with a high probability congestion spreads into the path.

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 $Q_{l_{\mathcal{H},i}}(t')\geq B-4\beta$ 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, $t-t_2\geq 3\beta/\max_{l\in\mathcal{L}}\sigma_l$.

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, $A_{l_{\mathcal{H},1}}(t_2,t)\geq S_{l_{\mathcal{H},1}}(t_2,t)=\Theta_{\mathcal{H}}(t_2,t)$. This implies that either $A_{l_{\mathcal{H},1}}(t_2,t)\geq(\rho_{l_{\mathcal{H},1}}+1/6)(t-t_2)$ or $\Theta_{\mathcal{H}}(t_2,t)\leq(\rho_{l_{\mathcal{H},1}}+1/6)(t-t_2)$. From (1), 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, $\mathbf{P}(\mathcal{A})\leq(\chi_{1/6}+\gamma_{\mathcal{H}})(t-t_2)^{-\alpha}$. The lemma follows for $i=1$ since $t-t_2\geq 3\beta/\max_{l\in\mathcal{L}}\sigma_l$.

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 $Q_{l_{\mathcal{H},i}}(t')\geq B-2\beta$ for all $t'\in[t_3,t]$. Clearly, $t-t_3\geq\beta/\max_{l\in\mathcal{L}}\sigma_l$. 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, $A_{l_{\mathcal{H},i}}(t_3,t)\geq S_{l_{\mathcal{H},i}}(t_3,t)=t-t_3$. From (1) and (2), $\mathbf{P}\left\{A_{l_{\mathcal{H},i}}(t_3,t)\geq t-t_3\right\}<\chi_{1/3}(t-t_3)^{-\alpha}$. Thus, $\mathbf{P}(\mathcal{A}\cap\mathcal{B})<\chi_{1/3}(t-t_3)^{-\alpha}\leq\chi_{1/3}(\max_{l\in\mathcal{L}}\sigma_l/\beta)^{\alpha}$.

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

$$\sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}} Q_l(t_4)\geq 2B-4\beta. \tag{15}$$

From the definition of t_2, there also exists a slot $t_5\in[t_2,t_4]$ such that $Q_l(t')\geq B-4\beta$ 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, $t_4-t_5\geq 2\beta/\max_{l\in\mathcal{L}}\sigma_l$. Since $4\beta<B$, and $1\leq i-1\leq 4$, in both cases (a) and (b), from lemma 4,

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,

$$\begin{aligned}
&\sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}} A_l(t_5,t_4)\\
&= \sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}} Q_l(t_4) - \sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}} Q_l(t_5)\\
&\quad + \sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}} S_l(t_5,t_4) \qquad (16)\\
&\geq (2B-4\beta) - (B-4\beta) - (B-1) + t_4 - t_5 \qquad (17)\\
&\geq \sum_{l\in\{l_{\mathcal{H},i-1},l_{\mathcal{H},i}\}} (\rho_l + 1/6)(t_4-t_5) \text{ (from (2))}. \qquad (18)
\end{aligned}$$

Note that (17) above follows from (15) 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 $A_{l_{\mathcal{H},i-1}}(t_5,t_4) \geq (\rho_{l_{\mathcal{H},i-1}} + 1/6)(t_4-t_5)$, or $A_{l_{\mathcal{H},i}}(t_5,t_4) \geq (\rho_{l_{\mathcal{H},i}} + 1/6)(t_4-t_5)$. From (1), 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 6: Let $\mathcal{H}$ be a path in $\{\mathcal{H}_k\}$ that satisfies the constraint-lower-bound. Consider an integer $B \geq 6$ 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

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

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

Lemma 6 shows that the probability that all links in a segment of a path consisting of 6 links has high queue lengths is small.

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 $l_{\mathcal{H},2}\ldots l_{\mathcal{H},6}$.

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

$$\begin{aligned}
&\sum_{i=j}^{j+5}\left(Q_{l_{\mathcal{H},i}}(t) - Q_{l_{\mathcal{H},i}}(t')\right)\\
&= \left(Q_{l_{\mathcal{H},k}}(t) - Q_{l_{\mathcal{H},k}}(t')\right) + \sum_{\substack{j\leq i\leq j+5\\ i\neq k}}\left(Q_{l_{\mathcal{H},i}}(t) - Q_{l_{\mathcal{H},i}}(t')\right)\\
&\geq 5B/6 + \sum_{\substack{j\leq i\leq j+5\\ i\neq k}}(5B/6 - B + 1) \ \geq\ 5. \qquad (19)
\end{aligned}$$

$$\begin{aligned}
&\text{Also, } \sum_{i=j}^{j+5}\left(Q_{l_{\mathcal{H},i}}(t) - Q_{l_{\mathcal{H},i}}(t')\right)\\
&= \sum_{i=j}^{j+5} A_{l_{\mathcal{H},i}}(t',t) - \sum_{i=j}^{j+5} S_{l_{\mathcal{H},i}}(t',t), \qquad (20)\\
&\text{and, } \sum_{i=j}^{j+5}\left(A_{l_{\mathcal{H},i}}(t',t) - \rho_{l_{\mathcal{H},i}}(t-t')\right)\\
&\geq \sum_{i=j}^{j+5} S_{l_{\mathcal{H},i}}(t',t) - \sum_{i=j}^{j+5}\rho_{l_{\mathcal{H},i}}(t-t') + 5 \text{ (from (19),(20))}\\
&\geq \sum_{i=j}^{j+5} S_{l_i}(t',t) - 3\left((2/3)-\xi\right)(t-t') + 5 \text{ (from (2))}.
\end{aligned}$$

Next, $Q_{l_{\mathcal{H},i}}(t_1) > 0 \ \ \forall\ i\in\{j,\ldots,j+5\},\ \ \forall\ t_1\in(t',t]$. The set of links scheduled at each slot constitutes a maximal scheduling among those that have positive queue lengths in the slot. Thus, at least two links in $\mathcal{P}$ are scheduled in every slot in $(t',t]$. Thus, $\sum_{i=j}^{j+5} S_{l_{\mathcal{H},i}}(t',t) \geq 2(t-t') - 2$. Thus,

$$\sum_{i=j}^{j+5}\left(A_{l_{\mathcal{H},i}}(t',t) - \rho_{l_{\mathcal{H},i}}(t-t')\right) \geq 3\xi(t-t').$$

Thus, for some $i\in\{j,\ldots,j+5\}$, $\mathcal{B}_i$ occurs. For the case $m=6$, the result follows.

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

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

$$\sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} Q_l(t) \geq 5B/3. \qquad (21)$$

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, $t - t_1 \geq 5B/6\max_{l\in\mathcal{L}}\sigma_l$. From lemma 4, either $l_{\mathcal{H},m-1}$ or $l_{\mathcal{H},m}$ is served in each slot in $(t_1,t]$. Thus, $\sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} S_l(t_1,t) \geq t - t_1 - 1$. Now,

$$\begin{aligned}
&\sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} A_l(t_1,t)\\
&= \sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} Q_l(t) - \sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} Q_l(t_1)\\
&\quad + \sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} S_l(t_1,t) \qquad (22)\\
&\geq 5B/3 - (B-1) + (t-t_1) - 1 \qquad (23)\\
&\geq \sum_{l\in\{l_{\mathcal{H},m-1},l_{\mathcal{H},m}\}} (\rho_l + 1/6)(t-t_1) \text{(from (2))}. \qquad (24)
\end{aligned}$$

Note that (23) above follows from (21) 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 (1), 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 (1), 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 $\mathcal{G}$. Let $\mathcal{F}_{\mathcal{H}} = \{v : v \text{ is either the parent or an older sibling of } u \text{ in } \mathcal{G}\}$. Let $d = \Delta + 1$.

Lemma 7: 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 $\gamma_{\mathcal{H}} = (d-1)\chi_{1/18(d-1)} + (d-1)\nu_{d,\gamma} + 18^{\alpha}(d-1)^{\alpha+1}\max_{i\in\mathcal{F}_{\mathcal{H}}}\mu_{\mathcal{H}_i}$.

Lemma 7 shows that a path satisfies the constraint-lower-bound if the probability that the links in the parent and older siblings of a path have high queue lengths is low.

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 $W = \lfloor \ln(36(d-1))/(-\ln(\gamma)) \rfloor$. 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 $1/(36(d-1)$. Thus,

$$\begin{aligned}
&\mathbf{P}\left\{U_l(t_1,t_2) \geq \frac{t_2-t_1}{18(d-1)}\right\} \\
\leq\ & \left(\frac{\exp\left((1/(18\gamma^W(d-1))) - 1\right)}{(1/(18\gamma^W(d-1)))^{(1/(18\gamma^W(d-1)))}}\right)^{\beta} \\
& \text{where } \beta = \gamma^W(t_2-t_1) \qquad (25)\\
\leq\ & \nu_{d,\gamma}(t_2-t_1)^{-\alpha} \text{ for some constant } \nu_{d,\gamma} \text{ that depends} \\
& \text{on } d, \gamma,\ \forall\ l \in \mathcal{H}_i \cap \mathcal{N}_{l_{\mathcal{H},1}}. \qquad (26)
\end{aligned}$$

In the above, (25) follows from the Chernoff bound.

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,

$$\begin{aligned}
&\sum_{l\in\cup_{i\in\mathcal{F}_{\mathcal{H}}}(\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}})}(S_l(t_1,t_2) + U_l(t_1,t_2)) \\
\geq\ & (5/6 - \rho_{l_{\mathcal{H},1}})(t_2-t_1) \\
\geq\ & (1/6 + \sum_{l\in\cup_{i\in\mathcal{F}_{\mathcal{H}}}(\mathcal{H}_i\cap\mathcal{N}_{l_{\mathcal{H},1}})}\rho_l)(t_2-t_1) \qquad (27)
\end{aligned}$$

The last inequality follows from (2) 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 G.

Now, $S_l(t_1,t_2) \leq Q_l(t_1) + A_l(t_1,t_2)$. Thus, from (27), and since $|\cup_{i\in\mathcal{F}_{\mathcal{H}}}(\mathcal{H}_i \cap \mathcal{N}_{l_{\mathcal{H},1}})| \leq d-1$,

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

Thus, either $A_l(t_1,t_2) \geq (1/18(d-1) + \rho_l)(t_2-t_1)$ or $Q_l(t_1) \geq (t_2-t_1)/18(d-1)$ or $U_l(t_1,t_2) \geq (t_2-t_1)/18(d-1)$ 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 $Q_l(t_1) \geq (t_2-t_1)/18(d-1)$ is at most $\mu_{\mathcal{H}_i}(\frac{t_2-t_1}{18(d-1)})^{-\alpha}$ if $l \in H_i$ and $i \in \mathcal{F}_{\mathcal{H}}$. The result follows from (1) and (26). ■

B. Main Result

Theorem 1: 1) For each $t > 0$, $l \in \mathcal{L}$, $\mathbf{P}\{Q_l(t) \geq B\} \leq \tau_{\hat{p},d-2}B^{-\alpha}$, where $\tau_{\hat{p},d-2}$ is obtained through the following recursions.

$$\gamma_{0,y} = 0\ \forall\ 0 \leq y \leq d-2 \qquad (28)$$

$$\begin{aligned}
\tau_{x,y} =\ & \left(\max(1, 151\chi_{\min(\xi/3,1/6)} + 11\gamma_{x,y})\right) \\
& \times(72\times 4^5\times 5^5\max_{l\in\mathcal{L}}\sigma_l)^{\alpha}\ 0 \leq x \leq \hat{p}, \\
& 0 \leq y \leq d-2 \qquad (29)
\end{aligned}$$

$$\begin{aligned}
\gamma_{x,y} =\ & (d-1)\left(\nu_{d,\gamma} + \chi_{1/18(d-1)}\right) + 18^{\alpha}(d-1)^{\alpha+1} \\
& \times\max(\tau_{x-1,d-2}, \max_{0\leq z\leq y-1}\tau_{x,z}), \\
& 0 \leq x \leq \hat{p}, 0 \leq y \leq d-2. \qquad (30)
\end{aligned}$$

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

We first outline the main arguments in the proof. First consider the root path in $\mathcal{G}^P$. This path does not have a parent or an older sibling, and hence does not have constrained slots. Thus, the constraint-lower-bound trivially holds for this path. Hence, lemmas 5 and 6 hold for this path. It follows from lemma 5 that if in this path a link has a high queue length in any slot, then with a high probability a link which is 6 hops or farther away from the first link in the path has a high queue length in the same slot. Now, it follows from lemma 3 that with a high probability all links in a segment consisting of 6 links in the path have high queue lengths in the slot. This contradicts lemma 6. Thus, the probability that a link in this path has high queue lengths is low. It now follows from lemma 7 that the constraint-lower-bound holds for the children paths of the root path. Using a recursive argument, it can now be shown that the probability that a link in the descendant paths has high queue lengths is low as well. The result follows.

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

$$\mathbf{P}\{Q_l(t) \geq B\} \leq \tau_{p_j,r_j}B^{-\alpha}, \qquad (31)$$

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 $r_i \leq d-2$ 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, $\tau_{0,x} \geq \left(12 \times 4^5 \times 5^5 \max_{l \in \mathcal{L}} \sigma_l\right)^\alpha$, (31) trivially holds for $B < 12 \times 4^5 \times 5^5 \max_{l \in \mathcal{L}} \sigma_l$. Let $B \geq 12 \times 4^5 \times 5^5 \max_{l \in \mathcal{L}} \sigma_l$. Now, j is the root of T 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 (2), $\mathcal{H}_j$ 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 $0 \leq q \leq 5-k$, if $|\mathcal{H}_j| > k+q$, event $\mathcal{B}_q$ is said to occur if the event $\mathcal{A}$ described in lemma 5 occurs with $i = k+q$ and $\beta = 4^q \lfloor B/(6 \times 4^5 \times 5^5) \rfloor$. If $|\mathcal{H}| \geq 6$, for $1 \leq c \leq 6$, $1 \leq d \leq c$, 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 $c \leq 5$ $l_{\mathcal{H}_1,1} \notin \mathcal{P}_{c,d}$. 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 3 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 $\mathcal{C}_{6,d}$ is said to occur if the event $\mathcal{A}$ described in lemma 6 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, $0 \leq q \leq 5-j, 1 \leq c \leq 6, 1 \leq d \leq c$. Thus, $\mathbf{P}\left(\mathcal{A}(B,t)\right)$ is upper bounded by the sum of the probabilities of the events $\mathcal{B}_q$, $\mathcal{C}_{c,d}$ for $0 \leq q \leq 5-j, 1 \leq c \leq 6, 1 \leq d \leq c$. Thus (31) follows from the upper bounds of the probabilities of these events provided in lemmas 3, 5, 6.

We now consider the induction case. Now, let (31) 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 T. Since i's parent's level is h, i's parent satisfies (31). Now, lemma 7 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 (31) holds for i. Now, let (31) 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 (31). Again, lemma 7 shows that $\mathcal{H}_i$ satisfies the constraint-lower-bound with $\gamma_{\mathcal{H}_i} = \gamma_{h+1,a+1}$. Thus, as before, (31) holds for i.

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

■

REFERENCES

[1] 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.

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

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

[4] 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.

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

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.