On Optimal Dynamic Scheduling for Sum-Queue Minimization in Trees

Srikanth Hariharan
Department of ECE
The Ohio State University
Columbus, OH - 43210
Email: harihars@ece.osu.edu

Ness B. Shroff
Departments of ECE and CSE
The Ohio State University
Columbus, OH - 43210
Email: shroff@ece.osu.edu

***Abstract*—We investigate the problem of minimizing the sum of the queues of all the nodes in a wireless network with a tree topology. Nodes send their packets to the tree's root (sink). We consider a time-slotted system, and a primary interference model. We first consider the case where the root has only one child while the rest of the tree is arbitrary, and provide a *causal sample-path delay optimal scheduling policy*, i.e., at each time slot, for any traffic arrival pattern, the sum of the queues of all the nodes is minimum among *all* policies. *We are able to fully characterize tree structures for which such policies exist*. In particular, when the root has multiple children, there exists a causal sample-path delay optimal policy as long as only one child is not a leaf node. We also show that for any other tree structure there exists no causal sample-path delay optimal policy, thus underscoring the inherent limitation of using sample-path optimality as a performance metric and implying that other weaker metrics of delay performance should be investigated.**

I. Introduction

We investigate the problem of finding sample-path optimal scheduling policies for minimizing the sum of the queues of all the nodes for *convergecasting* [1] in a wireless network with a tree topology. In the convergecasting problem, nodes send their packets to a sink (which is the root of the tree). Convergecasting has a number of applications, for instance, in wireless sensor networks, where the sink desires to obtain all the sensor measurements, and take a decision based on these measurements. Since the sink typically requires to receive these packets in a timely manner, and since interference is a critical aspect of wireless networks, it is important to find a transmission schedule such that packets reach the sink in minimum time. We are interested in minimizing the sum of queues in the system as it can be shown to minimize the delay experienced by packets in the system.

Delay efficient convergecasting has been well studied in the scheduling literature. Tassiulas et. al., [2] first studied the problem of dynamic scheduling for convergecasting in tandem networks with the sink at the root of the chain. They showed that for the primary (or one-hop) interference model (where two links that share a node cannot be active at the same time), for any traffic arrival pattern, any maximal matching policy that gives priority to the link closer to the sink is optimal in the sense that the sum of the queues of all the nodes in the network is minimum at each time slot. This is a very strong result because for any sample-path arrival pattern, this policy is optimal. Also, it is causal as it does not require future knowledge of arrivals. Ji et. al., [3] consider small generalized switches with less than or equal to four links. They develop a sample-path optimal policy for switches with three links, and a heavy-traffic optimal policy for switches with four links. In [4], Gupta et. al., have provided a sample-path delay optimal policy for a clique wireless network where only one link can transmit at any time, and there are multi-hop flows. Venkataramanan et. al., [1] have shown that the policy of giving priority to links closer to the sink is optimal in the large deviations sense, i.e., the rate of decay of the probability that the sum of all the queues $> B$ as $B \to \infty$ is maximum, even in a general tree topology.

Apart from the literature considering traffic arrivals, there exist a number of works studying the convergecasting problem in the absence of arrivals. Florens et. al., [5] have studied the problem of minimizing the time by which all the packets in a network (with a tree topology) reach the sink, for an one-hop interference model. They propose polynomial time algorithms for this problem. Bermond et. al., [6] and Gargano et. al., [7] have further studied this problem for disk based communication model, and arbitrary network topologies respectively.

In this work, we wish to study the problem of convergecasting for wireless networks with a tree topology for arbitrary traffic arrival patterns. Specifically, we are interested in being able to characterize all possible tree structures for which sample path delay-optimal policies exist.

Our contributions in this work are the following.

- For tree networks where the root of the tree has only one child while the rest of the tree is arbitrary, we show that the policy of giving priority to links closer to the sink minimizes the sum of the queues of all the nodes in the network for every time slot, and for any traffic arrival pattern.
- Further, we provide a sample-path delay optimal scheduling policy for tree structures, where all but one of the root's children is not a leaf node.
- Surprisingly, we show that for all other tree structures,

This work was supported in part by ARO MURI Awards W911NF-07-10376 (SA08-03) and W911NF-08-1-0238, and NSF Awards 1012700-CNS, 0721236-CNS, and 0721434-CNS.

there exists a traffic arrival pattern such that without having future knowledge of the arrival pattern, there exists no sample-path delay optimal policy. *Thus, we completely classify tree structures for which there exists a causal sample-path delay optimal policy, and for which there does not*.

The rest of this paper is organized as follows. In Section II, we describe the model and notations. In Section III, we discuss our result for trees with the root having exactly one child. In Section IV, we develop a sample-path delay optimal scheduling policy for trees where all but one of the root's children is not a leaf node. Further, we show that there does not exist any sample-path delay optimal scheduling policy for all other tree structures. We discuss various metrics of delay optimality for tree structures such as evacuation-time optimality, and delay optimality from a large deviations perspective. Finally, we conclude the paper in Section V.

II. System Model

We model the network as a graph $G(V, E)$, where V is the set of nodes and E is the set of links. The graph G is a tree. We denote 0 to be the sink which is the root of the tree. The sink does not make any transmissions. We assume a time-slotted and synchronized system. We consider a one-hop (or node exclusive or primary) interference model where two links that share a node cannot be active at the same time. As in [2], [5], we assume unit capacity links, i.e., a node can at most transmit one packet to its parent during each time slot. The external packet arrival pattern at nodes is arbitrary and unknown. All packets in the network have the sink 0 as the eventual destination.

III. Trees - Degree of Sink is One

In this section, we develop a sample-path optimal scheduling policy for tree networks where the degree of the sink 0 is one, i.e., the sink has exactly one child. We first define an *evacuation time optimal* policy.

Evacuation Time Optimal Policy: Consider a network where each node has a fixed number of packets destined to given destinations. An evacuation time optimal policy is a scheduling policy such that in the absence of further arrivals, the time by which the system is evacuated (i.e., the last packet reaches its destination) is minimum.

In [5], an evacuation time optimal policy has been proposed for trees where the degree of the sink is one. We show that this evacuation time optimal policy is actually *sample-path delay optimal* (i.e., at each time slot, for any traffic arrival pattern, the sum of the queues of all the nodes in the network is minimum among all policies) for these networks.

A. Sample-Path Optimal Policy

As defined in [5], we first define an equivalent linear network. For a tree network (with sink degree one) $G(V, E)$ with V nodes and E edges, where each node i has β_i packets during a given time slot, the equivalent linear network $G(V_l, E_l)$ is defined as follows: $V_l = \{0, 1, ..., N\}$, $E_l = \{(i-1, i), 1 \leq i \leq N\}$ where $N = \max_{i \in V}(d(0, i))$. $d(0, i)$ represents the distance of node i from the sink node 0. Further, each node $j \in V_l$ has α_j packets during the same time slot, where $\alpha_j = \sum_{i \in V : d(0,i) = j} \beta_i$.

Figure 1 gives an example of this transformation. The sink is denoted by the filled circle. The farthest node in the tree is 3 hops away from the sink. Therefore, the equivalent linear network has 3 nodes and the sink. The number of packets at each node is mentioned in the figure. The total number of packets from nodes that are 2 hops away from the sink is 7 (=3+4), and that from nodes that are 3 hops away from the sink is 9 (=6+1+2). Therefore, the equivalent linear network has 7 packets in Q, and 9 packets in R.

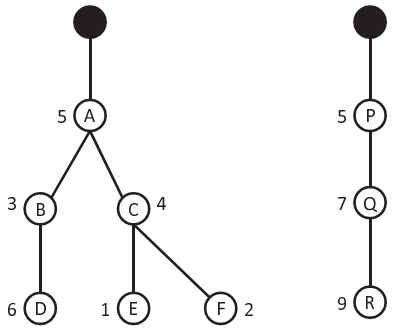

Fig. 1. Equivalent Linear Network

We show here that the sample-path optimal policy for this class of tree networks is simply to schedule packets in the equivalent linear network according to the sample-path optimal policy for linear networks defined in [2]. In [5], this policy is shown to be evacuation time optimal for this class of trees. For the reader's convenience, we provide the sample-path optimal policy for linear networks below. We also explain how to convert the schedule for the equivalent linear network into a schedule for the original tree.

Consider a tandem (linear) network consisting of $N + 1$ nodes indexed from 0 to N. Node 0 is the root of the tandem network. We have the following notations and definitions (Table I).

For the convergecasting problem, the queue length vector evolves as $\boldsymbol{X}(t+1) = \boldsymbol{X}(t) + \boldsymbol{R}\boldsymbol{I}(t+1) + \boldsymbol{A}(t+1)$, where $\boldsymbol{R}$ is an $N \times N$ matrix with elements

$$r_{ij} = \begin{cases} 1, & j = i+1 \\ -1, & i = j \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Note that the above equation is for the equivalent linear network. For the original tree, r_{ij} is the same except that "$j = i + 1$" is replaced by "j is a child of i".

Definition: We now define the stationary policy π_0 which, at slot t, selects the activation vector $\boldsymbol{I}(t) = g_0(\boldsymbol{X}(t-1))$, where $g_0 : \mathbb{Z}_+^N \to S$ is defined as follows. Let $\boldsymbol{i} = g_0(\boldsymbol{x})$ and i_j, x_j be the j^{th} elements of vectors $\boldsymbol{i}$ and $\boldsymbol{x}$ respectively. The vector

TABLE I
DEFINITIONS

1	Activation Set: A set of links that can be simultaneously activated such that no two links interfere with each other according to the one-hop interference model.
2	Activation Vector: An N-dimensional binary indicator vector $\boldsymbol{i}$ with one element for each link (which is not zero if and only if the link belongs to the activation set).
3	S: Set of all possible activation vectors.
4	$A_i(t)$: Set of exogenous packet arrivals to node i at slot t.
5	$\boldsymbol{A}(t)$: $\boldsymbol{A}(t) = (A_i(t), i = 1, ..., N)$ is the vector of arrivals at all nodes during slot t.
6	$X_i(t)$: Length of the queue of packets at node i by the end of slot t. $X_i(t) \geq 0 \ \forall i \in \{1, 2, ..., N\}$.
7	$\boldsymbol{X}(t)$: $\boldsymbol{X}(t) = (X_i(t), i = 1, ..., N)$ is the vector of queue lengths at all nodes at the end of slot t.
8	$\boldsymbol{X}$: The queue length process $\{X(t)\}_{t=1}^{\infty}$.
9	$\boldsymbol{I}(t)$: Indicator vector denoting the set of links activated at time slot t. A link is activated only if the corresponding node has packets to send.

$\boldsymbol{i}$ is defined recursively as follows. $i_1 = 1$, if $x_1 > 0$, and 0, otherwise. For $j = 2, ..., N$, $i_j = 1$, if $x_j > 0$ and $i_{j-1} = 0$. Otherwise, $i_j = 0$.

It has been shown in [2] that π_0 is a sample-path delay optimal scheduling policy for convergecasting in tandem networks under the one-hop interference model. However, when we apply this policy to the equivalent linear network that we described earlier, we need to clarify issues regarding transforming the schedule of the equivalent linear network back to a schedule for the original tree network.

- According to policy π_0, any node i in the equivalent linear network can schedule at most one packet during any time slot. This means that among all nodes that are i hops away from node 0 in the original tree, at most one packet will be scheduled. Note that the one-hop interference model allows multiple nodes (at the same distance from the sink) to potentially schedule their transmissions simultaneously if they do not have the same parent. However, policy π_0 does not allow such schedules. This implies that even without *Maximal Matching*, this policy is optimal. Note that a *Maximal Matching* policy is one that schedules a set of non-interfering links such that no additional link can be included in the set without interfering with at least one of the existing links, i.e., the set of links scheduled is maximal. It is interesting that even without scheduling additional non-interfering links, this policy is delay optimal.
- Suppose that a node i in the equivalent linear network is selected to schedule during a certain slot according to π_0. Consider nodes that are i hops away from node 0 in the original tree that have at least one packet to schedule. One of these nodes can be chosen *arbitrarily* to schedule its packet during that slot. This means that the optimal solution neither depends on the structure of the tree nor the number of packets at each node. For example, in Figure 1, we can arbitrarily choose to schedule one of $\{D, E, F\}$ according to π_0.
- If a node i in the equivalent linear network is selected to schedule during a certain slot according to π_0, none of the nodes that are $i - 1$ hops away from node 0 in the original tree can transmit. Since it is possible to potentially schedule a node that is at distance $i - 1$ and a node at distance i simultaneously without interference as long as the node at distance i is not a child of the node at distance $i - 1$, it is interesting that even without scheduling such non-interfering links, this policy is optimal. For example, in Figure 1, we can potentially simultaneously schedule B and E. However, this policy does not allow such a schedule because in the equivalent linear network, when R makes a transmission, Q cannot make a transmission.

Let $\mathbb{P}$ be the class of all possible activation policies. The proof that π_0 is optimal for trees where the degree of the sink is one is similar to Tassiulas's proof for tandem networks. Intuitively, the reason that π_0 is optimal for such a large class of trees (even though it is not a Maximal Matching policy) is that the link from the sink's child to the sink serves as a bottleneck for all the packets in the system. Therefore, even if we allow for a Maximal Matching based schedule, the packets have to ultimately queue up at the sink's child, and get transmitted one after another.

Theorem 1. *Consider the evolution of the system under policy π_0 and an arbitrary policy $\pi \in \mathbb{P}$. Let $\mathbf{X}$, $\mathbf{X}^0$ be the queue length processes under π and π_0 respectively when the system starts from the same initial state under both policies. For all $t = 0, 1, ...$ we have*

$$\sum_{i \in V} X_i^0(t) \leq \sum_{i \in V} X_i(t) \ a.s. \tag{2}$$

We first provide some definitions and lemmas before going into the proof of the theorem.

Definition: Let $\boldsymbol{X}$, $\boldsymbol{Y}$ be the queue length processes when the initial queue length vectors are $\boldsymbol{X}(0) = \boldsymbol{x}$, $\boldsymbol{Y}(0) = \boldsymbol{y}$ respectively, there are no exogenous arrivals, and policy π_0 schedules link activations. We say that the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are related with the partial ordering $\prec$, and we write $\boldsymbol{x} \prec \boldsymbol{y}$, if for all $t = 0, 1, ...$, we have

$$l(\boldsymbol{X}(t)) \leq l(\boldsymbol{Y}(t)), \tag{3}$$

where $l(\boldsymbol{x}) = \sum_{i \in V} x_i$ is the total number of packets in the system when the state is $\boldsymbol{x}$.

To each state $\boldsymbol{x}$ we define the *departure times* $t_i^{\boldsymbol{x}}$, $i = 1, ..., l(\boldsymbol{x})$ and the *positions* $d_i^{\boldsymbol{x}}$, $i = 1, ..., l(\boldsymbol{x})$ as follows.

Definition: Assume that the system is initially in state $\boldsymbol{x}$, there are no exogenous arrivals, and policy π_0 schedules link activations. Let $\{\boldsymbol{X}(t)\}_{t=1}^{\infty}$ be the corresponding queue length process. The time $t_i^{\boldsymbol{x}}$ is defined as

$$t_i^{\boldsymbol{x}} = \min\{t : t > 0, l(\boldsymbol{X}(t)) \leq l(\boldsymbol{x}) - i\}, \ i = 1, ..., l(\boldsymbol{x}), \tag{4}$$

and the position $d_i^{\boldsymbol{x}}$ is defined as

$$d_i^{\boldsymbol{x}} = \max\{j + 1 : \sum_{l : d(0,l) \leq j} \boldsymbol{X}_l(t) < i\}, \ i = 1, ..., l(\boldsymbol{x}). \tag{5}$$

Note that the definitions have been appropriately modified for

our topology. The corresponding definitions for the equivalent linear network will be exactly the same as in Definition 3.2 in [2]. Let us now index the packets by an index i that denotes the order in which the packets reach node 0 when the system is in state $\boldsymbol{x}$ at $t = 0$, π_0 schedules link activation, and there are no exogenous arrivals. The departure time $t_i^{\boldsymbol{x}}$ is the slot by the end of which packet i reaches node 0, and the position $d_i^{\boldsymbol{x}}$ is the distance of the node (from node 0) at which packet i was residing at $t = 0$. In the equivalent linear network, if $d_i^{\boldsymbol{x}} = k$, then packet i was residing at node k at $t = 0$.

We now show that the departure times and the positions for our topology are related in the same manner as in Equation (3.4) in [2].

Lemma 1. *For all states* $\mathbf{x}$ *we have*

$$t_i^{\mathbf{x}} = \begin{cases} d_i^{\mathbf{x}} & i = 1 \\ i & d_i^{\mathbf{x}} = 1 \\ \max\{t_{i-1}^{\mathbf{x}} + 2, d_i^{\mathbf{x}}\} & i > 1, d_i^{\mathbf{x}} > 1 \end{cases} \tag{6}$$

Proof: Consider the system operated under policy π_0, with initial state $\boldsymbol{x}$ and without arrivals. The time taken by the first packet to exit the system is simply the distance of the node (from node 0) at which it was residing at $t = 0$ because it gets forwarded by one hop during each time slot. Therefore, $t_1^{\boldsymbol{x}} = d_1^{\boldsymbol{x}}$.

For a packet i such that $d_i^{\boldsymbol{x}} = 1$, the time taken by this packet to leave the system is i since at each slot one packet will be forwarded from node 1 (in the equivalent linear network, and hence in the original tree) to node 0 until the time that node 1 has no more packets to send. Therefore, in this case, the packet i will reach node 0 at the end of slot i. Therefore, if $d_i^{\boldsymbol{x}} = 1$, $t_i^{\boldsymbol{x}} = i$.

If $i > 1$ and $d_i^{\boldsymbol{x}} > 1$, we distinguish the following cases.

Case 1: $d_i^{\boldsymbol{x}} - t_{i-1}^{\boldsymbol{x}} \geq 2$.

At any slot $t < t_{i-1}^{\boldsymbol{x}}$, the packet $i - 1$ should reside in a node j in the original tree such that $d(0, j) \leq t_{i-1}^{\boldsymbol{x}} - t$ because it should reach the destination in $t_{i-1}^{\boldsymbol{x}} - t$ slots, and cannot be forwarded faster than one hop during each slot. Also, at time t, the packet i should reside in a node m such that $d(0, m) \geq d_i^{\boldsymbol{x}} - t$ since it cannot move faster towards the destination than one hop per slot. Therefore we have $d(0, m) \geq d_i^{\boldsymbol{x}} - t \geq t_{i-1}^{\boldsymbol{x}} - t + 2 \geq d(0, j) + 2$. This implies that packet $i - 1$ will be, at each slot t, at least two hops closer to the destination than packet i in both the original tree as well as the equivalent linear network. Therefore packet i will be the first packet in its queue (according to our convention), and all the nodes in the tree that are one hop closer to the destination than the node at which packet i currently is have no packets in their respective queues. Therefore, packet i will be forwarded by one node towards the destination at each slot. Hence, packet i will reach the destination by the end of slot $d_i^{\boldsymbol{x}}$, i.e., $t_i^{\boldsymbol{x}} = d_i^{\boldsymbol{x}}$.

Case 2: $d_i^{\boldsymbol{x}} - t_{i-1}^{\boldsymbol{x}} \leq 1$.

If $i > 1$ and $d_i^{\boldsymbol{x}} > 1$, then $t_i^{\boldsymbol{x}} \geq t_{i-1}^{\boldsymbol{x}} + 2$. This is because any packet which is not residing in node 1 at $t = 0$, can reach node 1 only when there are no packets left to schedule in node 1, since node 1 is activated otherwise. Note that this is true only when the root has one child. If the root has multiple children, a packet can reach one of the root's children even when some other child of the root is transmitting its packet to the root. This is one of the most important reasons why all the proofs in [2] works for our topology. Node 1 serves as a bottleneck. Hence, during the slot at which $i - 1$ leaves the system, packet i will be in node 2 in the equivalent linear network (corresponding to one of the children of node 1 in the original tree) or further away from the destination, and therefore it requires at least two additional slots in order to reach the destination.

We now show that $t_i^{\boldsymbol{x}} = t_{i-1}^{\boldsymbol{x}} + 2$. If packet i is forwarded towards the destination by one node at each slot then it will reach the destination by slot $d_i^{\boldsymbol{x}}$. However, this is impossible since $d_i^{\boldsymbol{x}} - t_{i-1}^{\boldsymbol{x}} \leq 1$, and we need $t_i^{\boldsymbol{x}} \geq t_{i-1}^{\boldsymbol{x}} + 2$. This means that at some slot, packet i is not forwarded from its node (say node k). Suppose that packet $i - 1$ was residing at node j during this slot. Then we must either have $d(0, j) = d(0, k)$, or $d(0, j) = d(0, k) - 1$, i.e., in the equivalent linear network packet $i - 1$ is either in the same node with i or in the node in front of i towards the destination. Therefore, at the slot at which i was not forwarded and at all subsequent slots until the time packet $i - 1$ leaves the system, packets i and $i - 1$ cannot be in two nodes m, n such that $d(0, m) - d(0, n) > 2$. Therefore, two slots after the time packet $i-1$ reaches node 0, packet i also reaches node 0. Thus, $t_i^{\boldsymbol{x}} = t_{i-1}^{\boldsymbol{x}} + 2$. ■

Lemma 2. *For any two vectors* $\mathbf{x}$ *and* $\mathbf{y}$*, we have* $\mathbf{x} \prec \mathbf{y}$ *if and only if*

$$t_i^{\mathbf{x}} \leq t_{i+k}^{\mathbf{y}}, \ i = 1, ..., l(\mathbf{x}), \tag{7}$$

where $k = l(\mathbf{y}) - l(\mathbf{x})$.

Proof: Suppose that $\boldsymbol{x} \prec \boldsymbol{y}$. We prove (7) by contradiction.

Let $\boldsymbol{X}(t)$, $\boldsymbol{Y}(t)$, $t = 0, 1, ...$ be the queue length processes when the initial queue length vectors are $\boldsymbol{X}(0) = \boldsymbol{x}$, $\boldsymbol{Y}(0) = \boldsymbol{y}$ respectively, there are no exogenous arrivals, and π_0 schedules link activations. If $t_{i+k}^{\boldsymbol{y}} < t_i^{\boldsymbol{x}}$ for some i, then by the end of slot $t_{i+k}^{\boldsymbol{y}}$, exactly $i+k$ packets have departed from the system when the initial state is $\boldsymbol{y}$ while less than i packets have departed from the system when the initial state is $\boldsymbol{x}$. Hence

$$l(\boldsymbol{Y}(t_{i+k}^{\boldsymbol{y}})) = l(\boldsymbol{y}) - i - k = l(\boldsymbol{x}) - i < l(\boldsymbol{X}(t_{i+k}^{\boldsymbol{y}})), \tag{8}$$

which contradicts $\boldsymbol{x} \prec \boldsymbol{y}$.

To prove the converse, suppose that (7) holds.

For an arbitrary slot t, let j be the packet that most recently departed from the system when the initial state is $\boldsymbol{y}$. If $j \leq k$, then clearly $l(\boldsymbol{X}(t)) \leq l(\boldsymbol{Y}(t))$. If $j > k$, then since $t_{j-k}^{\boldsymbol{x}} \leq t_j^{\boldsymbol{y}}$, by time t at least $j - k$ packets have departed from the system with initial state $\boldsymbol{x}$. Hence we have

$$l(\boldsymbol{X}(t)) \leq l(\boldsymbol{x}) - j + k = l(\boldsymbol{y}) - j = l(\boldsymbol{Y}(t)), \tag{9}$$

and hence $\boldsymbol{x} \prec \boldsymbol{y}$. ■

We now show that if there are no exogenous arrivals, policy π_0 minimizes the sum of the queue lengths of all the nodes in the system for each time slot.

Lemma 3. *If we have* $\mathbf{x} \prec \mathbf{y}$*, and* $\mathbf{i}$ *is an arbitrary activation*

vector, then for $\mathbf{u} = \mathbf{x} + \mathbf{R}g_0(\mathbf{x})$ *and* $\mathbf{z} = \mathbf{y} + \mathbf{Ri}$*, we have* $\mathbf{u} \prec \mathbf{z}$*.*

Proof: We show that for all $i = 1, ..., l(\boldsymbol{x})$ we have $t_i^{\boldsymbol{u}} \leq t_{i+l(\boldsymbol{z})-l(\boldsymbol{u})}^{\boldsymbol{z}}$ and thus from the previous lemma, we must have $\boldsymbol{u} \prec \boldsymbol{z}$.

Let $l(\boldsymbol{y}) - l(\boldsymbol{x}) = k$. We have four cases.

Case 1: $l(\boldsymbol{u}) = l(\boldsymbol{x})$, $l(\boldsymbol{z}) = l(\boldsymbol{y})$.

In this case, we need to show that $t_i^{\boldsymbol{u}} \leq t_{i+k}^{\boldsymbol{z}}$, $\forall\ i = 1, ..., l(\boldsymbol{u})$.

Since $\boldsymbol{u}$ results from applying policy π_0, from the definition of departure times, it immediately follows that

$$t_i^{\boldsymbol{u}} = t_i^{\boldsymbol{x}} - 1. \tag{10}$$

We now show by induction on i that

$$t_i^{\boldsymbol{y}} \geq t_i^{\boldsymbol{z}} \geq t_i^{\boldsymbol{y}} - 1. \tag{11}$$

$i = 1$: The following hold.

$$t_1^{\boldsymbol{y}} = d_1^{\boldsymbol{y}}$$
$$d_1^{\boldsymbol{y}} \geq d_1^{\boldsymbol{z}} \geq d_1^{\boldsymbol{y}} - 1$$

The first equation follows from Lemma 1, and the second follows from the fact that the first packet cannot travel more than one hop in one slot. Therefore, we have

$$t_1^{\boldsymbol{z}} = d_1^{\boldsymbol{z}} \geq d_1^{\boldsymbol{y}} - 1 = t_1^{\boldsymbol{y}} - 1$$
$$t_1^{\boldsymbol{z}} \leq d_1^{\boldsymbol{y}} = t_1^{\boldsymbol{y}}.$$

Thus, the result holds for $i = 1$.

By the induction hypothesis, assume that the result holds for some i.

$i+1$: If $d_{i+1}^{\boldsymbol{z}} = 1$, then $t_{i+1}^{\boldsymbol{z}} = i+1$, and $t_{i+1}^{\boldsymbol{y}} = i+2$ if the packet $i+1$ was in node 2 in the equivalent linear network or $t_{i+1}^{\boldsymbol{y}} = i+1$ if the packet $i+1$ was in node 1 and it was not scheduled. Therefore, $t_{i+1}^{\boldsymbol{z}} = t_{i+1}^{\boldsymbol{y}}$ or $t_{i+1}^{\boldsymbol{z}} = t_{i+1}^{\boldsymbol{y}} - 1$. Thus the result holds when $d_{i+1}^{\boldsymbol{z}} = 1$.

If $d_{i+1}^{\boldsymbol{z}} > 1$, then $t_{i+1}^{\boldsymbol{z}} = \max\{t_i^{\boldsymbol{z}} + 2, d_{i+1}^{\boldsymbol{z}}\}$.

If the packet $i+1$ is forwarded by one node because of the activation vector $\boldsymbol{i}$, then $d_{i+1}^{\boldsymbol{z}} = d_{i+1}^{\boldsymbol{y}} - 1$. Else, $d_{i+1}^{\boldsymbol{z}} = d_{i+1}^{\boldsymbol{y}}$.

Now, we have

$$\begin{aligned} t_{i+1}^{\boldsymbol{y}} - 1 \quad &= \max\{t_i^{\boldsymbol{y}} + 2, d_{i+1}^{\boldsymbol{y}}\} - 1 \\ &= \max\{t_i^{\boldsymbol{y}} - 1 + 2, d_{i+1}^{\boldsymbol{y}} - 1\} \\ &\leq \max\{t_i^{\boldsymbol{z}} + 2, d_{i+1}^{\boldsymbol{z}}\} = t_{i+1}^{\boldsymbol{z}} \\ &\leq \max\{t_i^{\boldsymbol{y}} + 2, d_{i+1}^{\boldsymbol{y}}\} = t_{i+1}^{\boldsymbol{y}}, \end{aligned}$$

where the third relation follows because $t_i^{\boldsymbol{z}} \geq t_i^{\boldsymbol{y}} - 1$ and $d_{i+1}^{\boldsymbol{z}} \geq d_{i+1}^{\boldsymbol{y}} - 1$, and the fourth relation holds because $t_i^{\boldsymbol{z}} \leq t_i^{\boldsymbol{y}}$ and $d_{i+1}^{\boldsymbol{z}} \leq d_{i+1}^{\boldsymbol{y}}$.

Therefore by induction, the relation (11) holds. From the relations (10), (11), and the fact that $\boldsymbol{x} \prec \boldsymbol{y}$, it follows that $t_i^{\boldsymbol{u}} \leq t_{i+k}^{\boldsymbol{z}}$, $\forall\ i = 1, ..., l(\boldsymbol{u})$.

Case 2: $l(\boldsymbol{u}) = l(\boldsymbol{x}) - 1$, $l(\boldsymbol{z}) = l(\boldsymbol{y})$.

In this case, we need to show that $t_i^{\boldsymbol{u}} \leq t_{i+k+1}^{\boldsymbol{z}}$, $\forall\ i = 1, ..., l(\boldsymbol{u})$.

Since one packet exits the system according to policy π_0, the $i+1^{th}$ packet in the previous slot now becomes the i^{th} packet. Therefore,

$$t_i^{\boldsymbol{u}} = t_{i+1}^{\boldsymbol{x}}. \tag{12}$$

For $\boldsymbol{z}$, the situation is identical to that of Case 1. Therefore, the relation (11) holds. Therefore, it follows that $t_i^{\boldsymbol{u}} \leq t_{i+k+1}^{\boldsymbol{z}}$, $\forall\ i = 1, ..., l(\boldsymbol{u})$.

Case 3: $l(\boldsymbol{u}) = l(\boldsymbol{x}) - 1$, $l(\boldsymbol{z}) = l(\boldsymbol{y}) - 1$.

In this case, we need to show that $t_i^{\boldsymbol{u}} \leq t_{i+k}^{\boldsymbol{z}}$, $\forall\ i = 1, ..., l(\boldsymbol{u})$.

From Case 2 for $\boldsymbol{u}$, we have $t_i^{\boldsymbol{u}} = t_{i+1}^{\boldsymbol{x}}$.

For $\boldsymbol{z}$, we now show by induction that

$$t_{i+1}^{\boldsymbol{y}} \geq t_i^{\boldsymbol{z}} \geq t_{i+1}^{\boldsymbol{y}} - 1. \tag{13}$$

$i = 1$: We have $t_1^{\boldsymbol{z}} = d_1^{\boldsymbol{z}} \leq d_2^{\boldsymbol{y}} \leq t_2^{\boldsymbol{y}}$. If $t_2^{\boldsymbol{y}} = d_2^{\boldsymbol{y}}$, then $t_1^{\boldsymbol{z}} = d_1^{\boldsymbol{z}} \geq d_2^{\boldsymbol{y}} - 1 \geq t_2^{\boldsymbol{y}} - 1$, and $t_1^{\boldsymbol{z}} = d_1^{\boldsymbol{z}} \leq d_2^{\boldsymbol{y}} \leq t_2^{\boldsymbol{y}}$. Therefore, the result holds in this case. On the other hand, it is also possible that $t_2^{\boldsymbol{y}} = d_2^{\boldsymbol{y}} + 1$ if the second packet resided at node 2 at the previous slot in the equivalent linear network. Since the first packet was scheduled during this slot, the second packet still remains at node 2. In this case, $t_1^{\boldsymbol{z}} = d_1^{\boldsymbol{z}} = d_2^{\boldsymbol{y}} = t_2^{\boldsymbol{y}} - 1$. Therefore, the relation (13) holds for the first packet in state $\boldsymbol{z}$.

Assume that it holds for some i by the induction hypothesis.

$i+1$: If $t_{i+1}^{\boldsymbol{z}} = d_{i+1}^{\boldsymbol{z}}$, then it follows that

$$t_{i+1}^{\boldsymbol{z}} = d_{i+1}^{\boldsymbol{z}} \leq d_{i+2}^{\boldsymbol{y}} \leq t_{i+2}^{\boldsymbol{y}}. \tag{14}$$

If $t_{i+1}^{\boldsymbol{z}} = t_i^{\boldsymbol{z}} + 2$, then

$$t_{i+1}^{\boldsymbol{z}} = t_i^{\boldsymbol{z}} + 2 \leq t_{i+1}^{\boldsymbol{y}} + 2 \leq t_{i+2}^{\boldsymbol{y}}. \tag{15}$$

If $t_{i+2}^{\boldsymbol{y}} = d_{i+2}^{\boldsymbol{y}}$, then

$$t_{i+2}^{\boldsymbol{y}} = d_{i+2}^{\boldsymbol{y}} \leq d_{i+1}^{\boldsymbol{z}} + 1 \leq t_{i+1}^{\boldsymbol{z}} + 1. \tag{16}$$

If $t_{i+2}^{\boldsymbol{y}} = t_{i+1}^{\boldsymbol{y}} + 2$, then

$$t_{i+1}^{\boldsymbol{z}} \geq t_i^{\boldsymbol{z}} + 2 \geq t_{i+1}^{\boldsymbol{y}} + 2 - 1 = t_{i+2}^{\boldsymbol{y}} - 1. \tag{17}$$

From the four relations above, the relation (13) holds for $i+1$. Therefore, by induction, it holds for all i.

The relations (13) and (12) imply that $t_i^{\boldsymbol{u}} \leq t_{i+k}^{\boldsymbol{z}}$, $\forall\ i = 1, ..., l(\boldsymbol{u})$.

Case 4: $l(\boldsymbol{u}) = l(\boldsymbol{x})$, $l(\boldsymbol{z}) = l(\boldsymbol{y}) - 1$.

In this case, we need to show that $\forall i$, $t_i^{\boldsymbol{u}} \leq t_{i+k-1}^{\boldsymbol{z}}$.

The case for $\boldsymbol{u}$ is identical to that in Case 1, and the case for $\boldsymbol{z}$ is identical to that in Case 3. Hence, the result follows.

Thus, we have shown that $\boldsymbol{u} \prec \boldsymbol{z}$. ■

We now show that the ordering $\prec$ is preserved even after a packet arrives at any node in the network. To be precise, let $\boldsymbol{e}_j$ be the vector which has all its elements equal to zero except for the element j which is 1. Then we have the following.

Lemma 4. *If we have* $\mathbf{x} \prec \mathbf{y}$*, then for all* $j \in V$*, we also have* $\mathbf{x} + \mathbf{e}_j \prec \mathbf{y} + \mathbf{e}_j$.

Proof: First note that if a packet arrives at a node j in the tree, then it arrives at node $d(0,j)$ in the equivalent linear network. For notational convenience, we represent $d(0,j)$ as simply j for the rest of this proof. Then the lemma means that for all $j \in \{1,2,...,N\}$, we have $\boldsymbol{x} + \boldsymbol{e}_j \prec \boldsymbol{y} + \boldsymbol{e}_j$.

Let $\boldsymbol{u} = \boldsymbol{x} + \boldsymbol{e}_j$, and $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{e}_j$. Since $\boldsymbol{x} \prec \boldsymbol{y}$, $l(\boldsymbol{y}) - l(\boldsymbol{x}) = k \geq 0$. Thus, $l(\boldsymbol{z}) - l(\boldsymbol{u}) = k$.

We need to show that $\forall i = 1,2,...,l(\boldsymbol{x})+1$

$$t_i^{\boldsymbol{u}} \leq t_{i+k}^{\boldsymbol{z}} \tag{18}$$

Let $m = \sum_{l=1}^{j} x_l + 1$, and $n = \sum_{l=1}^{j} y_l + 1$.

Case 1: $i < m$, $i + k < n$.

For all such packets, $t_i^{\boldsymbol{u}} = t_i^{\boldsymbol{x}}$, and $t_{i+k}^{\boldsymbol{z}} = t_{i+k}^{\boldsymbol{y}}$ because the transmission schedules of these packets are not affected by the arrival of a packet at node j. Therefore, the relation (18) holds for this case.

Case 2: $i \geq m$, $i + k < n$.

We have $d_i^{\boldsymbol{u}} \leq d_i^{\boldsymbol{x}}$ $\forall i = 1,...,l(\boldsymbol{x})$ because if $i < m$, $d_i^{\boldsymbol{u}} = d_i^{\boldsymbol{x}}$, and if $m < i \leq l(\boldsymbol{x})$, $d_i^{\boldsymbol{u}} = d_{i-1}^{\boldsymbol{x}} \leq d_i^{\boldsymbol{x}}$.

We want to show now that $t_i^{\boldsymbol{u}} \leq t_i^{\boldsymbol{x}}$. We do this by induction.

$i = 1$: We have $t_1^{\boldsymbol{u}} = d_1^{\boldsymbol{u}} \leq d_1^{\boldsymbol{x}} = t_1^{\boldsymbol{x}}$. Thus, it holds for $i = 1$.

Assume that it holds for some i by the induction hypothesis.

$i + 1$: If $t_{i+1}^{\boldsymbol{u}} = d_{i+1}^{\boldsymbol{u}}$ and $t_{i+1}^{\boldsymbol{x}} = d_{i+1}^{\boldsymbol{x}}$, then since $d_{i+1}^{\boldsymbol{u}} \leq d_{i+1}^{\boldsymbol{x}}$, we have $t_{i+1}^{\boldsymbol{u}} \leq t_{i+1}^{\boldsymbol{x}}$.

If $t_{i+1}^{\boldsymbol{u}} = t_i^{\boldsymbol{u}} + 2$ and $t_{i+1}^{\boldsymbol{x}} = t_i^{\boldsymbol{x}} + 2$, then it immediately follows that $t_{i+1}^{\boldsymbol{u}} \leq t_{i+1}^{\boldsymbol{x}}$ since $t_i^{\boldsymbol{u}} \leq t_i^{\boldsymbol{x}}$ by the induction hypothesis.

If $t_{i+1}^{\boldsymbol{u}} = d_{i+1}^{\boldsymbol{u}}$ and $t_{i+1}^{\boldsymbol{x}} = t_i^{\boldsymbol{x}} + 2$, then we have $t_i^{\boldsymbol{x}} + 2 \geq d_{i+1}^{\boldsymbol{x}} \geq d_{i+1}^{\boldsymbol{u}} = t_{i+1}^{\boldsymbol{u}}$.

If $t_{i+1}^{\boldsymbol{u}} = t_i^{\boldsymbol{u}} + 2$ and $t_{i+1}^{\boldsymbol{x}} = d_{i+1}^{\boldsymbol{x}}$, then we have $d_{i+1}^{\boldsymbol{x}} \geq t_i^{\boldsymbol{x}} + 2 \geq t_i^{\boldsymbol{u}} + 2 = t_{i+1}^{\boldsymbol{u}}$.

Thus, the result holds for $i+1$. Hence, by induction, $t_i^{\boldsymbol{u}} \leq t_i^{\boldsymbol{x}}$ $\forall i$.

Also, from Case 1, for $\boldsymbol{z}$, $t_{i+k}^{\boldsymbol{z}} = t_{i+k}^{\boldsymbol{y}}$ when $i + k < n$. Therefore, $t_i^{\boldsymbol{u}} \leq t_{i+k}^{\boldsymbol{z}}$ for $i \geq m$ and $i + k < n$.

Case 3: $i \leq m$, $i + k \geq n$.

For this case, we give a proof by contradiction.

Suppose that for some i, $t_i^{\boldsymbol{u}} > t_{i+k}^{\boldsymbol{z}}$.

Since $i + k \geq n$ and $i < m$, we have

$$t_{i+k}^{\boldsymbol{z}} \geq d_{i+k}^{\boldsymbol{z}} \geq j \geq d_i^{\boldsymbol{u}}, \tag{19}$$

$t_i^{\boldsymbol{u}} > t_{i+k}^{\boldsymbol{z}}$ means that $t_i^{\boldsymbol{u}} > d_i^{\boldsymbol{u}}$. Therefore, $t_i^{\boldsymbol{u}} = t_{i-1}^{\boldsymbol{u}} + 2$. So $t_{i-1}^{\boldsymbol{u}} + 2 > t_{i+k}^{\boldsymbol{z}} \geq t_{i+k-1}^{\boldsymbol{z}} + 2$. This implies that $t_{i-1}^{\boldsymbol{u}} > t_{i+k-1}^{\boldsymbol{z}}$.

Iteratively substitute $i = i-1$ until either $i = 1$ or $i+k < n$. If $i = 1$, then $t_1^{\boldsymbol{u}} = d_1^{\boldsymbol{u}}$. However, this contradicts $t_i^{\boldsymbol{u}} > d_i^{\boldsymbol{u}}$ $\forall i$. If $i + k < n$, then $t_{i+k}^{\boldsymbol{z}} = t_{i+k}^{\boldsymbol{y}}$ and $t_i^{\boldsymbol{u}} = t_i^{\boldsymbol{x}}$ for $i < m$. However, $t_{i+k}^{\boldsymbol{z}} < t_i^{\boldsymbol{u}}$ then contradicts the fact that $\boldsymbol{x} \prec \boldsymbol{y}$.

Hence $t_i^{\boldsymbol{u}} \leq t_{i+k}^{\boldsymbol{z}}$ by contradiction.

Case 4: $i > m$, $i + k \leq n$.

Since $i > m$, we have $d_i^{\boldsymbol{u}} = d_{i-1}^{\boldsymbol{x}}$. Similarly, when $i+k > n$ we have $d_{i+k}^{\boldsymbol{z}} = d_{i+k-1}^{\boldsymbol{y}}$.

If $i + k = n < l(\boldsymbol{z})$, $d_n^{\boldsymbol{z}} \leq d_n^{\boldsymbol{y}}$, and so $t_n^{\boldsymbol{z}} = \max\{t_{n-1}^{\boldsymbol{z}} + 2, d_n^{\boldsymbol{z}}\} \leq \max\{t_{n-1}^{\boldsymbol{y}} + 2, d_n^{\boldsymbol{y}}\} = t_n^{\boldsymbol{y}}$. Therefore, $t_n^{\boldsymbol{z}} \leq t_n^{\boldsymbol{y}}$.

We now show that $t_{i+k}^{\boldsymbol{z}} \leq t_{i+k-1}^{\boldsymbol{y}}$ when $i + k \geq n$.

For the base case, consider $i + k = n$: Since $t_n^{\boldsymbol{z}} \geq t_{n-1}^{\boldsymbol{z}} = t_{n-1}^{\boldsymbol{y}}$, we have $t_n^{\boldsymbol{z}} \geq t_{n-1}^{\boldsymbol{y}}$. Thus the result holds for $i+k = n$.

Assume that the result holds for $i + k = l > n$.

Consider $i + k = l + 1$: We have

$$t_{l+1}^{\boldsymbol{z}} = \max\{t_l^{\boldsymbol{z}} + 2, d_{l+1}^{\boldsymbol{z}}\} \geq \max\{t_{l-1}^{\boldsymbol{y}} + 2, d_l^{\boldsymbol{y}}\} = t_l^{\boldsymbol{y}}, \tag{20}$$

since $d_{l+1}^{\boldsymbol{z}} = t_l^{\boldsymbol{y}}$ and $t_l^{\boldsymbol{z}} \geq t_{l-1}^{\boldsymbol{y}}$.

Thus the result holds for $l + 1$.

Therefore, by induction, $t_{i+k}^{\boldsymbol{z}} \geq t_{i+k-1}^{\boldsymbol{y}}$ $\forall\, i + k \geq n$.

We now prove the main result for Case 4 by contradiction. Suppose that $t_i^{\boldsymbol{u}} > t_{i+k}^{\boldsymbol{z}}$ for some i.

If $t_i^{\boldsymbol{u}} = d_i^{\boldsymbol{u}}$, then we have

$$t_{i-1}^{\boldsymbol{x}} \geq d_{i-1}^{\boldsymbol{x}} = d_i^{\boldsymbol{u}} = t_i^{\boldsymbol{u}} > t_{i+k}^{\boldsymbol{z}} \geq t_{i+k-1}^{\boldsymbol{y}}. \tag{21}$$

This clearly contradicts the fact that $\boldsymbol{x} \prec \boldsymbol{y}$. This implies that $t_i^{\boldsymbol{u}} = t_{i-1}^{\boldsymbol{u}} + 2$.

If $d_{i+k}^{\boldsymbol{z}} = 1$, then $i + k = n$, and hence $i = m$ (since the packet arrives at node 1). Since we have $i > m$, we need to have $d_{i+k}^{\boldsymbol{z}} > 1$. If this is the case, then $t_{i+k}^{\boldsymbol{z}} \geq t_{i+k-1}^{\boldsymbol{z}} + 2$. So we have

$$t_i^{\boldsymbol{u}} = t_{i-1}^{\boldsymbol{u}} + 2 > t_{i+k}^{\boldsymbol{z}} \geq t_{i+k-1}^{\boldsymbol{z}} + 2. \tag{22}$$

Therefore, $t_{i-1}^{\boldsymbol{u}} > t_{i+k-1}^{\boldsymbol{z}}$.

Iteratively substitute $i = i-1$ until either $i = m$ or $i+k < n$. If $i = m$, then $t_{m-1}^{\boldsymbol{u}} > t_{i+k-1}^{\boldsymbol{z}}$ which contradicts either Case 2 or Case 3 depending on whether $i + k < n$ or $i + k \geq n$ respectively. If $i + k < n$, then $t_{i-1}^{\boldsymbol{u}} > t_{i+k-1}^{\boldsymbol{z}}$ contradicts Case 2.

Hence, the result is proven for this case by contradiction.

Since the four cases exhaustively include all possibilities, the lemma follows. ■

We now proceed to the proof of Theorem 1.

Proof of Theorem 1:

Proof: For $t = 0$, we have $\boldsymbol{X}^0(0) = \boldsymbol{X}(0)$, and hence $\boldsymbol{X}^0(t) \prec \boldsymbol{X}(t)$ at $t = 0$. Assume that $\boldsymbol{X}^0(t) \prec \boldsymbol{X}(t)$ is true for some t. We show that it holds for $t + 1$ as well. Let $\boldsymbol{I}(t+1)$ be the activation vector under some policy π at $t + 1$. Then from Lemma 3 we have

$$(\boldsymbol{X}^0(t) + \boldsymbol{R}g_0(\boldsymbol{X}^0(t))) \prec \boldsymbol{X}(t) + \boldsymbol{R}\boldsymbol{I}(t+1). \tag{23}$$

The arrival vector $\boldsymbol{A}(t+1)$ for the equivalent linear network can be written as

$$\boldsymbol{A}(t+1) = \sum_{i=1}^{N} A_i(t+1)\boldsymbol{e}_i. \tag{24}$$

Hence from Lemma 4 and the relation (23) we can see that

$$\boldsymbol{X}^0(t+1) \qquad = \boldsymbol{X}^0(t) + \boldsymbol{R}g_0(\boldsymbol{X}^0(t)) + \sum_{i=1}^{N} A_i(t+1)\boldsymbol{e}_i$$

$$\prec \boldsymbol{X}(t) + \boldsymbol{RI}(t+1) + \sum_{i=1}^{N} A_i(t+1)\boldsymbol{e}_i$$
$$= \boldsymbol{X}(t+1).$$

Since we are only interested in the sum of all the queues, the result for the equivalent linear network holds for the tree topology as well. ■

IV. General Tree Topologies

In this section, we analyze whether there exists a sample-path delay optimal scheduling policy for general trees. We first consider trees where the root has multiple children, and all but one of the children are leaf nodes.

Consider the following policy π_0^T.

Policy π_0^T:

1) At any time slot t, if the root's child that is not a leaf node has a packet, then schedule the root's child. Do not schedule the other children of the root. Schedule the rest of the tree according to policy π_0. If all of the root's children are leaf nodes, pick any one of them that has a packet to transmit, and schedule it. Do not schedule the other children.
2) If the root's child that is not a leaf node does not have a packet, schedule any one of the root's other children that has a packet, and do not schedule the other children. Schedule the rest of the tree according to π_0.

Theorem 2. *Policy π_0^T is a sample-path delay optimal scheduling policy for trees where the root has multiple children, and all but one of the children are leaf nodes.*

Proof: We prove this theorem by induction on time $t = 0, 1, ...$ Specifically, we show that at any time slot t, the number of packets that have exited the system by that slot is maximum among all policies. Therefore, the sum of the queues of all the nodes in the network is minimum at that time slot.

$t = 0$: The number of packets that have exited the system is zero, and this is maximum among all policies.

Induction hypothesis at time t: Suppose that the number of packets that have exited the system by t is m according to policy π_0^T, i.e., m is the maximum number of packets that could have exited the system according to any policy.

$t+1$: There are two cases.

Case 1: At least one of the root's children have a packet to schedule.

In this case, according to π_0^T, exactly one of these children will be scheduled, and the number of packets that exit the system by $t+1$ is $m+1$. This is maximum since at most one packet can exit during a time slot, and the maximum number of packets that have exited the system by t is m.

Case 2: None of the root's children have a packet to schedule.

This means that all the packets from all the root's children that are leaf nodes have exited the system, and there are no new arrivals at these nodes. Further, the branch corresponding to the root's child that is not a leaf node is scheduled according to policy π_0 at all time slots. Therefore, the number of packets that have exited from this branch at any time slot is maximum because of the optimality of π_0. Hence, the maximum number of packets that have exited the system by $t+1$ is also m.

Therefore, by induction, at each time slot, for any arrival pattern, the number of packets that have exited the system is maximum among all policies, and hence π_0^T is sample-path delay optimal. ■

To conclude our analysis on sample-path delay optimal policies for trees, we now show that for all other tree structures there exists no sample-path delay optimal policy without knowledge about future packet arrivals.

Theorem 3. *There exists no causal sample path delay optimal scheduling policy for the convergecasting problem when the sink node has two or more non-leaf children, i.e., without knowing future information, for such trees, there is at least one type of traffic arrival pattern for which no sample path delay optimal scheduling policy exists.*

Proof: Let us begin by first considering the smallest tree with multiple non-leaf children, as shown in Figure 2. Assume that at $t = 0$, A and B have one packet each while C and D have no packets. In the absence of information of future traffic arrivals, we have no choice but to pick one of A or B to schedule at $t = 0$.

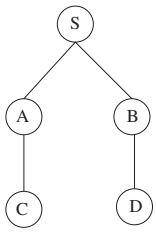

Fig. 2. Tree with no sample-path delay optimal scheduling policy

Suppose we pick A to schedule. So at $t = 1$, A has no packets left while B has one packet left. Now suppose that at $t = 1$, we have an arrival at D. There are no arrivals at any other node, and there are no future arrivals in the system. It then takes an additional three time slots for the packets at B and D to exit the system.

Suppose that we had picked B to schedule at $t = 0$. Then, at $t = 1$, A has one packet left to schedule while B has no packets left. In this case, at $t = 1$, A would have transmitted to the root and D would have transmitted its packet to B simultaneously. At $t = 2$, B would have transmitted this packet to the root. Therefore, it just takes two time slots for all the packets to exit the system.

Note that if we had picked B to schedule at $t = 0$ without knowing about future arrivals, an arrival at C at $t = 1$ would have ensured that picking B in the earlier time slot was sub-optimal.

Thus, this example shows that even for this simple tree with four nodes, there exists no causal sample-path delay optimal

policy. It is now straightforward to see that for a general tree where the root has multiple children that are not leaf nodes, there exists no causal sample-path delay optimal policy. This is because such a tree would contain the above example as a part of the structure. So by simply considering the arrival pattern described above, and assuming that there are no packets and arrivals in the rest of the tree, the same argument would apply. ■

Thus, from Theorem 2 and Theorem 3, we have completely classified tree structures for which there exists sample-path delay optimal policies, and for which there do not.

A. Discussion

While having a sample-path delay optimal policy is ideal, they exist for very limited topologies. We now make a number of interesting observations about other delay metrics studied in the literature for tree structures.

- *Large deviations metric*: In [1], Venkataramanan et. al., have shown that for the convergecasting problem in general trees, Tassiulas's policy is delay optimal in the large deviations sense. However, this policy is actually not even evacuation time optimal. For instance, consider the following tree (Figure 3) with four nodes A, B, C, D, and a root. Suppose that A, B, and D have one packet each. According to the policy in [1], either A or B can be chosen to schedule arbitrarily during the first time slot. However, one can easily see that if A were chosen to schedule during the first time slot, the time to evacuate the system is 3 slots. But if B were chosen to schedule during the first time slot, the time to evacuate the system is 4 slots.
- *Evacuation time optimality*: In [5], Florens et. al., have proposed an evacuation time optimal policy for general trees by prioritizing the branches of the tree according to the time required to evacuate each individual branch.

Both these metrics are interesting since they provide optimal scheduling policies (in their respective senses) for general trees. However, the evacuation time metric does not consider arrivals, and there exists instances where the policy based on large deviations is not evacuation time optimal. Therefore, care should be taken when designing scheduling algorithms based on these policies.

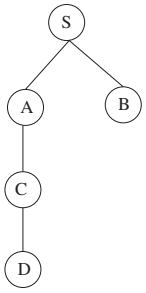

Fig. 3. Arbitrarily choosing branches is not evacuation time optimal

V. Conclusion

We studied sample-path delay optimal scheduling for convergecasting in trees. We showed that sample path optimal policies can only exist for tree structures where all, but one, of the root's children is a non-leaf node. We obtained a class of sample-path delay optimal policies for all feasible tree structures. Further, we showed that these are the only tree structures for which there exists sample-path delay optimal policies for the convergecasting problem. The fact that causal sample path optimal policies exist in very limited cases is, however, a limitation of this metric. This emphasizes the need to study other "softer" metrics for delay. For instance, when the stochastic nature of the arrival pattern is known, one could study the expected delay. These are interesting problems for future work.

References

[1] V. J. Venkataramanan and X. Lin, "Low-complexity scheduling algorithm for sum-queue minimization in wireless convergecast," Purdue University, Tech. Rep., 2010.

[2] L. Tassiulas and A. Ephremides, "Dynamic scheduling for minimum delay in tandem and parallel constrained queue networks," *Annals of Operations Research*, 1993.

[3] T. Ji, E. Athanasopoulou, and R. Srikant, "Optimal scheduling policies in small generalized switches," in *IEEE INFOCOM*, 2009.

[4] G. R. Gupta and N. B. Shroff, "Delay analysis and optimality of scheduling policies for multi-hop wireless networks," *To appear in IEEE/ACM Transactions on Networking*, 2010.

[5] C. Florens, M. Franceschetti, and R. J. McEliece, "Lower bounds on data collection time in sensory networks," *IEEE Journal on Selected Areas in Communications*, 2004.

[6] J. C. Bermond, L. Gargano, and A. A. Rescigno, "Gathering with minimum delay in tree sensor networks," *Lecture Notes in Computer Science*, 2008.

[7] L. Gargano and A. A. Rescigno, "Optimally fast data gathering in sensor networks," *Lecture Notes in Computer Science*, 2006.