Online Packet Scheduling with Hard Deadlines in Wired Networks

Zhoujia Mao, C. Emre Koksal, Ness B. Shroff
E-mail: maoz@ece.osu.edu, koksal@ece.osu.edu, shroff@ece.osu.edu

***Abstract*—The problem of online job or packet scheduling with hard deadlines has been studied extensively in the single hop setting, whereas it is notoriously difficult in the multihop setting. This difficulty stems from the fact that packet scheduling decisions at each hop influences and are influenced by decisions on other hops and only a few provably efficient online scheduling algorithms exist in the multihop setting. We consider a general multihop wired network topology in which packets with various deadlines and weights arrive at and are destined to different nodes through given routes. We study the problem of joint admission control and packet scheduling in order to maximize the cumulative weights of the packets that reach their destinations within their deadlines. We first focus on uplink transmissions in the tree topology and show that the well known earliest deadline first algorithm achieves the same performance as the optimal off-line algorithm for any feasible arrival pattern. We then address the general topology with multiple source-destination pairs, develop a simple online algorithm and show that it is $O(P_M \log P_M)$-competitive where P_M is the maximum route length among all packets. Our algorithm only requires information along the route of each packet and our result is valid for general arrival samples. Via numerical results, we show that our algorithm achieves performance that is comparable to the non-causal optimal off-line algorithm. To the best of our knowledge, this is the first algorithm with a provable (based on a sample-path construction) competitive ratio, subject to hard deadline constraints for general network topologies.**

I. Introduction

We consider a wired network in which nodes receive packets with various (hard) deadlines, enqueued at the intermediate nodes through multiple hops along given routes to given destinations. We assume a time slotted system in which each packet has an identical (unit) length and each link in the network can serve an integer number of packets at a given time slot. Each packet has a certain weight and a deadline and we address the problem of scheduler design in order to maximize the total weight over the packets that are successfully transferred to their destinations within their deadlines. We first focus on the tree topology and show that the Earliest Deadline First (EDF) algorithm achieves the same performance as the optimal off-line algorithm for any feasible or under-loaded (i.e., there exists an off-line algorithm under which all jobs can be served before their deadlines) network arrival pattern. Next, we study the general topology with multiple source-destination pairs. We develop a low-complexity on-line joint admission control and packet scheduling scheme and evaluate its competitive ratio[1] with respect to the cumulative weight achieved by the optimal off-line algorithm. Our scheme only requires information of the packet queues along the route of each packet. To the best of our knowledge, this is the first scheme with a provable (based on a sample-path construction) competitive ratio in general network topologies.

The on-line job scheduling problem with hard deadlines is gaining increasing importance with the emergence of cloud computing, large data centers, and grid communications. In such applications, a large amount of time-sensitive information needs to be carried among servers and users over a mainly-wired infrastructure. Meeting the deadline requirements of these jobs with an efficient use of resources requires a careful design of schedulers that decide on how and when data should be transferred over the network. Due to the large volume of data, the complexity of schedulers should be kept low to reduce the amount of energy consumed by these data centers. To that end, our objective is to develop a low-complexity and provably efficient scheduler and an associated admission controller for deadline-constrained data.

On-line job scheduling has been a widely-studied problem. Since the seminal work in [1], various versions of the problem for single hop systems have been considered. It has been shown that EDF has the same performance as the optimal off-line algorithm [1, 2] for the scenario in which the system is under-loaded. When considering over-loaded arrivals (i.e., the case when even the best off-line policy drops some jobs), there is the additional question of whether the controller needs to decide to accept or reject a job upon arrival, i.e., admission control. With the constraint that the admission controller and the scheduler do not have to decide on a job's admission into the system and the period that it is scheduled upon arrival, it is shown in [3] that $\frac{1}{4}$ is the best competitive ratio among all on-line algorithms and an online algorithm is provided [4] to achieve this ratio. With this constraint the problem is addressed in [5, 6]. In addition to immediate decisions, the model studied in [7] imposes a penalty on the unfinished workload, and the authors propose an on-line algorithm with competitive ratio $3-2\sqrt{2}$ and show that this ratio is the best achievable ratio for all on-line algorithms. Within the single hop setting, similar problems of job scheduling have been studied in [8–12] for the scenario with parallel processors, where the controller needs to decide which machine to process each job as well as scheduling and admission control. An on-line algorithm requiring immediate decision upon job arrival is proposed in [9] with an asymptotic competitive ratio $\frac{e-1}{e}$. It is later shown

[1]The competitive ratio of an on-line algorithm is the minimum ratio of the revenue of the on-line algorithm to the revenue of the optimal off-line algorithm, where the minimization is over all possible arrival patterns.

in [10] that this ratio is the maximum achievable ratio for any on-line algorithm. In [11, 12] a penalty-based model is introduced for unfinished workload and competitive ratios were derived for various algorithms.

All of the works mentioned above require continuous processing of jobs, i.e., each job can be processed, paused, and restarted at any point in time *preemptively*. In [13], a slotted single queue system is considered in which all jobs have unit length and uniform weights, and it is shown that EDF has the same performance as the optimal off-line algorithm. In [14, 15], the same discrete model is considered with jobs having heterogenous weights and it is shown that the achievable competitive ratio is within $[0.5, \frac{\sqrt{5}-1}{2}]$. Furthermore, it is shown that the lower bound 0.5 is achieved by the largest weight first policy and a lex-optimal scheduling policy is provided.

There have also been a few works that have investigated the problem of scheduling jobs with deadlines in the multihop setting. In [16], it has been shown that a modified version of EDF achieves the same performance as the optimal off-line algorithm for any arrival sample path under a wired uplink tree with uniform link capacities and packet weights. In the first part of our paper, we consider under-loaded arrivals but allow links to have *heterogenous link capacities* and show that EDF has the same performance as that of the optimal off-line algorithm. Our approach to the problem with the general multihop topology is motivated by competitive routing in virtual circuit network [17]. In the competitive routing model, link bandwidth is the resource to be allocated. By viewing the time slots as resources, the packet scheduling problem can be transferred to a similar resource allocation problem. However, the model of packet scheduling with deadlines is different from that of routing in virtual circuit networks. In virtual circuit networks, the constraint is the link bandwidth. In contrast, here, the packet scheduling problem is constrained by the deadlines. It cannot be mapped to a bandwidth constraint problem by simply viewing time slots as resources since each packet and its scheduling decision have an impact on the potential time resources of other packets. However, as we show in the paper, the idea used in competitive routing can still be applied with clever modifications to develop a competitive ratio based framework for packet scheduling with deadlines.

Other studies that focus on wireless resource allocation with hard deadlines include [18–25]. In all of these studies except [25], either the notions of deadlines are more restricted or the flows are single hop. In [25], a utility maximization framework is developed under a general multihop wireless network with an arbitrary deadline model for arrivals, but the scheduler in their proposed algorithm needs to enumerate over all possible schedules that satisfy the interference and deadline constraints, which is extremely complex. Furthermore, the authors in [25] are only interested in the feasible (under-loaded) arrival regions.

To summarize our main contributions in this paper:

- We show that EDF has the same performance of the optimal off-line algorithm under a wired uplink tree with heterogenous link capacities for any under-loaded arrivals.
- We develop a competitive ratio based admission control and packet scheduling framework that has low complexity and is $O(P_M \log P_M)$-competitive[2] where P_M is the maximum route length among all packets, under general wired network topologies and arrival samples. To the best of our knowledge, this is the first work on packet scheduling problems with hard deadlines in a general network setting under a sample path argument. This framework also has a nice structure to be extended to wireless networks.

II. Problem Statement

We study the packet scheduling problem with hard deadlines in wired networks. We assume a time slotted system. The arrival sample path consists of K packets where each packet $i \in \{1, 2, \ldots, K\}$ (the packet set is indexed in the order of arrival times of the packets) is associated with a triplet (a_i, d_i, ρ_i). Here, a_i and d_i are the arrival time and the deadline, respectively, both of which are given in slot indices. We allow each packet i to have a weight ρ_i, which is an arbitrary real number that represents the importance of the packet. We assume infinite packet buffers at all nodes. If packet i is still at a non-destination node by the end of slot d_i, then it expires and is deleted from the network. Note that, when all packets have finite deadlines, the packet queues in the network will always remain bounded. We assume that each packet has an identical (unit) length and each link in the network can serve an integer number (possibly different for different links) of packets at a given time slot. Let us define the indicator function for any policy p, to identify whether a packet reaches its destination within its deadline as:

$$I_i^p = \begin{cases} 1 & \text{if } i \text{ reaches its destination before the end of } d_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

and the weighted revenue gained by the successfully received packets as:

$$R^p = \sum_{i \in \{1,2,\ldots,K\}} \rho_i I_i^p. \tag{2}$$

Our **objective** is to solve $\max_p R^p$.

III. Optimal Packet Scheduling in Uplink Tree Networks with Under-Loaded Arrivals

In this section, we consider a wired uplink tree network as shown in Fig. 1. Each packet i arrives at an arbitrary non-root node in slot a_i and is destined to the root through multiple hops within its deadline d_i. We assume uniform weight here for all packets, then our objective reduces to maximizing the throughput, i.e., $\max_p R^p \equiv \max_p \sum_{i \in \{1,2,\ldots,K\}} I_i^p$.

Definition 1: The *slack time* of packet i in time slot $t \in [a_i, d_i]$ is defined as $d_i - t - h_i(t) + 1$, where $h_i(t)$ is the

[2] $O(x)$-competitive means the competitive ratio goes to 0 at least as fast as $x \to \infty$.

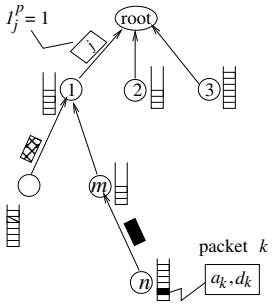

Fig. 1. An Uplink Wired Tree

number of hops from the node at which packet i resides in time slot t to the destination of i.

Definition 2: Policy p is called a *work-conserving* packet scheduling policy if it keeps each link fully utilized in each time slot, as long as the queue feeding the link contains sufficiently many unexpired packets.

Lemma 1: For any non-work-conserving policy, there exists a work conserving policy that achieves an identical or higher throughput under any given arrival pattern.

The intuition behind Lemma 1 is fairly clear, since non-work-conserving policies waste resources unnecessarily under all arrival patterns. The detailed proof can be found in [26]. From Lemma 1, we only need to focus on work conserving packet scheduling policies to solve the problem, i.e., $\max_p R^p \equiv \max_{p\in\mathcal{C}} R^p$, where $\mathcal{C}$ is the set of work conserving packet scheduling policies.

Definition 3: A *work conserving earliest deadline first (WC-EDF)* policy is one in which each node transmits the largest possible set of packets that the link capacity allows with the earliest deadlines among all the packets in its packet queue.

Theorem 1: [16] For an uplink wired tree with identical link capacity, given any arrival sample path (either under-loaded or over-loaded), the WC-EDF policy that only serves packets with non-negative slack times in each slot achieves the same performance as the optimal off-line algorithm in maximizing the throughput.

This theorem, proven in [16], is for an uplink wired tree with identical link rates. Generalization to the case in our scenario, i.e., links may have different rates (as long as they are integer number of packets/time slot) is not straightforward and not provided in [16]. In the following theorem, we extend the result for under-loaded traffic.

Theorem 2: For an uplink wired tree with possibly different link capacities that are integer number of packets per time slot, for all under-loaded arrivals, the WC-EDF policy ensures that all packets reach their destinations before their deadlines.

The detailed proof of Theorem 2 can be found in Appendix A. Note that the proof of Theorem 2 is completely different from that of Theorem 1.

Theorem 2 shows that under WC-EDF, all packets in any under-loaded arrival sample reach their destinations before their deadlines and hence generates the same throughput as the optimal off-line algorithm. Theorem 2 can then be used to test whether an arrival sample is feasible. Note that, both Theorem 1 and Theorem 2 are not generalizable to a general network topology. Indeed, it can be shown that even for simple topologies, such as the followings, there exists no on-line algorithm that has the same performance as the optimal off-line algorithm by constructing appropriate arrival patterns.

- *Down-link wired tree even with homogeneous link capacity and under-loaded arrivals*
- *Line network with multiple flow destinations even with under-loaded arrivals*
- *Uplink wired tree with heterogeneous link capacities and overloaded arrivals*

The details of all three of these examples can be found in [26]. For brevity, here, we focus only on the line network with multiple destinations, and show that no online algorithm can have the same performance as the optimal off-line algorithm even for under-loaded arrivals, by constructing sample arrivals (that the online algorithms can not support) via an adversary.

Example 1: Consider a line network $3 \rightarrow 1 \rightarrow 2$ and suppose that the link capacity of each link is 1. Initially at node 3, there are two packets k_1 and k_2 with deadline $d_{k_1} = 2, d_{k_2} = 4$ whose destinations are node 1 and 2, respectively. Suppose that node 3 transmits k_1 to node 1 in time slot 1, and that there is no arrival by the end of slot 1, then node 3 transmits k_2 to node 1 in slot 2. Let an adversary inject a packet k_3 with deadline $d_{k_3} = 3$ whose destination is node 2, at node 1 by the end of slot 2, then node 1 transmits k_3 to node 2 in slot 3. Further let the adversary inject a packet k_4 with deadline $d_{k_4} = 4$ whose destination is node 2, at node 1 by the end of slot 3, then by the end of slot 4, either k_2 or k_4 expires. However, this arrival sample is feasible since the off-line algorithm transmits k_2 in slot 1 and all four packets are able to reach their destinations within their deadlines. Similarly if node 3 transmits k_2 to node 1 in time slot 1, then let the adversary inject a packet k_3 with deadline $d_{k_3} = 2$ (whose destination is node 1), at node 3 by the end of slot 1, then by the end of slot 2, either k_1 or k_3 expires. However, this arrival sample is also feasible since the off-line algorithm transmits k_1 in slot 1 and all three packets are able to reach their destinations before their deadlines. This means under this scenario, no matter what online decision node 3 makes in slot 1, the adversary can always chooses future arrivals so that the online decision is worse than the optimal off-line algorithm even though the whole arrival sample is under-loaded.

One of the main conclusion one can draw from this section is that, other than some particular settings, there exists no on-line algorithm that achieves the same performance as an optimal off-line algorithm. This motivates our study for developing on-line algorithms that have a provable (non-zero) competitive ratio, relative to the optimal off-line algorithm for the general network topology and arrival patterns.

IV. Competitive Packet Scheduling for General Topologies and Arrival Patterns

In this section, we consider a general network topology represented by a directed graph as shown in Fig. 2. We assume

that each packet i is routed through a given path[3] P_i from its source node to its destination, where P_i denotes the set of links through which the packet is routed in order. For any link $l \in P_i$, let $h_l(i)$ denote the hop index of link l in the route of packet i. We assume in this section that each link transmits at most one packet in each slot[4] for ease of notation. We allow packets to have different weights $\rho_i,\ i = 1, 2, \ldots, K$, and our objective function is as stated in Equation (2).

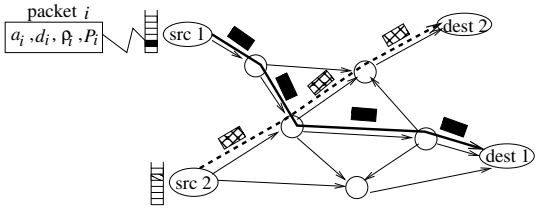

Fig. 2. General Network Topology with Multiple Source-Destination Pairs

Upon arrival of each packet, the controller of its source node decides whether to accept or reject this packet. If there are multiple packets arriving at the network in the same time slot, we assume the controllers of different packets make decisions at different instances (in the same time slot) in the same order as the packet index. If packet i is accepted, then each link $l \in P_i$ needs to reserve a time slot so that packet i will be transmitted through link l in this reserved slot. The reserved time slot is not changed in the subsequent time slots. Let $t_i(l)$ denote the index of this reserved time slot in which packet i is transmitted through link $l \in P_i$ if i is accepted. Define for any i, j, l the following indicator

$$I_l(i,j) = \begin{cases} 1 & \text{packets } i,j \text{ are accepted; } l \in P_i, l \in P_j; \\ & t_j(l) \in [a_i + (h_l(i)-1)s_i, a_i + h_l(i)s_i - 1] \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where

$$s_i = \left\lfloor \frac{d_i - a_i + 1}{|P_i|} \right\rfloor \tag{4}$$

is the average slack time per hop for packet i with $|P_i|$ as the number of total hops on its route. Note that $d_i - a_i + 1$ is the maximum allowable end to end delay for packet i in the network. If we divide this delay evenly on each hop, then s_i is the maximum allowable delay on each hop and $[a_i + (h_l(i) - 1)s_i, a_i + h_l(i)s_i - 1]$ is the set of time slots that i can use to be transmitted through link l. From another perspective, a time slot can be viewed as a resource and $[a_i + (h_l(i) - 1)s_i, a_i + h_l(i)s_i - 1]$ is then the set of available resources for packet i and s_i is the total amount of resources at each link on the route of i. When both i and j are accepted and link l is in the route of both i and j, if the reserved transmission slot of j takes one resource in i's resource set at link l, then the indicator $I_l(i,j)$ becomes 1. This means packet j consumes one unit of resource of packet i at link l. Furthermore, we define the cost of packet j taking a resource of i at link l as

$$c_l(i,j) = s_i(\mu^{\lambda_l(i,j)} - 1), \tag{5}$$

where μ is a control parameter (we will discuss how to choose the value of μ later when we analyze the performance of our algorithm proposed later in this section) and

$$\lambda_l(i,j) = \sum_{m<j} \frac{I_l(i,m)}{s_i} \tag{6}$$

is the fraction of i's resources[5] that have already been taken before arrival of packet j at link l. By letting $\lambda_l(i,1) = 0$ for all i, l, we have the following recursive relationship:

$$\lambda_l(i,j+1) = \lambda_l(i,j) + \frac{I_l(i,j)}{s_i}, \tag{7}$$

i.e., $\lambda_l(i,j)$ and thus $c_l(i,j)$ is increasing in j for any given i and l.

Before describing our algorithm, we further need to define for $i \neq j$ that

$$\tilde{I}_l(i,j) = \begin{cases} 1 & \text{the intersection of the intervals} \\ & \left[a_j + h_l(j) - 1, d_j - |P_j| + h_l(j)\right] \text{ and} \\ & \left[a_i + (h_l(i)-1)s_i, a_i + h_l(i)s_i - 1\right] \\ & \text{are nonempty; } l \in P_i, l \in P_j;\ i \text{ is accepted} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

and

$$\tilde{I}_l(j,j) = 1,\ \forall j,\ l \in P_j. \tag{9}$$

Note that $\left[a_i + h_l(i) - 1, d_i - |P_i| + h_l(i)\right]$ is the region of time slots that packet i can possibly stay at link l for it to reach the destination before its deadline under any algorithm. Define

$$S_i = d_i - a_i - |P_i| + 2 \tag{10}$$

to be the maximum possible delay of packet i at each link $l \in P_i$ and it is easy to see that $s_i \leq S_i$ for all i. Variable $\tilde{I}_l(i,j)$ indicates whether packet j may take a resource of packet i under any possible scenario. Note that $t_l(j) \in [a_j + h_l(j) - 1, d_j - |P_j| + h_l(j)]$ for all $l \in P_j$ for any scheduler, since allocating any time slot out of this interval to transmit j over $l \in P_j$ will lead to the expiration of j. Hence, one can see that $I_l(i,j) \leq \tilde{I}_l(i,j)$ for all i, j, l from Eqs. (3), (8), and (9).

Our admission control and packet scheduling algorithm is described in Algorithm 1. Note that from Equation (8), we have $\sum_l \sum_{i \leq j} \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,j) = \sum_{l \in P_j} \sum_{i \leq j} \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,j)$, i.e., the calculation only needs information on the route[6] of packet j.

[3]Our framework can be easily generalized when the route of each packet is not predetermined, but for ease of notation, we only present our idea for fixed routing.

[4]Our results can be generalized when link rates are distinct, as long as they are integer number of packets per time slot.

[5]If link capacity is integer $C > 1$, then Cs_i is the total amount of resources per link on route and definitions of $c_l(i,j)$ and $\lambda_l(i,j)$ need to be modified accordingly.

[6]If the route of each packet is not predetermined and there exist multiple possible routes whose total cost is less than the packet weight, then choose the route with the smallest cost to route the accepted packet.

Algorithm 1 Admission Control and Packet Scheduling

Upon arrival of packet j:
1) If $\sum_l \sum_{i\le j} \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,j) > \rho_j$, then reject j;
2) If $\sum_l \sum_{i\le j} \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,j) \le \rho_j$, then accept j and let $t_l(j)$ be the empty time slot with the largest index in $[a_j + (h_l(j)-1)s_j, a_j + h_l(j)s_j - 1]$. Put packet j into $t_l(j),\ \forall l \in P_j$;
3) Any accepted packet j is transmitted through link $l \in P_j$ at time slot $t_l(j)$.

For each $l \in P_j$, the calculation of the term $\sum_{i\le j} \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,j)$ only requires the information of packets that may route through link l. It is also easy to see that the calculation of the cost term does not require future information for times after the arrival of each packet j, i.e., Algorithm 1 is on-line. Furthermore, $\lambda_l(i, j+1),\ i \le j$ is calculated from $\lambda_l(i,j)$ using Equation (7) when packet j is processed by Algorithm 1, and Equation (6) is only used to calculate $\lambda_l(j,j)$ upon j's arrival.

The basic intuition behind our algorithm is simple: We first allocate the end-to-end delay of each packet evenly over the links along its path. The algorithm then schedules the transmission for an accepted packet in a slot within its allocated time region at each link. Consequently, the end-to-end deadline constraint is met. With this approach, the natural questions are: (1) When a packet j is accepted, is there always an empty (non-reserved) slot in $[a_j + (h_l(j)-1)s_j, a_j + h_l(j)s_j - 1],\ \forall l \in P_j$ so that we can reserve a slot $t_l(j)$ for j at link l? (2) What is the performance of Algorithm 1 compared to the optimal off-line algorithm? We answer these questions in the following theorem. Note that we measure the performance in terms of the *competitive ratio*: if r is the competitive ratio achieved by our algorithm, then $R \ge rR^*$, where R is the weighted revenue of successful reception achieved by Algorithm 1 and R^* is the weighted revenue of successful reception achieved by the optimal off-line algorithm.

Theorem 3: If the arrival sample satisfies $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$, where $P_M = \max_i |P_i|$ is the maximum route length, $s_m = \min_i s_i$ is the minimum average slack time, $s_M = \max_i s_i$ is the maximum average slack time, $\rho_m = \min_i \rho_i$ is the minimum weight and $\rho_M = \max_i \rho_i$ is the maximum weight among all packets, then every packet accepted by Algorithm 1 reaches its destination before its deadline. Furthermore, Algorithm 1 achieves competitive ratio $r \triangleq \left[2\left(2\frac{S_M}{s_m}+1\right)\left(1+s_M\frac{\rho_M}{\rho_m}\right)\log(\mu)+1\right]^{-1}$ with $S_M = \max_i S_i$ as the maximum possible delay per hop among all packets.

Hence, our algorithm is $O(P_M \log P_M)$-competitive, where P_M is the maximum route length. The assumption $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$ in Theorem 3 imposes an upper bound on the maximum route length P_M for the validity of the provided competitive ratio. Roughly, P_M must be upper bounded by an exponential function of the minimum average slack time s_m. In Section V, we can see from the numerical examples that our algorithm achieves a high performance relative to the optimal off-line algorithm, even when this condition is not satisfied.

To prove this theorem, we first make the following transformation. With condition $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$, we can choose μ so that $\log(\mu) \le s_m$ and $\mu > 2\frac{\rho_M}{\rho_m}s_M P_M + 1 \ge 1$, i.e., $\frac{\mu-1}{2P_M} > \frac{\rho_M}{\rho_m}s_M \ge s_M$. Then, we make the following transformation. First, we choose a factor, $F \in \left[\frac{2P_M s_M}{\rho_m}, \frac{\mu-1}{\rho_M}\right)$, and normalize the weight ρ_i for all i with factor F and use $F\rho_i$, instead of ρ_i for all i in the problem (the objective is still equivalent to the original one). With this change, we have $\log(\mu) \le s_m \le s_M \le \frac{\rho_m}{2P_M} \le \frac{\rho_M}{2P_M} < \frac{\mu-1}{2P_M}$, i.e.,

$$I_l(i,j) \le 1 \le \frac{s_m}{\log(\mu)} \le \frac{s_i}{\log(\mu)},\ \forall i,j; \tag{11}$$

$$2|P_j| s_M \le 2P_M s_M \le \rho_m \le \rho_j,\ \forall j; \tag{12}$$

$$\rho_j \le \rho_M < \mu - 1,\ \forall j. \tag{13}$$

We need the following lemmas to prove Theorem 3:

Lemma 2: If j is accepted by Algorithm 1, then there exists at least one time slot in the interval $[a_j + (h_l(j)-1)s_j, a_j + h_l(j)s_j - 1]$ that has not been reserved by other accepted packets for each $l \in P_j$.

Lemma 3: Let $\mathcal{Q}$ denote the set of packets that are rejected by Algorithm 1 but successfully received by the optimal off-line algorithm, then $\sum_{j\in\mathcal{Q}} \rho_j \le \left(2\frac{S_M}{s_m}+1\right)\sum_l \sum_i c_l(i,K)$, where K is the last packet of the arrival sample.

Lemma 4: Let $\mathcal{A}$ denote the set of packets that are accepted by Algorithm 1, then $\sum_l \sum_i c_l(i,K) \le 2\left(1+s_M\frac{\rho_M}{\rho_m}\right)\log(\mu)\sum_{j\in\mathcal{A}}\rho_j$.

Proofs of Lemmas 2, 3 and 4 can be found in Appendix B, C and D, respectively.

Proof of Theorem 3: From Lemma 2, we see that for every accepted packet j, there exists an unreserved time slot $t_l(j)$ in the interval $[a_j + (h_l(j)-1)s_j, a_i + h_l(j)s_j - 1]$ for transmission at any $l \in P_j$. By the way of allocating the total slack time on each hop, it is apparent that j can reach its destination before deadline d_j if it is transmitted through l in slot $t_l(j)$ for all $l \in P_j$.

Lemma 2 completes the proof of the first part of the theorem. The remaining part is proved in two steps: First we upper bound the weighted revenue of packets that are rejected by our on-line algorithm but have successful receptions by the optimal off-line algorithm as in Lemma 3. We then lower bound the weighted revenue of packets that are accepted by our on-line algorithm as in Lemma 4. Combining the Lemmas 2, 3 and 4, we have

$$\begin{aligned} R^* &\le \sum_{j\in\mathcal{Q}}\rho_j + \sum_{j\in\mathcal{A}}\rho_j \\ &\le \left(2\frac{S_M}{s_m}+1\right)\sum_l\sum_i c_l(i,K) + \sum_{j\in\mathcal{A}}\rho_j \\ &\le \left[2\left(2\frac{S_M}{s_m}+1\right)\left(1+s_M\frac{\rho_M}{\rho_m}\right)\log(\mu)+1\right]\sum_{j\in\mathcal{A}}\rho_j \\ &= \left[2\left(2\frac{S_M}{s_m}+1\right)\left(1+s_M\frac{\rho_M}{\rho_m}\right)\log(\mu)+1\right]R = \frac{R}{r}. \end{aligned}$$

Note that $s_m \geq 1$ for a slotted system. From Eqs. (4) and (10), we have $S_i \leq (s_i+1)|P_i| - |P_i| + 1 \leq s_i|P_i| + 1$ and then $S_M \leq s_M P_M + 1$. Recall that $\mu > 2\frac{\rho_M}{\rho_m} s_M P_M + 1$. By letting $\mu = 2\frac{\rho_M}{\rho_m} s_M P_M + 1 + \epsilon$, where $\epsilon > 0$ can be arbitrarily small, we have $r \geq \left[2\big(2(s_M P_M + 1) + 1\big)\big(1 + \frac{\rho_M}{\rho_m} s_M\big) \log\big(2\frac{\rho_M}{\rho_m} s_M P_M + 1 + \epsilon\big) + 1\right]^{-1}$. Hence, our algorithm is $O(P_M \log P_M)$-competitive. ■

Our competitive ratio based framework of packet scheduling with deadlines is motivated by the competitive routing model [17]. However, there are significant differences between the two models: 1) The amount of link bandwidth is fixed and known at the beginning in the routing model, but the time slot resources are related to the accepted packets and are then related to the algorithm itself; 2) An earlier packet may use the resource of a later packet, which brings difficulties to making online decisions; 3) There are multiple ways of allocating the total end-to-end delay $d_j - a_j + 1$ of each packet j on each hop, which brings more complexities.

In Algorithm 1, we allocate the total end-to-end delay of each packet evenly on each hop, i.e., every accepted packet j reserves slot $t_l(j)$ in $[a_j + (h_l(j)-1)s_j, a_j + h_l(j)s_j - 1]$ at link $l \in P_j$. We briefly explain the reason for evenly allocating the total end-to-end delay on each hop as follows: Suppose we use a general way of allocating the total end-to-end delay of packet j. Let $s_{l,j}$ denote the slack time allocated to link $l \in P_j$, then we always have $\sum_{l \in P_j} s_{l,j} = d_j - a_j + 1$. Note that the maximum allowable delay on each hop S_j remains unchanged and $S_M = \max_j S_j$. However, the maximum and minimum per hop slack time are changed to $s_M = \max_j \max_{l \in P_j} s_{l,j} \geq \max_j s_j$ and $s_m = \min_j \min_{l \in P_j} s_{l,j} \leq \min_j s_j$ since $\min_{l \in P_j} s_{l,j} \leq s_j = \lfloor \frac{d_j - a_j + 1}{|P_j|} \rfloor \leq \max_{l \in P_j} s_{l,j}$. This means allocating the total slack time evenly will lead to smallest s_M and $\frac{S_M}{s_m}$, i.e., largest r (better lower bound) as stated in Theorem 3 among all methods of allocating the slack time.

Note that r characterizes the worst-case performance ratio between our online algorithm and the optimal off-line algorithm. When the route length P_M, the average slack times s_M and the maximum allowable per hop delay to the smallest average slack time ratio $\frac{S_M}{s_m}$ are larger, an accepted packet has more freedom to reserve a transmission slot and is more likely to deviate from the off-line algorithm while the online algorithm has no information of future arrivals to help make the decisions. Similarly, if the ratio of weights $\frac{\rho_M}{\rho_m}$ is large, then the difference of the contribution of each packet becomes large and the online decision becomes more difficult. Therefore, when these parameters increase, r decreases and the worst-case performance of the online algorithm tends to become worse. From the discussion in Section III, in a general wired topology, there is usually no online algorithms with competitive ratio 1. It is then interesting to ask: What is the largest achievable value for r? Is $O(P_M \log P_M)$-competitive already the best an online algorithm can do under a general wired network topology? In the literature of competitive analysis [3, 17], the upper bound of competitive ratio is usually derived by a converse via an adversary argument. We leave these questions to our future work. Although the theoretical worst-case ratio can be small, in Section V, we see from the numerical examples that by appropriately choosing the control parameter $\log(\mu)$ and normalizing the weights $\rho_i,\ i = 1, 2, \ldots, K$, our algorithm achieves good performance compared to the optimal off-line algorithm in practice.

V. Numerical Examples

We first consider an uplink wired tree shown in Fig. 3 (a) with homogeneous link rates. There are 10000 packets with homogeneous weights and the inter-arrival times of packets are chosen to be 0 w.p. $\frac{1}{2}$ and 1 time slot w.p. $\frac{1}{2}$. We generate the initial slack time (the difference between the deadline and the arrival time, increased by 1) uniformly at random between 24 and 30. The source node of each packet is chosen uniformly at random over all non-root nodes. It is easy to see that the maximum route length $P_M = 3$ for all packets in this network. Thus, the maximum average slack time is $s_M = 30$ and the minimum average slack time is $s_m = 8$, i.e., $P_M = 3 < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)} = \frac{255}{60}$. As given in the proof of Theorem 3, we choose the control parameter $\log(\mu) = 7.5$ so that $\log\big(2\frac{\rho_M}{\rho_m} s_M P_M + 1\big) = 7.4 < \log(\mu) \leq s_m = 8$. From Fig. 4 (a), we can see that by increasing the normalization factor of the packets' weight, the performance of our online algorithm increases and achieves more than 90% of the optimal off-line algorithm (the optimal value is obtained using the algorithm in Theorem 1). When the normalized weight is large enough, almost all packets are admitted and this result shows that admission control is only necessary in the proof of worst-case lower bound and our algorithm can still be efficient without the admission control component in practice.

Similarly, for another choice of slack times, generated uniformly at random between 0 and 5, when the condition $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$ is violated, our online algorithm still achieves about 80% of the optimal algorithm.

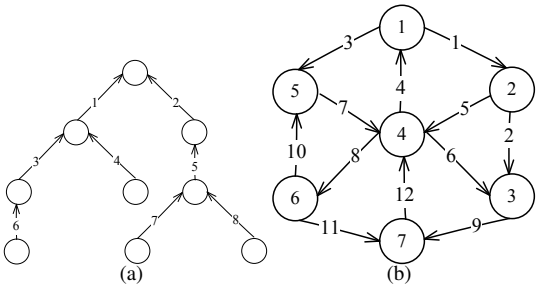

Fig. 3. Example topologies: (a) An Uplink Wired Tree, and (b) A General Wired Mutlihop Network with Multiple Source-Destination Flows

We then consider a general topology as shown in Fig. 3 (b). There are 10000 packets and the inter-arrival times of packets are chosen to be 0 w.p. $\frac{1}{2}$ and 1 time slot w.p. $\frac{1}{2}$. The packets weights are generated uniformly at random between 1 and 100 with $\frac{\rho_M}{\rho_m} = 100$. The given route P_i of each packet i is randomly generated with $1 \leq |P_i| \leq 3$ in the setup stage. We

first generate the initial slack time uniformly at random between 45 and 50 so that the condition $P_M = 3 < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)} = \frac{2^{15}-1}{10000}$ is satisfied. Note that the theoretical region of the control parameter μ is $\log\left(2\frac{\rho_M}{\rho_m}s_M P_M + 1\right) = 11.55 < \log(\mu) \leq s_m = 15$ in this example. With normalized packets weights (normalization factor is 300), we simulate the generated revenue as a function of $\log(\mu)$. We see from Fig. 4 (b) that our online algorithm achieves a revenue close to the upper bound of the optimal off-line algorithm (we compare our scheme with the revenue upper bound, i.e., the total normalized revenue, rather than the actual revenue of the off-line algorithm due to the extremely high complexity of the calculation of the off-line algorithm). Note that $\log(\mu)$ is in the denominator of the competitive ratio by Theorem 3, so this result is consistent with the theorem.

Similarly, for another choice of slack times, generated uniformly at random between 0 and 5, when the condition $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$ is violated, our online algorithm still achieves more than 80% of the upper bound with appropriately chosen $\log(\mu)$.

From examples (a) and (b), we can see that the assumption $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$, the theoretical region of μ, the weight normalization factor and the admission control component are only needed to prove the worst-case bound. In practice while the theoretical constraints are relaxed, the control parameters can be appropriately tuned for different scenarios so that our algorithm has efficient performance.

Finally, we consider a line network $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ with four nodes and 3 links (identical link rates of 1 packet/time slot). The arrival sample is periodic with 6 slots as a period. In slot 1, packet k_1 arrives at node 1 with destination 4, deadline 6 and weight 100, and k_2 arrives at node 1 with destination 2, deadline 1 and weight 90. In slot 2, k_3 arrives at node 1 with destination 2, deadline 2 and weight 1. In slot 3, k_4 arrives at node 1 with destination 2, deadline 3 and weight 1, k_5 arrives at node 2 with destination 4, deadline 5 and weight 200, and k_6 arrives at node 2 with destination 4, deadline 4 and weight 1. In slot 4, k_7 arrives at node 1 with destination 2, deadline 4 and weight 1, and k_8 arrives at node 2 with destination 4, deadline 5 and weight 50. This arrival pattern repeats every 6 slots, until a total of 10000 packets arrive. We use the normalization factor $F = 12$ for the packets weights in the algorithm. It is easy to calculate the optimal weighted revenue $\frac{10000}{8} * (100 + 90 + 200 + 50 + 1 + 1) * 12 = 6.6 \times 10^6$. We compare our online algorithm with two well known algorithms EDF and LWF. In EDF, each link transmits the packet with the earliest deadline among the packets with nonnegative slack time every time slot. In LWF, each link transmits the packet with the largest weight among the packets with nonnegative slack time every time slot. Note that the condition $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$ is already violated for this arrival sample. By choosing $\log(\mu) = 10$, we have that the total revenue of successful reception for our online algorithm is 5.9×10^6, for EDF is 5.3×10^6 and for LWF is 4.4×10^6. This means our algorithm outperforms EDF and LWF, and achieves about 90% of the off-line performance. This example also shows that our provably efficient online algorithm is more robust than EDF and LWF for a general network and arrival sample.

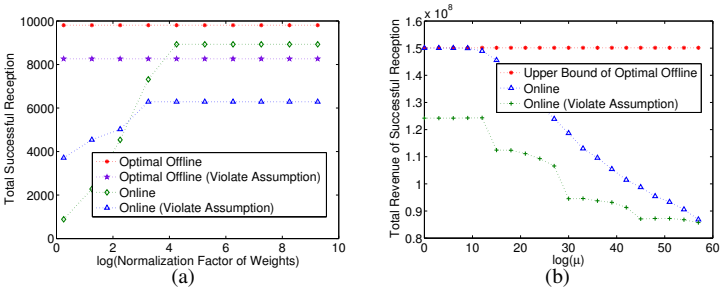

Fig. 4. (a) Performance of Online Algorithm with Different Normalized Packets Weights, and (b) Performance of Online Algorithm with Different $\log(\mu)$

VI. Conclusion

In this paper, we studied the packet scheduling problem with hard deadline constraints in wired networks. We first show that WC-EDF has the same performance as the optimal off-line algorithm in maximizing the throughput given any feasible arrival sample path for the uplink tree topology. We then proposed an on-line joint admission control and packet scheduling algorithm that requires only information on the route of each packet in the calculation and has provable competitive ratio to the optimal off-line algorithm in maximizing the weighted revenue. Furthermore, we show through numerical examples that our algorithm usually performs much better than the theoretical worst-case lower bound. As future directions, we will investigate whether the competitive ratio of our on-line algorithm is the highest achievable among all on-line algorithms for the general network topology. Furthermore, we are extending the competitive admission and packet scheduling framework to a wireless network setting, in which we need to take into account the scheduling constraints as well.

References

[1] C. L. Liu and J. W. Layland, "Scheduling Algorithms For Multiprogramming In a Hard-Real-Time Environment," *Journal of ACM*, vol. 20, pp. 46–61, 1973.

[2] M. Dertouzos, "Control Robotics: The Procedural Control of Physical Processes," in *IFIP Congress*, 1974, pp. 807–813.

[3] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang, "On The Competitiveness of On-Line Real-Time Task Scheduling," *Real-Time Systems*, vol. 4, pp. 125–144, 1992.

[4] G. Koren and D. Shasha, "Dover: An Optimal On-Line Scheduling Algorithm for Overloaded Uniprocessor Real-Time Systems," *SIAM Journal of Computing*, vol. 24, pp. 318–339, 1995.

[5] A. Bar-Noy, J. A. Garay, and A. Herzberg, "Sharing Video on Demand," *Discrete Applied Mathematics*, vol. 129, no. 1, pp. 3–30, 2003.

[6] M. Goldwasser and B. Kerbikov, "Admission Control with Immediate Notification," *Journal of Scheduling*, pp. 269–285, 2003.

[7] S. Chen, L. Tong, and T. He, "Optimal Deadline Scheduling with Commitment," *49th Allerton Conference on Communication, Control and Computing*, September 2011.

[8] J. Ding and G. Zhang, "Online Scheduling with Hard Deadlines on Parallel Machines," in *2nd AAIM*, 2006, pp. 32–42.

[9] J. Ding, T. Ebenlendr, J. Sgall, and G. Zhang, "Online Scheduling of Equal-Length Jobs on Parallel Machines," in *15th ESA*, 2007, pp. 427–438.

[10] T. Ebenlendr and J. Sgall, "A Lower Bound for Scheduling of Unit Jobs with Immediate Decision on Parallel Machines," in *6th WAOA*, 2008, pp. 43–52.

[11] N. Thibault and C. Laforest, "Online Time Constrained Scheduling with Penalties," in *23rd IEEE Int. Symposium on Parallel and Distributed Processing*, 2009.

[12] S. P. Y. Fung, "Online Preemptive Scheduling with Immediate Decision or Notification and Penalties," in *the 16th Annual International Conference on Computing and Combinatorics, ser. COCOON10*, 2010, pp. 389–398.

[13] T. Ling and N. B. Shroff, "Scheduling Real-Time Traffic in ATM Networks," in *Proc. of IEEE INFOCOM*, March 1996, pp. 198–205.

[14] B. Hajek, "On the Competitiveness of On-Line Scheduling of Unit-Length Packets with Hard Deadlines in Slotted Time," in *Conference on Information Sciences and Systems*, Johns Hopkins University, March 21-23 2001, pp. 434–439.

[15] B. Hajek and P. Seri, "Lex-Optimal Online Multiclass Scheduling with Hard Deadlines," *Mathematics of Operations Research*, vol. 30, no. 3, pp. 562–596, August 2005.

[16] P. P. Bhattacharya, L. Tassiulas, and A. Ephremides, "Optimal Scheduling with Deadline Constraints in Tree Networks," *IEEE/ACM Transactions on Automatic Control*, vol. 42, no. 12, pp. 1703–1705, December 1997.

[17] B. Awerbuch, Y. Azar, and S. Plotkin, "Throughput Competitive Online Routing," in *Proc. of the 34th Annual Symposium on Foundations of Computer Science*, November 1993, pp. 32–40.

[18] S. Hariharan and N. B. Shroff, "Maximizing Aggregated Information in Sensor Networks Under Deadline Constraints," *IEEE/ACM Transactions on Automatic Control*, vol. 56, no. 10, pp. 2369–2380, 2011.

[19] S. S. Panwar, D. Towsley, and J. K. Wolf, "Optimal Scheduling Policies for a Class of Queues with Customer Deadlines to the Beginning of Service," *Journal of ACM*, vol. 35, no. 4, pp. 832–844, 1988.

[20] S. Shakkottai and R. Srikant, "Scheduling Real-Time Traffic with Deadlines Over a Wireless Channel," *Wireless Networks*, vol. 8, no. 1, pp. 13–26, January 2002.

[21] A. Dua and N. Bambos, "Downlink Wireless Packet Scheduling with Deadlines," *IEEE Transactions on Mobile Computing*, vol. 6, no. 12, pp. 1410–1425, December 2007.

[22] I. H. Hou and P. R. Kumar, "Scheduling Heterogeneous Real-Time Traffic over Fading Wireless Channels," in *Proc. of IEEE INFOCOM*, San Diego, CA, March 2010.

[23] V. Raghunathan, V. Borkar, M. Cao, and P. R. Kumar, "Index Policies for Real-Time Multicast Scheduling for Wireless Broadcast Systems," *Proc. of IEEE INFOCOM*, pp. 1570–1578, April 2008.

[24] Q. Liu, X. Wang, and G. B. Giannakis, "A Cross-Layer Scheduling Algorithm with QoS Support in Wireless Networks," *IEEE/ACM Transactions on Vehicle Technology*, vol. 55, no. 3, pp. 839–847, May 2006.

[25] J. J. Jaramillo and R. Srikant, "Optimal Scheduling for Fair Resource Allocation in Ad Hoc Networks with Elastic and Inelastic Traffic," *IEEE/ACM Transactions on Networking*, 2011.

[26] Z. Mao, C. E. Koksal, and N. B. Shroff, "Online packet scheduling with hard deadlines in wired networks," Tech. Rep., 2012. [Online]. Available: http://www.ece.osu.edu/~maoz/Infocom2013.pdf

APPENDIX A
PROOF OF THEOREM 2

We provide an outline of the proof here. For a more detailed version, we refer the readers to [26].

Let policy p^* be a work conserving optimal off-line policy. Suppose T_0 is the first time slot in which policy p^* differs from WC-EDF p. Then, all the nodes in the network have the same set of packets in all slots before slot T_0. We show that WC-EDF p is optimal for any under-loaded arrival sample by induction. Choose $T-1 > T_0$ and let the following holds as hypothesis $\forall t \in (T_0, T-1]$:

1) There is no packet expiring under p at the end of slot t;

2) The total number of packets at any node n under policy p^* and p are the same at the end of slot t. Any packet k at any node n under policy p can be paired with a packet k^* at the same node n under policy p^* at the end slot t;

3) For any packet pair (k_0^*, k_0) at any node n at the end of slot t, there is a sequence of packet pairs $(k_0^*, k_0), \ldots, (k_i^*, k_i)$ at node n and its descendent nodes so that $d_{k_1^*} \leq d_{k_0}, \ldots, d_{k_i^*} \leq d_{k_{i-1}}, d_{k_0^*} \leq d_{k_i}$ if $i \geq 1$ and $d_{k_0^*} \leq d_{k_0}$ if $i = 0$.

It is easy to show that the base case holds (details in [26]) and we need to show that hypothesis 1)-3) hold for T. At the beginning of slot T, all nodes have the same amount of packets under p^* and p by the hypothesis. We use the following rule to pair the transmitted and remaining packets: If (k^*, k) forms a packet pair at the beginning of slot T, and k^*, k are transmitted by node n in slot T under policy p^* and p, respectively, then (k^*, k) still forms a packet pair after transmission. If (k^*, k) forms a packet pair at the beginning of slot T, k^* is transmitted by node n in slot T under p^* but k is not transmitted by p, since the total number of transmitted packets are the same for both policies, there must exist another packet pair (q^*, q) at node n so that q is transmitted by node n in slot T under p but q^* is not transmitted p^*, then (k^*, q) is a transmitted packet pair by node n in slot T and (q^*, k) is a remaining packet pair after transmission. New arrivals form common packet pairs. Consider an arbitrary node n and suppose the capacity of the link between node n and its parent node is 1 without loss of generality. If the link capacity is c, then repeat the same arguments c times. Let k^* and q denote the transmitted packets by node n under policy p^* and p, respectively. Without loss of generality, we assume (k^*, k) and (q^*, q) form different packet pairs at the beginning of slot T, i.e., the end of slot $T-1$. Consider any packet pair (k_0^*, k_0) (assume this packet pair is at an arbitrary node m) in the network and by hypothesis 3) for slot $T-1$, (k_0^*, k_0) has a sequence of packet pairs $(k_0^*, k_0), \ldots, (k_i^*, k_i)$ at node m and its descendent nodes at the beginning of slot T so that $d_{k_1^*} \leq d_{k_0}, \ldots, d_{k_i^*} \leq d_{k_{i-1}}, d_{k_0^*} \leq d_{k_i}$ if $i \geq 1$ and $d_{k_0^*} \leq d_{k_0}$ if $i = 0$. We have the following cases:

I) The sequence $(k_0^*, k_0), \ldots, (k_i^*, k_i)$ does not contain the packet pairs (k^*, k) and (q^*, q), then the reforming of (k^*, k) and (q^*, q) and the transmission of (k^*, q) do not influent the packet pair (k_0^*, k_0);

II) The sequence $(k_0^*, k_0), \ldots, (k_i^*, k_i)$ contains the packet pair (k^*, k) but does not contain (q^*, q). This means node n is node m or a descendent node of node m. By hypothesis 3) for packet pair (q^*, q) in slot $T-1$, there is a sequence $(q^*, q), \ldots, (q_j^*, q_j)$ at node n and its descendent nodes so that $d_{q_1^*} \leq d_q, \ldots, d_{q_j^*} \leq d_{q_{j-1}}, d_{q^*} \leq d_{q_j}$. Without loss of generality, let $(k_0^*, k_0), \ldots, (k^*, k), \ldots, (k_i^*, k_i)$ denote the sequence of (k_0^*, k_0) that contains (k^*, k), then $(k_0^*, k_0), \ldots, (k^*, q), (q_1^*, q_1), \ldots, (q_j^*, q_j), (q^*, k), \ldots, (k_i^*, k_i)$ is a sequence of packet pairs at node m and its descendent nodes so that $d_{k_1^*} \leq d_{k_0}, \ldots, d_{q_1^*} \leq d_q, \ldots, d_{q^*} \leq d_{q_j}, \ldots, d_{k_i^*} \leq d_{k_{i-1}}, d_{k_0^*} \leq d_{k_i}$. If node n is a descendent node of node m, then after the transmission of packet pair (k^*, q), hypothesis 3) holds for (k_0^*, k_0). If node n is node m, then packets k_0 and q are both at node n at the beginning

of slot T, and $d_q \leq d_{k_0^*}$ by EDF. Then, $d_{q_1^*} \leq d_q \leq d_{k_0^*}$ and $(k_0^*,k_0),(q_1^*,q_1),\ldots,(q_j^*,q_j),(q^*,k),\ldots,(k_i^*,k_i)$ is a sequence of packet pairs at node n and its descendent nodes that satisfies hypothesis 3) after transmission of (k^*,q). Similarly, if the sequence $(k_0^*,k_0),\ldots,(k_i^*,k_i)$ contains the packet pair (q^*,q) but does not contain (k^*,k), hypothesis 3) also holds for (k_0^*,k_0) at the end of slot T;

III) The sequence $(k_0^*,k_0),\ldots,(k_i^*,k_i)$ contains the packet pairs (k^*,k) and (q^*,q). Similar to case II) (details are in [26]).

Therefore, by repeating the above argument for all transmitted packet pairs, hypothesis 3) holds for all packet pairs at the end of slot T. Suppose a packet k at any node n is expiring under policy p at the end of slot T, then there is another packet k^* with $d_{k^*} \leq d_k$ and k^* is at node n or its descendent node by hypothesis 3), i.e., k^* is also expiring which contradicting the optimality of policy p^*. Therefore, there is no packet expiring under p at the end of slot T, i.e., hypothesis 1) holds. Note that the total number of transmitted packets and received packets are the same under both policies in the wired uplink tree network and no packets expire under both policies, then the total number of packets at any node under both policies are the same at the end of slot T, i.e., hypothesis 2) holds.

Appendix B
Proof of Lemma 2

Suppose j is the first packet that is accepted by Algorithm 1 but there exists $l \in P_j$ so that all time slots in the interval $[a_j+(h_l(j)-1)s_j, a_j+h_l(j)s_j-1]$ are occupied by other accepted packets when j is accepted. From Eqs. (3) and (6), we have $\lambda_l(j,j)=1$ and $\frac{c_l(j,j)}{s_j}=\mu^{\lambda_l(j,j)}-1 \geq \mu-1$. Note that $\tilde{I}_l(j,j)=1$ by Equation (9), combined with Equation (13), we then have $\sum_{l'}\sum_{i'\leq j}\frac{\tilde{I}_{l'}(i',j)}{s_{i'}}c_{l'}(i',j) \geq \frac{\tilde{I}_l(j,j)}{s_j}c_l(j,j) = \frac{c_l(j,j)}{s_j} \geq \mu-1 > \rho_j$, i.e., j is rejected by Algorithm 1 which is a contradiction.

Appendix C
Proof of Lemma 3

For any $j \in \mathcal{Q}$, from Algorithm 1 and the fact that $c_l(i,j)$ is increasing in j, given any i and l, i.e., $c_l(i,j) \leq c_l(i,k),\ \forall l,i$ if $j \leq k$, we have

$$\begin{aligned}\rho_j <& \sum_l \sum_{i\leq j} \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,j) \leq \sum_l \sum_i \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,j)\\ \leq& \sum_l \sum_i \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,K).\end{aligned}$$

Consider any packet i and any link $l \in P_i$, if i is rejected by Algorithm 1, then $\tilde{I}_l(i,j)=0,\ \forall j \neq i$ and $\tilde{I}_l(i,i)=1$, we then have $\sum_{j\in\mathcal{Q}}\frac{\tilde{I}_l(i,j)}{s_i} \leq \frac{1}{s_i}$; otherwise, we have $\tilde{I}(i,j)=1$ only for j with $a_i+(h_l(i)-1)s_i \leq d_j-|P_j|+h_l(j)$ and $a_j+h_l(j)-1 \leq a_i+h_l(i)s_i-1$. Let $t_l^*(j)$ be the time slot in which packet j is transmitted through link $l \in P_j$ under the optimal off-line algorithm. Note that $t_l^*(j) \in [a_j+h_l(j)-1, d_j-|P_j|+h_l(j)]$ since this interval is the maximum allowable transmission interval for the successful reception of j under all algorithms. Furthermore, if $t_l^*(j) < a_i+(h_l(i)-1)s_i-S_j+1$ or $t_l^*(j) > a_i+h_l(i)s_i-1+S_j-1$, then it means $d_j-|P_j|+h_l(j) \leq t_l^*(j)+S_j-1 < a_i+(h_l(i)-1)s_i$ or $a_j+h_l(j)-1 \geq t_l^*(j)-S_j+1 > a_i+h_l(i)s_i-1$, i.e., $\tilde{I}(i,j)=0$. Therefore, we must have $t_l^*(j) \in [a_i+(h_l(i)-1)s_i-S_j+1, a_i+h_l(i)s_i-1+S_j-1]$ if $\tilde{I}(i,j)=1$, then $\sum_{j\in\mathcal{Q}}\frac{\tilde{I}_l(i,j)}{s_i} \leq \frac{2S_j+s_i-2}{s_i}$. By summing over all $j \in \mathcal{Q}$ and combining with the fact $\tilde{I}_l(i,j)=0,\ \forall l \notin P_i$, we have

$$\begin{aligned}\sum_{j\in\mathcal{Q}}\rho_j <& \sum_l\sum_i c_l(i,K)\sum_{j\in\mathcal{Q}}\frac{\tilde{I}_l(i,j)}{s_i}\\ \leq& \left(2\frac{S_M}{s_m}+1\right)\sum_l\sum_i c_l(i,K),\end{aligned}$$

where $S_M=\max_i S_i$ is the maximum link delay among all packet and $s_m=\min_i s_i$ is the minimum average slack time among all packets.

Appendix D
Proof of Lemma 4

Note that for any l,i and $j \in \mathcal{A}$, we have

$$\begin{aligned}c_l(i,j+1)-c_l(i,j) =& s_i\mu^{\lambda_l(i,j)}\left(\mu^{\frac{I_l(i,j)}{s_i}}-1\right)\\ =& s_i\mu^{\lambda_l(i,j)}\left(2^{\log(\mu)\frac{I_l(i,j)}{s_i}}-1\right)\\ \leq& \mu^{\lambda_l(i,j)}I_l(i,j)\log(\mu)\\ =& \left(\frac{c_l(i,j)}{s_i}+1\right)I_l(i,j)\log(\mu), \quad (14)\end{aligned}$$

where the inequality is from Equation (11) and the fact $2^x-1 \leq x,\ x\in[0,1]$. The last equality is from Equation (5).

Recall that $I_l(i,j) \leq \tilde{I}_l(i,j),\ \forall i,j,l$ by Equation (3) and Equation (8); $c_l(i,j) \leq c_l(i,i),\ \forall i > j$ by Equation (5), Equation (7) and Equation (3); and $\sum_l\sum_{i\leq j}\frac{\tilde{I}_l(i,j)}{s_i}c_l(i,j) \leq \rho_j,\ \forall j \in \mathcal{A}$ from Algorithm 1. From Equation (3), we have $\sum_i I_l(i,j)=\sum_{i\in\mathcal{A}}I_l(i,j),\ \forall j\in\mathcal{A}$. From Lemma 2, $t_l(i) \in [a_i+(h_l(i)-1)s_i, a_i+h_l(i)s_i-1]$ for any $i \in \mathcal{A}$. Note that in order to have $I_l(i,j)=1$, we must have $a_i+(h_l(i)-1)s_i \leq t_l(j) \leq a_i+h_l(i)s_i-1$. For any $i\in\mathcal{A}$, if $t_l(i) < t_l(j)-s_i+1$ or $t_l(i) > t_l(j)+s_i-1$, then it means $a_i+h_l(i)s_i-1 \leq t_l(i)+s_i-1 < t_l(j)$ or $a_i+(h_l(i)-1)s_i \geq t_l(i)-s_i+1 > t_l(j)$, i.e., $I_l(i,j)=0$. Therefore, if $I_l(i,j)=1,\ \forall i\in\mathcal{A}$, then $t_l(i) \in [t_l(j)-s_i+1, t_l(j)+s_i-1]$, i.e., $\sum_l\sum_i I_l(i,j)=\sum_l\sum_{i\in\mathcal{A}}I_l(i,j) \leq 2|P_j|s_i \leq 2|P_j|s_M \leq \rho_j$ using Equation (12), and $\sum_{i\in\mathcal{A}}\frac{\rho_i}{\rho_j}I_l(i,j) \leq 2\frac{\rho_M}{\rho_m}s_M$. For any l,i and $j\in\mathcal{A}$, by summing over l and i of Equation (14), we have $\sum_l\sum_i\left[c_l(i,j+1)-c_l(i,j)\right] \leq 2\left(1+s_M\frac{\rho_M}{\rho_m}\right)\log(\mu)\rho_j$ (detailed steps can be found in [26]). Combined with $c_l(i,j+1)-c_l(i,j)=0,\ \forall j\notin\mathcal{A}$, we have

$$\begin{aligned}\sum_l\sum_i c_l(i,K) =& \sum_l\sum_i\sum_{j\in\mathcal{A}}\left[c_l(i,j+1)-c_l(i,j)\right]\\ \leq& 2\left(1+s_M\frac{\rho_M}{\rho_m}\right)\log(\mu)\sum_{j\in\mathcal{A}}\rho_j.\end{aligned}$$