Online Routing and Scheduling With Capacity Redundancy for Timely Delivery Guarantees in Multihop Networks

Han Deng*, Tao Zhao*, and I-Hong Hou

***Abstract*—It has been shown that it is impossible to achieve stringent timely delivery guarantees in a large network without having complete information of all future packet arrivals. In order to maintain desirable performance in the presence of uncertainty of future, a viable approach is to add redundancy by increasing link capacities. This paper studies the amount of capacity needed to provide stringent timely delivery guarantees. We propose a low-complexity online algorithm and prove that it only requires a small amount of redundancy to guarantee the timely delivery of most packets. Further, we show that in large networks with very high timely delivery requirements, the redundancy needed by our policy is at most twice as large as a theoretical lower bound. For practical implementation, we propose a distributed protocol based on this centralized policy. Without adding redundancy, we further propose a low-complexity order-optimal online policy for the network. Simulation results show that our policies achieve much better performance than other state-of-the-art policies.**

***Index Terms*—Competitive ratio, cyber-physical systems, multihop networks, online algorithms, optimal scheduling.**

I. Introduction

Many emerging safety-critical applications for Internet of Things and Cyber-Physical Systems require communication protocols that support stringent timely delivery guarantees for packet transmissions in multihop networks. In a typical scenario, when sensors detect unusual events that can cause system instability, they send out this information to actuators or control centers. This information needs to be delivered within a strict deadline for actuators or control centers to resolve the unusual events. The system can suffer from a critical fault when a small portion of packets fail to be delivered in time.

Despite the huge literature on quality of service (QoS), there is little work that can provide stringent timely delivery guarantees, especially when packet arrivals are time-varying and unpredictable. The lack of progress is mainly caused by two fundamental challenges. On one hand, practical solutions need to rely on online policies that do not have knowledge of future packet arrivals and thus often suboptimal compared to offline policies. On the other hand, in a multihop network, the scheduling decision of one communication link will impact the decisions of subsequent links. The negative effects of suboptimal decisions by online policies therefore get accumulated along the path of multihop transmissions. In fact, a recent work by Mao *et al.* [2] has proved that the performance of any online policies deteriorates as the maximum route length in the network increases. As a result, no online policy can provide reasonable performance guarantees when the size of the network is large.

In order to achieve desirable performance using online policies, a viable approach is to add redundancy into the system. During system deployment, it is common practice to provision redundant capacities of communication links. Such redundancy can alleviate the negative impacts of suboptimal decisions by online policies. Using this approach, a critical question is to determine the amount of redundancy needed to provide the desirable performance guarantees. This paper aims to answer this question.

We first show that the problem of maximizing the number of timely packet deliveries can be formulated as a linear programming problem when one knows the complete knowledge of all future packet arrivals. In the setting of online policies, some of the parameters of this linear programming problem will only be revealed when the corresponding packets arrive. Therefore, online policies need to make routing and scheduling decisions for packets without knowing all parameters. On the other hand, we also observe that adding redundancy by increasing link capacities is equivalent to relaxing a subset of constraints in the linear programming problem. Based on these observations, we define a competitive ratio that, given the amount of redundancy, quantifies the relative performance of online policies in comparison to the optimal offline solution.

Using the primal-dual method [3], we propose an online policy that achieves good performance in terms of competitive ratio. This policy has several important features: First, when there is no redundancy added to the system, the performance of our online policy is asymptotically better than that of the recent work [2] when the size of the network increases. Second, we also show that only a small amount of redundancy is needed to achieve strict performance guarantees. Specifically, in order to guarantee the timely delivery of at least $1-\frac{1}{\theta}$ as many packets as the optimal solution in a network whose longest path has length L, our policy only needs to increase link capacities by

* These authors contributed equally to this work.

Tao Zhao and I-Hong Hou are with Department of ECE, Texas A&M University, College Station, Texas 77843-3128, USA. Email: {alick, ihou}@tamu.edu

Han Deng is with Houston Methodist Research Institute, Houston, TX, USA. Email: hdeng@houstonmethodist.org

This material is based upon work supported in part by NSF and Intel under contract number CNS-1719384, in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/Grant Number W911NF-18-1-0331, and in part by Office of Naval Research under Contract N00014-18-1-2048.

Part of this work has been presented at ACM SIGMETRICS 2017 [1].

at most $\ln L + \ln \theta$ times.[1] Finally, we also show that our policy can be implemented with very low complexity.

Next, we establish a theoretical lower bound of competitive ratio for all online policies. We show that, in order to guarantee a certain degree of performance, the redundancy needed by our policy is only a small amount away from the theoretical limit. In particular, when both L and θ go to infinity, the redundancy needed by our policy is at most twice as large as the theoretical limit.

To address the performance gap between the above online policy and the theoretical lower bound, we propose another online policy and prove that it is order optimal in the case where there is no capacity redundancy in the network. Specifically, we show that this online policy guarantees to deliver at least $\frac{1}{O(\log L)}$ as many packets before their deadlines as the optimal offline solution, where L is the maximum route length. As Mao *et al.* [2] has proved no online policy can deliver more than $\frac{1}{O(\log L)}$ packets without redundancy[2], our policy is order optimal.

Noting that both the above online policies are centralized algorithms[3], we also propose a distributed protocol that is inspired by the design principles of our centralized online policies. This distributed protocol only requires each node to broadcast its local load information infrequently, and therefore it only incurs a small amount of communication overhead. When a packet arrives at a source node, the source node determines a suggested route for the packet using its received load information. Each link on the route makes scheduling decisions solely based on its local information.

All our three policies are evaluated by simulations, and compared with the well-known earliest deadline first (EDF) policy and the online policy proposed by Mao *et al.* [2], which will be denoted by MKS afterwards. Simulation results show that all our policies perform better than the other two policies in most cases under various system settings.

The rest of the paper is organized as follows. Section II reviews related work. Section III introduces our system model and defines the competitive ratio. Section IV proposes our first online policy and studies its competitive ratio and computation complexity. Section V establishes a theoretical lower bound of competitive ratio. Section VI proposes an order-optimal policy and proves its competitive ratio. Section VII proposes a distributed protocol based on the intuitions of our centralized online policies. Section VIII provides simulations on our proposed policies and compare them with EDF and MKS. Finally, Section IX concludes this paper.

II. Related Work

Online scheduling in real-time environment has been extensively studied. Studies show that the earliest deadline first (EDF) algorithm [4] and the least laxity first (LLF) algorithm [5] achieve the same performance as the optimal offline algorithm in an underloaded uniprocessor system. Considering overload, Baruah *et al.* [5] proved that no online algorithm can guarantee to serve more than $1/4$ of the jobs that can be served by optimal offline algorithm and provided an algorithm in a uniprocessor system which achieves the $1/4$ bound. Goldman *et al.* [6], Goldwasser and Kerbikov [7], and Goldwasser [8] considered admission control in online scheduling. When all jobs have equal length, the best deterministic algorithm is $(1 + 1/(\lfloor k \rfloor + 1))$-competitive, where $k \geq 0$ denotes the willingness of a job to endure a delay before being served.

Besides the uniprocessor case, online scheduling with multiple servers has also been studied. Goldwasser and Pedigo [9] studied the scheduling of equal-length jobs on two identical machines. Ding and Zhang [10], Ding *et al.* [11], and Ebenlendr and Sgall [12] studied the case with parallel machines. The scheduler needs to decide whether to accept or reject a job and which machine is chosen to serve the job. Ding *et al.* [11] proposed an algorithm with immediate decision which has the optimal competitive ratio of 1.8 when there are two machines and approaches $\frac{\mathrm{e}}{\mathrm{e}-1}$-competitive as the number of machines increases. Later Ebenlendr and Sgall [12] showed that $\frac{\mathrm{e}}{\mathrm{e}-1}$ is the lower bound of all online algorithms with immediate decision when the number of machines approaches infinity.

There are also many works studying the scheduling problem in multihop networks. An early study by Andrews and Zhang [13] focuses on the problem of packet scheduling with arbitrary end-to-end delay, fixed route, and known packet injection rate. It proposed a distributed algorithm which achieves a certain delay bound. Bhattacharya *et al.* [14] studied the scheduling problem on a tree network. Packets arrive at an arbitrary node and they need to be transmitted to the root node before the deadlines. Therefore, this is also a fixed-route problem. The shortest time to extinction (STE) algorithm was proposed and it is shown to achieve the performance of the optimal offline policy. Hou [15] proposed a throughput-optimal policy for up-link tree networks with end-to-end delay constraints and delivery ratio requirement. Li and Eryilmaz [16] studied end-to-end deadline constrained flow scheduling in multihop wired networks. They assumed stochastic arrival processes and developed algorithms that exploit the freedom of choosing service disciplines. However, they only considered the fixed-route model and did not provide theoretical performance analysis. Singh and Kumar [17], [18] proposed decentralized scheduling policies which maximize the timely throughput for multihop wireless networks based on Markov Decision Process. However, their work considers average performance guarantees, rather than worst-case performance guarantees. Mao *et al.* [2] considered a fixed-route online scheduling problem. The network has arbitrary packet arrival and packets have different weights. The paper aims to maximize the total cumulative weights of packets that reach destination before their deadline. The paper proved that the competitive ratio of any online policy is no better than $O(\log L)$, where L is the maximum route length. It also proposed an online policy that is $O(L \log L)$-competitive.

Taking online routing into account, Li *et al.* [19] pro-

[1] When the optimal policy delivers all packets in time, θ is the loss frequency, i.e. the number of packet losses per unit time, under the online policy, and $1 - \frac{1}{\theta}$ is the delivery ratio, i.e. the percentage of delivered packets among all packets, of the online policy.

[2] The original paper contains an error that is fixed in http://newslab.ece.ohio-state.edu/research/resources/Mao_errata.pdf.

[3] We use the words "algorithm" and "policy" interchangeably in this paper.

posed using expected end-to-end delay for path selection in wireless mesh networks. Their work aims to minimize the average end-to-end delay, and cannot provide guarantees on per-packet delays. Liu and Yang [20] studied the multihop routing problem with end-to-end hard deadlines. They developed a distributed routing algorithm called spatial-temporal backpressure which can support any periodic traffic flows within the throughput region. Wang *et al.* [21] studied the problem of routing and scheduling in multihop wireless sensor networks to minimize channel usage with the constraint of end-to-end delay and proposed a sub-optimal algorithm to the NP-complete problem. Our work will focus on online routing and scheduling in multihop networks to guarantee strict timely delivery requirement for any possible sequence of packet arrivals.

There has been a line of research on online routing in networks with bounded node buffers. Aiello *et al.* [22] formalized the system model for store-and-forward routers with limited buffer sizes and analyzed the performance of various online algorithms. The competitive ratios of online algorithms have been subsequently improved on uni-directional grid networks [23], [24], [25]. In contrast, our work considers unbounded buffers and focuses on the requirement of link capacity redundancy to achieve strict timely delivery guarantees in a general network.

Besides, there are many works characterizing end-to-end delay in multihop networks. Rodoplu *et al.* [26] studied the problem of dynamic estimating end-to-end delay over multihop mobile wireless networks. Sanada *et al.* [27] used a Markov-chain model to study the string topology and analyzed the end-to-end throughput and delay. Jiao *et al.* [28] studied the problem of estimating the end-to-end delay distribution for general traffic arrival process and Nakagami-m channel model by analyzing packet delay at each hop. In our model, end-to-end delay is a hard requirement imposed by each arriving packet, and a successful delivery must occur within the preset deadline.

Our analysis follows the primal-dual method illustrated by Buchbinder and Naor [3]. They have used the method to study bi-criteria competitive algorithms for online routing [29]. The main difference is that we consider online scheduling with timely delivery guarantees in addition to online routing. Besides, we employ a different approach in analyzing the impact of capacity redundancy, leading to a more flexible tradeoff between redundancy and quality of service. Our approach has been successfully applied to online job allocation problems [30], [31].

III. System Model

We consider a network with multihop transmissions. The network is represented by a directed graph where each node represents a router and an edge from one node to another represents a link between the corresponding routers. Packets arrive at their respective source nodes following some unknown sequence. We use $\mathcal{M}$ to denote the set of all packets and $\mathcal{L}$ the set of all links. When a packet $m \in \mathcal{M}$ arrives at its source node s_m at time a_m, it specifies its destination node δ_m and a deadline f_m. The packet requests to be delivered to its destination at or before its specified deadline. Packets that are not delivered in time do not have any value, and can be dropped from the network. We aim to deliver as many packets in time as possible.

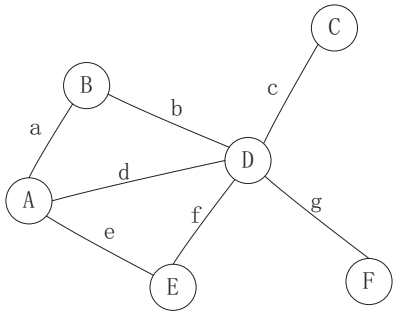


Fig. 1. An example network topology.

We assume that time is slotted and numbered by $t = \{1, 2, 3, \dots\}$. Different links in the network may have different link capacities, and we denote by C_l the number of packets that link l can transmit in a time slot. At the beginning of each time slot, each node decides which packets to transmit over its links, subject to capacity constraints of the links. Packets transmitted toward a node in time slot t will be received by that node at the end of the time slot, so that the node can transmit these packets to subsequent nodes starting from time slot $t + 1$.

Delivering a packet to its destination at or before its deadline requires determining two things: the route used to forward the packet from its source to its destination, and the times at which the packet is transmitted along its route. We define a *valid schedule* for each packet m as the collection of links of a route, as well as the times of transmissions for each of these links, so that packet m can be delivered to its destination in time. For example, consider the network shown in Fig. 1. Suppose a packet arrives at node A at time slot 1, and needs to be delivered to node F before the end of time slot 3. One valid schedule for this packet is to transmit it over link d in time slot 1, and then over link g in time slot 2. We use {(d, 1), (g, 2)} to represent this valid schedule. Other valid schedules include {(d, 1), (g, 3)}, {(e, 1), (f, 2), (g, 3)}, etc. On the other hand, {(d,1), (g,4)} is not a valid schedule because the packet is delivered to its destination after its deadline at time slot 3. The schedule {(d,3), (g,2)} is not valid because it would require node D to transmit the packet over link g at time slot 2 before it receives the packet at time slot 3. For each packet m, we let $V(m)$ denote the set of valid schedules for m. The problem of deciding how to deliver packets in time then becomes one of choosing valid schedules for packets.

We use $X_{mk} \in \{0, 1\}$ to indicate whether packet m chooses valid schedule $k \in V(m)$. $X_{mk} = 1$ if and only if packet m is transmitted using valid schedule k. We aim to maximize the total number of timely deliveries by deciding the values of all X_{mk}. Our optimization problem is formally as follows:

Schedule:

$$\max \quad \sum_{\substack{m\in\mathcal{M}\\ k\in V(m)}} X_{mk} \tag{1a}$$

$$\text{s.t.} \quad \sum_{k\in V(m)} X_{mk} \le 1, \quad \forall m \in \mathcal{M}, \tag{1b}$$

$$\sum_{(l,t)\in k} X_{mk} \le C_l, \quad \forall l \in \mathcal{L}, t \in \{1,2,\dots\}, \tag{1c}$$

$$X_{mk} \ge 0, \quad \forall m \in \mathcal{M}, k \in V(m). \tag{1d}$$

As shown in Eq. (1a), the objective is to maximize the total number of packets that are delivered in time. Eq. (1b) states that at most one valid schedule can be chosen for each packet. Eq. (1c) states that each link can transmit at most C_l packets in any time slot. In practice, X_{mk} can only be either 0 or 1, but our problem formulation allows X_{mk} to be any real number in $[0, 1]$. Thus, the optimal solution to **Schedule** describes an upper bound on the total number of timely deliveries.

If information of all packets is available when the system starts, the optimal solution to **Schedule** can be found by standard linear programming methods. In practice, however, packets arrive sequentially, and we need to rely on online policies that determines the values of X_{mk} for each arriving packet m without knowing future packet arrivals. Without the knowledge of future arrivals, it is obvious that online policies cannot always achieve the optimal solution to **Schedule**. In fact, Mao *et al.* [2] has shown that, when the maximum route length between a source node and a destination node is L, no online policy can guarantee to deliver more than $\frac{1}{O(\log L)}$ as many packets as the optimal solution. Therefore, when L is large, the performance of online policies can be unacceptable for virtually any applications.

In order to achieve good performance for online policies in the presence of unknown future arrivals, we consider the scenario where service providers can increase link capacities by, for example, upgrading network infrastructures. When the link capacities are increased by R times, link l can transmit RC_l packets in each time slot. With the capacity redundancy, our problem can be rewritten as follows:

Schedule(R):

$$\max \quad \sum_{\substack{m\in\mathcal{M}\\ k\in V(m)}} X_{mk} \tag{2a}$$

$$\text{s.t.} \quad \sum_{k\in V(m)} X_{mk} \le 1, \quad \forall m \in \mathcal{M}, \tag{2b}$$

$$\sum_{(l,t)\in k} X_{mk} \le RC_l, \quad \forall l \in \mathcal{L}, t \in \{1,2,\dots\}, \tag{2c}$$

$$X_{mk} \ge 0, \quad \forall m \in \mathcal{M}, k \in V(m). \tag{2d}$$

To evaluate the performance of online policies, we define a competitive ratio that incorporates capacity redundancy:

Definition 1: Given a sequence of packet arrivals, let Γ_{opt} be the optimal value of $\sum_{mk:k\in V(m)} X_{mk}$ in **Schedule**, and $\Gamma_\eta(R)$ be the number of packets that are delivered under an online policy η when the link capacities are increased by R times. The online policy η is said to be (R, ρ)-*competitive* if $\Gamma_{\text{opt}}/\Gamma_\eta(R) \le \rho$, for any sequence of packet arrivals.

Remark: Although the competitive ratio is defined against the relaxed problem **Schedule**, it is also guaranteed against the original binary linear problem, since Γ_{opt} is an upper bound on the optimal total number of timely deliveries.

IV. An Online Algorithm and Its Competitive Ratio

A. Algorithm Description

In this section, we propose an online policy based on primal-dual method [3] and analyze the competitive ratio. We first note that the dual problem of **Schedule** is:

Dual:

$$\min \quad \sum_m \alpha_m + \sum_{l,t} C_l \beta_{lt} \tag{3a}$$

$$\text{s.t.} \quad \alpha_m + \sum_{(l,t)\in k} \beta_{lt} \ge 1, \quad \forall m \in \mathcal{M}, k \in V(m), \tag{3b}$$

$$\alpha_m \ge 0, \quad \forall m \in \mathcal{M}, \tag{3c}$$

$$\beta_{lt} \ge 0, \quad \forall l \in \mathcal{L}, t \in \{1,2,\dots\}, \tag{3d}$$

where α_m is the Lagrange multiplier corresponding to constraint (1b), and β_{lt} is the Lagrange multiplier corresponding to constraint (1c).

By the Weak Duality Theorem, we have the following lemma:

Lemma 1: Given any vectors of $\{\alpha_m\}$ and $\{\beta_{lt}\}$ that satisfy the constraints (3b)–(3d), we have $\sum_m \alpha_m + \sum_{(l,t)} C_l \beta_{lt} \ge \Gamma_{\text{opt}}$.

We now introduce our online algorithm. Our algorithm constructs $\{X_{mk}\}, \{\alpha_m\}, \{\beta_{lt}\}$ simultaneously while ensuring they satisfy all constraints in **Schedule(R)** and **Dual**. As shown in Algorithm 1, initially it sets all $\alpha_m, \beta_{lt}, X_{mk}$ to be 0. When a packet m arrives, the algorithm finds the valid schedule k^* that has the smallest $\sum_{(l,t)\in k} \beta_{lt}$ among all $k \in V(m)$. Here $\sum_{(l,t)\in k} \beta_{lt}$ can be viewed as the total cost of delivering packet m using the schedule k. If the optimal cost $\sum_{(l,t)\in k^*} \beta_{lt} \ge 1$, then the algorithm drops packet m. On the other hand, if $\sum_{(l,t)\in k^*} \beta_{lt} < 1$, packet m is transmitted using the valid schedule k^*. Our algorithm then sets $X_{mk^*} = 1$, $\alpha_m = 1 - \sum_{(l,t)\in k^*} \beta_{lt}$, and updates β_{lt} as $\beta_{lt} = \beta_{lt}(1 + \frac{1}{C_l}) + \frac{1}{(d_l - 1)C_l}$ for all $(l, t) \in k^*$, where d_l is chosen to be $(1 + 1/C_l)^{RC_l}$. Note that our algorithm always produces integer solutions for X_{mk}.

B. Complexity of the Algorithm

In step 4, the algorithm needs to find the valid schedule k^* that minimizes the cost of delivering packet m. We now show that this step can be completed in polynomial time by dynamic programming. Before presenting the algorithm, recall that packet m joins the network at the beginning of time slot a_m, and specifies its deadline as the end of time slot f_m. Its source node and destination node are s_m and δ_m, respectively. Therefore, a valid schedule for m is one that can deliver a packet from node s_m to node δ_m between time slots a_m and f_m.

Let $\Theta(v, \tau)$ be the smallest cost of delivering packet m among all schedules $k \in V(m)$ that can deliver a packet from

Algorithm 1 Primal Dual (PD) Algorithm

1: $\alpha_m \leftarrow 0$, $\beta_{lt} \leftarrow 0$, $X_{mk} \leftarrow 0$
2: $d_l \leftarrow (1+1/C_l)^{RC_l}, \forall l$.
3: **for** each arriving packet m **do**
4: $k^* \leftarrow \text{argmin}_k \sum_{(l,t)\in k} \beta_{lt}$
5: **if** $\sum_{(l,t)\in k^*} \beta_{lt} < 1$ **then**
6: $\alpha_m \leftarrow 1 - \sum_{(l,t)\in k^*} \beta_{lt}$
7: $\beta_{lt} \leftarrow \beta_{lt}\left(1+\frac{1}{C_l}\right) + \frac{1}{(d_l-1)C_l}, \forall (l,t) \in k^*$
8: $X_{mk^*} \leftarrow 1$
9: Transmit packet m using valid schedule k^*.
10: **else**
11: Drop packet m.
12: **end if**
13: **end for**

Algorithm 2 Dynamic Programming

1: **for** each arriving packet m **do**
2: $\Theta(s_m, a_m - 1) \leftarrow 0$
3: $\Theta(v, a_m - 1) \leftarrow \infty, \forall v \neq s_m$
4: $S(v, a_m - 1) \leftarrow \phi, \forall v$
5: **for** $\tau = a_m$ **to** f_m **do**
6: **for** node v **do**
7: $\Theta(v,\tau) \leftarrow \Theta(v, \tau-1)$
8: $S(v,\tau) \leftarrow S(v, \tau-1)$
9: **for** link $l := (u,v) \in \mathcal{L}$ **do**
10: **if** $\Theta(u, \tau-1) + \beta_{l\tau} < \Theta(v,\tau)$ **then**
11: $\Theta(v,\tau) \leftarrow \Theta(u,\tau-1) + \beta_{l\tau}$
12: $S(v,\tau) \leftarrow S(u,\tau-1) \cup \{(l,\tau)\}$
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: **end for**

node s_m to node v between time slots a_m and τ. $\Theta(v,\tau) = \infty$ if there is no schedule that delivers a packet from s_m to v between time slots a_m and τ_m. Step 4 of Algorithm 1 is then equivalent to finding the valid schedule that achieves $\Theta(\delta_m, f_m)$. Since packet m arrives at the beginning of time slot a_m, or equivalently, at the end of time slot $a_m - 1$, we set $\Theta(s_m, a_m - 1) = 0$ and $\Theta(v, a_m - 1) = \infty, \forall v \neq s_m$.

There are only two different ways to deliver a packet to node v at or before the end of time slot τ: The first is to deliver the packet to v at or before the end of time slot $\tau - 1$, in which case the smallest cost is $\Theta(v, \tau-1)$. The second is to deliver the packet to one of v's neighbors, say, node u, at or before the end of time slot $\tau - 1$, and then forward the packet along the link l from u to v at time slot τ. In this case, the cost of delivering the packet to node v by time τ is $\Theta(u, \tau-1)+\beta_{l\tau}$. Therefore, we have

$$\Theta(v,\tau) = \min \begin{cases} \Theta(v,\tau-1), \\ \Theta(u,\tau-1) + \beta_{l\tau}, \quad \forall l := (u,v) \in \mathcal{L}. \end{cases}$$

Based on the above recursive equation, we design Algorithm 2 to compute the optimal cost $\Theta(\delta_m, f_m)$ and the optimal schedule k^* for each packet m. In the algorithm, we use $S(v,\tau)$ to denote the schedule that achieves $\Theta(v,\tau)$.

In Algorithm 2, the inequality $\Theta(u,\tau-1)+\beta_{l\tau} < \Theta(v,\tau)$ is only evaluated once for any link and time slot. Let $E := |\mathcal{L}|$ be the number of links in the network. Let $T := \max_m(f_m - a_m + 1)$ be the maximum relative deadline for all packets. Then the time complexity of both Algorithm 2 and Algorithm 1 is $O(ET)$.

C. Competitive Ratio Analysis

Before analyzing the performance of Algorithm 1, we first establish a basic property of β_{lt}.

Lemma 2: Let $\beta_{lt}[n]$ be the value of β_{lt} after n packets are scheduled to use link l at time t. Then,

$$\beta_{lt}[n] = \frac{1}{d_l - 1}\left(d_l^{\frac{n}{RC_l}} - 1\right). \tag{4}$$

Proof: First, note that the value of β_{lt} is only changed when Algorithm 1 uses link l at time t to transmit a packet. Therefore, the value of β_{lt} only depends on the number of packets that are scheduled to use link l at time t.

We then prove (4) by induction. Initially, when $n = 0$, $\beta_{lt}[0] = 0 = (\frac{1}{d_l-1})(d_l^0 - 1)$ and (4) holds.

Suppose (4) holds for the first n packets. When the $(n+1)$-th packet is scheduled for link l at time t, we have

$$\begin{aligned}\beta_{lt}[n+1] =& \beta_{lt}[n]\left(1+\frac{1}{C_l}\right) + \frac{1}{(d_l-1)C_l} \\ =& \frac{1}{d_l-1}\left(d_l^{\frac{n}{RC_l}} - 1\right)\left(1+\frac{1}{C_l}\right) + \frac{1}{(d_l-1)C_l} \\ =& \frac{1}{d_l-1}\left[d_l^{\frac{n}{RC_l}}\left(1+\frac{1}{C_l}\right) - 1\right]\end{aligned}$$

We select $d_l = (1+\frac{1}{C_l})^{RC_l}$, and therefore

$$\beta_{lt}[n+1] = \frac{1}{d_l-1}\left(d_l^{\frac{n+1}{RC_l}} - 1\right),$$

and (4) still holds for $n+1$. Thus, by induction, (4) holds for all n. ∎

Remark: Since β_{lt} is an exponential function of the load $\frac{n}{RC_l}$ of link l at time t, we can call β_{lt} the *exponential load* of link l at time t. It is monotonic, and the value is 0 (resp. 1) when the load is 0 (resp. 1).

We now establish the competitive ratio of Algorithm 1.

Theorem 1: Let $C_{\min} := \min C_l$, $d_{\min} := (1+1/C_{\min})^{RC_{\min}}$, and L be the longest path between a source node and a destination node, that is, all valid schedules have $|k| \leq L$, for all $m \in \mathcal{M}, k \in V(m)$. Algorithm 1 produces solutions that satisfy all constraints in **Schedule**(R) and **Dual**. Moreover, Algorithm 1 is $(R, 1+\frac{L}{d_{\min}-1})$-competitive, which converges to $(R, 1+\frac{L}{e^R-1})$-competitive, as $C_{\min} \to \infty$.

Proof: First, we show that the dual solutions $\{\alpha_m\}$ and $\{\beta_{lt}\}$ satisfy constraints (3b) to (3d). Initially, we have $\beta_{lt} = 0$. By Lemma 2, $\beta_{lt} \geq 0$ holds. Since step 6 is only used when $\sum_{(l,t)\in k^*} \beta_{lt} < 1$, $\alpha_m \geq 0$ holds. From step 4 and 6, we know that $\alpha_m + \sum_{(l,t)\in k} \beta_{lt} \geq (1 - \sum_{(l,t)\in k} \beta_{lt}) + \sum_{(l,t)\in k} \beta_{lt} = 1$. Thus (3b) to (3d) hold.

Next, we show $\{X_{mk}\}$ satisfies constraints (2b) to (2d). By step 4, the algorithm picks at most one schedule k^* for packet m, constraint (2b) holds. With Lemma 2, $\beta_{lt} = 1$ when RC_l packets use link l at time t. Since a valid schedule including (l,t) will be chosen for packet m only when $\sum_{(l,t)\in k^*} \beta_{lt} < 1$, all (l,t) in the chosen valid schedule must have $\beta_{lt} < 1$, and therefore the number of packets transmitted over link l at time t must be less than RC_l. Thus, at any time t, there are at most RC_l packets using link l. Constraint (2c) holds. By initialization and step 8, constraint (2d) holds.

We derive the ratio between $\sum_m \alpha_m + \sum_{(l,t)} C_l\beta_{lt}$ and $\sum_{mk} X_{mk}$. Initially, both are equal to 0. We consider the increasing amount for both when a new packet m arrives at the network. We use $\Delta P(R)$ to denote the change of $\sum_{mk} X_{mk}$, and ΔD to denote the change of $\sum_m \alpha_m + \sum_{(l,t)} C_l\beta_{lt}$.

If packet m is dropped, both $\Delta P(R)$ and ΔD are 0. If packet m is accepted and transmitted using valid schedule k^*, we have $X_{mk^*} = 1$. Thus, $\Delta P(R) = 1$. On the other hand, ΔD is increased as:

$$\begin{aligned}\Delta D =&\Delta\alpha_m + \sum_{(l,t)\in k^*} C_l\Delta\beta_{lt}\\ =&(1 - \sum_{(l,t)\in k^*} \beta_{lt}) + \sum_{(l,t)\in k^*} (\beta_{lt} + \frac{1}{(d_l-1)C_l})\\ =&1 + \sum_{(l,t)\in k^*} \frac{1}{(d_l-1)} \le 1 + \frac{L}{d_{\min}-1}\end{aligned}$$

Therefore, for each packet arrival, the ratio between ΔD and $\Delta P(R)$is no larger than $1+\frac{L}{d_{\min}-1}$ if $\Delta D > 0$. When the algorithm terminates, we have $\frac{\sum_m \alpha_m+\sum_{(l,t)} C_l\beta_{lt}}{\sum_{mk} X_{mk}} \le 1 + \frac{L}{d_{\min}-1}$. By Lemma 1, $\frac{\Gamma_{\text{opt}}}{\sum_{mk} X_{mk}} \le 1 + \frac{L}{d_{\min}-1}$, and the competitive ratio of Algorithm 1 is $(R, 1 + \frac{L}{d_{\min}-1})$. When $C_{\min} \to \infty$, $d_{\min} = (1 + \frac{1}{C_{\min}})^{RC_{\min}} \to \mathrm{e}^R$, and the competitive ratio of Algorithm 1 converges to $(R, 1 + \frac{L}{\mathrm{e}^R-1})$. ∎

There are several important implications of Theorem 1. First, without increasing capacity, that is, when $R = 1$, the competitive ratio of our policy is $(1, O(L))$. In comparison, the MKS policy proposed by Mao *et al.* [2] focuses on the special case of $R = 1$ and has a competitive ratio of $(1, O(L \log L))$. Therefore, our algorithm is asymptotically better than the MKS online algorithm. Second, this theorem allows us to quantify the amount of capacity needed to a certain performance guarantee. For the PD algorithm to guarantee to deliver at least $1-\frac{1}{\theta}$ as many packets as the optimal solution, Theorem 1 states that we only need to increase all link capacities by R_θ times such that $1+\frac{L}{\mathrm{e}^{R_\theta}-1} \le 1/(1-\frac{1}{\theta}) = 1+\frac{1}{\theta-1}$. Therefore, we have $R_\theta = \ln(L(\theta-1)+1) \le \ln L + \ln\theta$. For example, if we are required to use PD to deliver 99% of the packets and the longest path consists of 10 hops, then we need to increase link capacities by 6.9 times.

V. A Theoretical Lower Bound for Competitive Ratio

In Section IV, we showed that our PD policy is $(R, 1 + \frac{L}{\mathrm{e}^R-1})$-competitive. In this section, we will establish a lower bound for the competitive ratio of online policies.

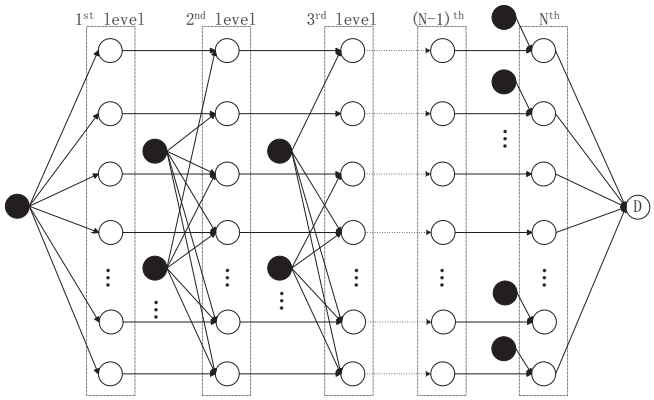


Fig. 2. Network topology for lower bound analysis.

Theorem 2: Any online algorithm cannot be better than $(R, 1 + \frac{L-2\mathrm{e}^R}{(L+1)\mathrm{e}^R-L})$-competitive.

Proof: We design a network as shown in Fig 2. We start to construct the network from an up-link tree, which is shown as the white nodes in Fig 2. Root is marked as node D and it is the destination of all packets. There are N levels of non-root nodes with N nodes in each level. Each node is connected to one node in the next level. Nodes do not share parent except the N-th level nodes share the same root node. At the j-th level, where $1 \le j \le N$, there are $\binom{N}{N+1-j}$ extra nodes, which is shown as the black nodes in Fig 2, with each node connecting to an unique set of $N+1-j$ nodes in this level. For example, there is one black node connected to all white nodes in level 1, and there are N black nodes connected to white nodes in level 2, where each of these black nodes is connected all but one white nodes in level 2. Likewise, there are $\binom{N}{N-2}$ black nodes connected to white nodes in level 3, with each black node connected to $N-2$ white nodes in level 3, and no two black nodes are connected to the same subset of white nodes.

Next, we describe packet arrivals. Packets only arrive at black nodes. Of all black nodes connected to the same level of white nodes, only one black node has packet arrival. Let $\mathcal{W}_j$ be the set of white nodes in j-th level which connects to the black node with packet arrivals. The black nodes with packet arrivals are chosen such that all nodes in $\mathcal{W}_{j+1}$ are connected to those in $\mathcal{W}_j$. Fig 3 is a simplified network of Fig 2, where we omit the black nodes with no packet arrival and marked each black node with a number from 1 to N.

Packets arrive at nodes $1, 2, ..., N$. Their destination is node D. Each link in the network has capacity C. At the beginning of time slot 1, there are C packets arriving at node 1. Node 1 is connected to N links: l_{11}, l_{12}, $\cdots$, l_{1N}. At the beginning of time slot 2, there are C packets arriving at node 2. Node 2 is connected to $N-1$ links: l_{21}, l_{22}, $\cdots$, $l_{2(N-1)}$. Similarly for nodes 3, 4, $\cdots$. At the beginning of time N, there are C packets arriving at node N. The deadline of all packets is $N+1$. Node N is connected only to link l_{N1}.

When one knows which black nodes have packet arrivals, the offline optimal algorithm is to transmit the first C packets

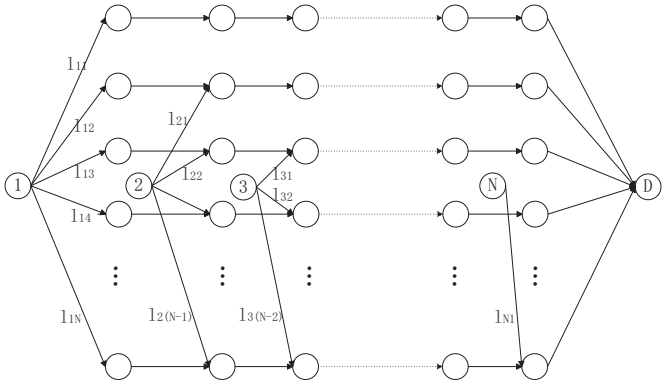


Fig. 3. Simplified network topology for lower bound analysis.

through link l_{11} and the following links, the second C packets through link l_{21} and the following links, ..., and the N-th C packets through link l_{N1} and the following link. The total number of delivered packets is NC.

Next we consider the online algorithm when all links' capacity is increased by R times. Since online policies do not know which black nodes will have packet arrivals, the optimal online policy is to distribute packets evenly among all connected links. That is, at time 1, each of links l_{1i}, $i = 1, 2, \cdots, N$, transmit C/N packets. At time 2, each of link l_{2i}, $i = 1, 2, \cdots, (N-1)$, transmits $C/(N-1)$ packets. At time K, link l_{Ki}, $i = 1, 2, \cdots, (N-K+1)$, transmits $C/(N-K+1)$ packets. For simplicity, we call the routes from node 1 to node D through l_{1i} route r_i. If all packets arrive at node K are accepted, routes r_i, $i = K, K+1, \cdots, N$ have the same load on each link. When any link on a single route reaches its capacity, the route cannot be used for future arrival packets. Suppose the route gets over-loaded at time $K+1$, that is, packets arrive at node K are accepted and packets arrive at node $K+1$ are not fully accepted. The maximum load of a single link on route r_N is at most $\frac{C}{N} + \frac{C}{N-1} + \cdots + \frac{C}{N-K+1}$ and at least $\frac{C}{N} + \frac{C}{N-1} + \cdots + \frac{C}{N-K}$. We then have:

$$C(\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \ldots + \frac{1}{N-K+1}) \leq RC,$$

and

$$C(\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \ldots + \frac{1}{N-K}) \geq RC.$$

Since

$$\int_{N-K+1}^{N+1} \frac{1}{x}\mathrm{d}x < (\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \ldots + \frac{1}{N-K+1}),$$

and

$$\int_{N-K-1}^{N} \frac{1}{x}\mathrm{d}x > (\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \ldots + \frac{1}{N-K}).$$

We have:

$$\log(N+1) - \log(N-K+1) = \log\frac{N+1}{N-K+1} < R,$$

and

$$\log(N) - \log(N-K-1) = \log\frac{N}{N-K-1} > R.$$

Then we can derive the value of K as: $N - \frac{N}{\mathrm{e}^R} - 1 \leq K \leq N + 1 - \frac{N+1}{\mathrm{e}^R}$. The total number of accepted packets is in the range $((N - \frac{N}{\mathrm{e}^R} - 1)C, (N + 2 - \frac{N+1}{\mathrm{e}^R})C)$.

Thus the competitive ratio of an online policy is at best $(R, \frac{N}{N+2-\frac{N+1}{\mathrm{e}^R}})$. In Fig. 2, the longest path in the network is between the leftmost black node and the sink, which has length $L = N + 1$. The competitive ratio can then be rewritten as $(R, 1 + \frac{L-2\mathrm{e}^R}{(L+1)\mathrm{e}^R - L})$. ■

Let us once again consider the scenario where online policies need to guarantee to deliver at least $1 - \frac{1}{\theta}$ as many packets as the optimal solution. Theorem 2 states that any online policy needs to increase its link capacities by at least R_θ times so that $1 + \frac{L-2e^{R_\theta}}{(L+1)e^{R_\theta} - L} \leq 1 + \frac{1}{\theta - 1}$. Solving this equation, we have R_θ needs to be at least $\ln L + \ln\theta - \ln(L + 2\theta - 1)$. In comparison, our policy only needs to increase link capacities by $(\ln L + \ln\theta)$ times to ensure the delivery of $1 - \frac{1}{\theta}$ as many packets as the optimal solution. Therefore, the capacity requirement of our policy is at most $\ln(L + 2\theta - 1)$ away from the lower bound. Suppose we fix the ratio between L and θ, and let them both go to infinity, then we have $(\ln L + \ln\theta)/(\ln L + \ln\theta - \ln(L + 2\theta - 1)) \to 2$. Therefore, when both L and θ are large, our policy at most requires twice as much capacity as the theoretical lower bound.

VI. An Order-Optimal Online Policy When $R = 1$

We have shown that our PD algorithm is $(R, 1 + \frac{L}{d_{\min} - 1})$-competitive. Without increasing link capacity, i.e, when $R = 1$, the algorithm is $(1, 1 + \frac{L}{\mathrm{e}-1})$-competitive, as $C_{\min} \to \infty$. While the competitive ratio of our PD algorithm is an order better than the MKS algorithm [2], it still fails to achieve the theoretical bound of $(1, O(\log L))$-competitive. In this section, we propose another online algorithm and prove that it achieves the theoretical bound when $R = 1$.

A. Algorithm Description

Similar to the design of Algorithm 1, we aim to design an algorithm that constructs $\{X_{mk}\}, \{\alpha_m\}, \{\beta_{lt}\}$ while ensuring they satisfy all constraints in **Schedule** and **Dual**. The algorithm is described in Algorithm 3. One can see that Algorithm 3 is very similar to Algorithm 1, and their only difference lies in the update rules for β_{lt}. Recall that $\beta_{lt}[n]$ is the value of β_{lt} when link l serves a total number of n packets at time t. Define $\beta(x)$ as

$$\beta(x) := \begin{cases} \frac{\mathrm{e}^x - 1}{L(\mathrm{e}^{\frac{1}{\ln L + 1}} - 1)}, & \text{if } x \leq \frac{1}{\ln L + 1}; \\ \mathrm{e}^{(x-1)(\ln L + 1)}, & \text{if } x \geq \frac{1}{\ln L + 1}. \end{cases} \tag{5}$$

Then Algorithm 3 chooses the value of $\beta_{lt}[n]$ as $\beta_{lt}[n] = \beta(\frac{n}{RC_l})$.

To illustrate the difference in β_{lt}, we plot the values of $\beta_{lt}[n]$ for a link with $C_l = 1000$ under the two policies in Fig. 4, where we consider the two cases $L = 8$ and $L = 64$ for Algorithm 3. As can be shown in the figure, when n is small,

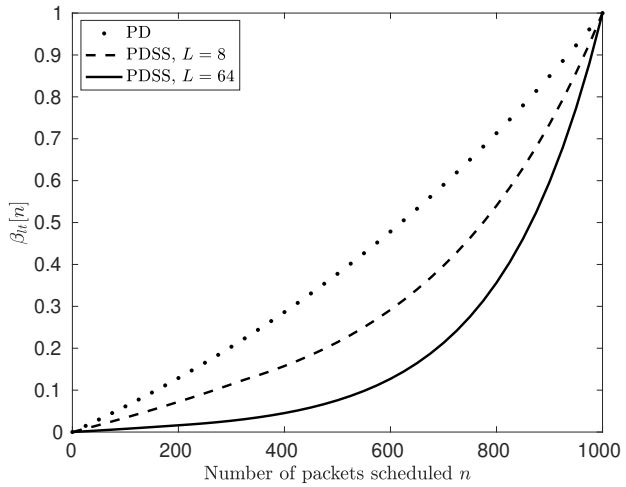


Fig. 4. Values of $\beta_{lt}[n]$ under different policies.

Algorithm 3 increases the value of β_{lt} much more slowly than Algorithm 1 does. Moreover, Algorithm 3 increases β_{lt} slower when L is larger. Based on this observation, we call Algorithm 3 "PD With Slow Start (PDSS)". Recall that both Algorithm 1 and Algorithm 3 only schedule a packet when $\min_k \sum_{(l,t)\in k} \beta_{lt} < 1$. By increasing β_{lt} slower when n is small, Algorithm 3 ensures that more packets with long routes can be accepted, especially when the network is lightly loaded.

Algorithm 3 PDSS: Primal Dual With Slow Start

Initially, $\alpha_m \leftarrow 0$, $\beta_{lt} \leftarrow 0$, $X_{mk} \leftarrow 0$.
for each arriving packet m **do**
  $k^* \leftarrow \text{argmin}_k \sum_{(l,t)\in k} \beta_{lt}$
  if $\sum_{(l,t)\in k^*} \beta_{lt} < 1$ **then**
    $\alpha_m \leftarrow (1 - \sum_{(l,t)\in k^*} \beta_{lt})$
    for each $(l,t) \in k^*$ **do**
      if total number of packets n at time t on link l: $n \le \frac{RC_l}{\ln L+1}$ **then**
        $\beta_{lt} \leftarrow \frac{1}{L(\mathrm{e}^{\frac{1}{\ln L+1}}-1)}(\mathrm{e}^{\frac{n}{RC_l}} - 1)$,
      else
        $\beta_{lt} \leftarrow \mathrm{e}^{(\frac{n}{RC_l}-1)(\ln L+1)}$
      end if
    end for
    $X_{mk^*} \leftarrow 1$.
    Transmit packet m using valid schedule k^*.
  else
    Drop packet m.
  end if
end for

B. Competitive Ratio Analysis

We now prove that Algorithm 3 achieves the theoretical bound in [2] by being $(1, O(\log L))$-competitive.

Lemma 3: Let $C_{\min} := \min C_l$. In Algorithm 3, each time a new packet is scheduled, the ratio between the change of **Schedule** and **Dual** is bounded by $2(\ln L + 1) + \frac{B}{C_{\min}}$, where the value of B is independent of $C_{\min}$.

Proof: If a new packet is admitted to the network, the increasing amount of **Dual** is

$$\begin{aligned}\Delta D =& \Delta\alpha_m + \sum_{(l,t)\in k^*} C_l \Delta\beta_{lt} \\ =& 1 + \sum_{(l,t)\in k^*} (C_l\Delta\beta_{lt} - \beta_{lt})\end{aligned}$$

Using Taylor Sequence, we then have

$$\begin{aligned}\Delta\beta_{lt}[n] :=& \beta_{lt}[n+1] - \beta_{lt}[n] = \beta(\frac{n+1}{C_l}) - \beta(\frac{n}{C_l}) \\ \le& \frac{1}{C_l}\beta'(\frac{n}{C_l}) + \epsilon\frac{1}{C_l^2}\beta''(\frac{n}{C_l}),\end{aligned}$$

for some bounded constant $\epsilon < \infty$, where β' and β'' are the first and second derivative of β, respectively. We note that the function $\beta(x)$ is continuous for all x, and infinitely differentiable for all x except at the point $x_0 := \frac{1}{\ln L+1}$. At the point x_0, we define $\beta'(x_0) = \lim_{x\to x_0^+} \beta'(x)$ and $\epsilon\beta''(x_0) = \lim_{x\to x_0^+} \epsilon\beta''(x)$. This ensures that the above inequality still holds.

By (5) we know that $n \le \frac{C_l}{\ln L+1}$ if and only if $\beta_{lt}[n] \le \frac{1}{L}$. If $x = \frac{n}{C_l} \le \frac{1}{\ln L+1}$, then $\beta'(x) = \beta''(x) = \frac{\mathrm{e}^x}{L(\mathrm{e}^{\frac{1}{\ln L+1}}-1)}$. We have:

$$\begin{aligned}C_l\Delta\beta_{lt}[n] - \beta_{lt}[n] \le& \frac{C_l(\frac{1}{C_l}\mathrm{e}^{\frac{n}{C_l}}) + \epsilon(\frac{1}{C_l})^2\mathrm{e}^{\frac{n}{C_l}}) - (\mathrm{e}^{\frac{n}{C_l}} - 1)}{L(\mathrm{e}^{\frac{1}{\ln L+1}} - 1)} \\ \le& \frac{1 + \epsilon\frac{1}{C_l}\mathrm{e}^{\frac{n}{C_l}}}{L(1 + \frac{1}{\ln L+1} - 1)} \\ \le& \frac{\ln L + 1}{L}(1 + \epsilon\frac{1}{C_l}\mathrm{e})\end{aligned}$$

Let $B_1 = \epsilon\mathrm{e}\frac{\ln L+1}{L}$, then

$$C_l\Delta\beta_{lt}[n] - \beta_{lt}[n] \le \frac{\ln L + 1}{L} + B_1\frac{1}{C_{\min}}, \tag{6}$$

when $\frac{n}{C_l} \le \frac{1}{\ln L+1}$.

On the other hand, If $x = \frac{n}{C_l} \ge \frac{1}{\ln L+1}$, then $\beta'(x) = (\ln L + 1)\beta(x)$ and $\beta''(x) = (\ln L + 1)^2\beta(x)$. We have:

$$\begin{aligned}&C_l\Delta\beta_{lt}[n] - \beta_{lt}[n] \\ &\quad\le C_l[\frac{\ln L + 1}{C_l}\beta_{lt}[n] + \epsilon(\frac{\ln L + 1}{C_l})^2\beta_{lt}[n]] - \beta_{lt}[n] \\ &\quad\le \ln L \cdot \beta_{lt}[n] + \frac{1}{C_l}\epsilon(\ln L + 1)^2\beta_{lt}[n]\end{aligned}$$

Let $B_2 = \epsilon(\ln L + 1)^2$, then

$$C_l\Delta\beta_{lt}[n] - \beta_{lt}[n] \le (\ln L + B_2\frac{1}{C_{\min}})\beta_{lt}[n], \tag{7}$$

when $\frac{n}{C_l} \ge \frac{1}{\ln L+1}$.

If packet m is transmitted using valid schedule k^*, we have $X_{mk^*} = 1$. Thus, $\Delta P = 1$. On the other hand, ΔD is increased as:

$$\begin{aligned}\Delta D =& 1 + \sum_{(l,t):(l,t)\in k^*} C_l \Delta\beta_{lt} - \beta_{lt} \\ \leq& 1 + \sum_{(l,t):(l,t)\in k^*, \beta_{lt}\leq \frac{1}{L}} C_l \Delta\beta_{lt} - \beta_{lt} \\ &+ \sum_{(l,t):(l,t)\in k^*, \beta_{lt}\geq \frac{1}{L}} C_l \Delta\beta_{lt} - \beta_{lt}\end{aligned}$$

From (6) and (7) we have:

$$\begin{aligned}\Delta D \leq& 1 + \sum_{(l,t):(l,t)\in k^*, \beta_{lt}\leq \frac{1}{L}} (\frac{\ln L + 1}{L} + B_1 \frac{1}{C_{\min}}) \\ &+ \sum_{(l,t):(l,t)\in k^*, \beta_{lt}\geq \frac{1}{L}} ((\ln L + B_2 \frac{1}{C_{\min}})\beta_{lt})\end{aligned}$$

From Algorithm 3 step 4 we know that $\sum \beta_{lt} \leq 1$, thus we have

$$\begin{aligned}\Delta D \leq& 1 + (\ln L + 1 + B_1 \frac{L}{C_{\min}}) + (\ln L + B_2 \frac{1}{C_{\min}}) \\ =& 2 + 2\ln L + \frac{B_1 L + B_2}{C_{\min}},\end{aligned}$$

and the proof is complete. ■

Theorem 3: Algorithm 3 produces solutions that satisfy all constraints in **Schedule** and **Dual**. Moreover, it is $(1, 2(1 + \ln L))$-competitive, as $C_{\min} \to \infty$.

Proof: We use the same approach as in the proof of Theorem 1 to establish that all constraints are satisfied in **Schedule** and **Dual**. First, we show that the dual solutions $\{\alpha_m\}$ and $\{\beta_{lt}\}$ satisfy constraints (3b) to (3d). Initially, we have $\beta_{lt} = 0$. By (5), $\beta_{lt} \geq 0$ holds. Since step 5 is only used when $\sum_{(l,t)\in k^*} \beta_{lt} < 1$, $\alpha_m \geq 0$ holds. From step 3 and 5, we know that $\alpha_m + \sum_{(l,t)\in k} \beta_{lt} \geq (1 - \sum_{(l,t)\in k} \beta_{lt}) + \sum_{(l,t)\in k} \beta_{lt} = 1$. Thus (3b) to (3d) hold.

Next, we show $\{X_{mk}\}$ satisfies constraints (1b) to (1d). By step 3, the algorithm picks at most one schedule k^* for packet m, constraint (1b) holds. With (5), when the number of packets on link l at t is C_l, we have $\beta_{lt} = 1$. Also, since a packet is scheduled if $\sum_{(l,t)\in k^*} \beta_{lt} < 1$, we have $\beta_{lt} < 1$ for all $(l,t) \in k^*$. Therefore, the number of packets transmitted on link l at any time t is at most C_l. Constraint (1c) holds. By initialization and step 13, constraint (1d) holds.

When a new packet m arrives, it will either be dropped or scheduled. If it is dropped, both ΔP and ΔD are 0. If it is scheduled, both (3a) and (1a) increase. With Lemma 3, the ratio between ΔP and ΔD is bounded by $2(1 + \ln L) + \frac{B}{C_{\min}}$. Therefore the competitive ratio of Algorithm 3 is $(1, 2(1 + \ln L) + \frac{B}{C_{\min}}) \to (1, 2(1 + \ln L))$, as $C_{\min} \to \infty$. ■

Thus, comparing with the result in [2], Algorithm 3 achieves the optimal competitive ratio when $R = 1$.

Remark: Applying the above proof to a general $R > 1$ leads to trivial results. However, PDSS performs very well when $R > 1$ as suggested by Section VIII.

VII. A Distributed Protocol for Implementation

The two algorithms that we have proposed so far are both centralized algorithms. Specifically, when a packet arrives at a node, the node needs to have complete knowledge of all exponential loads β_{lt} of all links to find a valid schedule. These algorithms are applicable in software defined networks (SDN) where there is a centralized controller. However, in other systems, it is preferable to have distributed algorithms which do not require always up-to-date global information. In this section, we propose a distributed protocol called PDD (Primal Dual Distributed) based on the design of PD in Algorithm 1. Note that we cannot directly employ the distributed method by Kuhn *et al.* [32] since we aim to design online algorithms without knowledge of future information.

In our distributed protocol, the task of transmitting a packet to its destination is decomposed into two parts: First, when a packet arrives at its source node, the node determines a suggested schedule based on past system history. This suggested schedule consists of the route for forwarding the packet, as well as a local deadline for each link. After determining the suggested schedule, the node simply forwards it to the first link of the route. On the other hand, when a link receives a packet along with a suggested schedule, the link tries to forward the packet to the next link in the suggested schedule by its local deadline. The link drops the packet when it cannot forward the packet in time.

To facilitate this protocol, each link keeps track of its own exponential load β_{lt}, which reflects the number of packets that have been scheduled to be transmitted over link l at time t. The value of β_{lt} changes over time, and is updated when more packets are scheduled to transmit over link l at time t. Link l, more specifically the start node of link l, broadcasts the exponential load of its own and others which it has received periodically so that all nodes can learn the values of β_{lt} for all links. Broadcasts occur infrequently to minimize its overhead on network bandwidth. We also do not need to broadcast β_{lt} for time t that is before the current broadcast time.

We now describe how a node s_m determines a suggested schedule upon the exogenous arrival of a packet m at time $t_{\text{a}} := a_m$. Let t_{b} be the last broadcast time. We use $\tilde{\beta}_{lt}$ to denote the latest values of β_{lt} that node s_m has received for all $l \in \mathcal{L}, t \geq t_{\text{b}}$. Recall that in the PD Algorithm, the node would like to find a valid schedule that minimizes $\sum_{(l,t)\in k} \beta_{lt}$. In distributed networks, the node only knows the exact values of β_{lt} for links that are incident to the node. However, it receives $\tilde{\beta}_{lt}$ for all other links. In our protocol, the node treats the time in broadcasted exponential load relatively. It assumes that the values of β_{lt} starting from $t = t_{\text{a}}$ for some other link l are the same as $\tilde{\beta}_{lt}$ starting from $t = t_{\text{b}}$ respectively, that is $\beta_{lt} = \tilde{\beta}_{l,t-t_{\text{a}}+t_{\text{b}}}, \forall t \geq t_{\text{a}}$. It then finds a valid schedule k^* that minimizes $\sum_{(l,t)\in k} \beta_{lt}$. Similar to the original PD algorithm, the node drops the packet if $\sum_{(l,t)\in k^*} \beta_{lt} \geq 1$. If $\sum_{(l,t)\in k^*} \beta_{lt} < 1$, then the node puts information of k^* into the header of the packet, and forwards the packet to the first link in k^*. Algorithm 4 summarizes the steps of schedule suggestion for each packet at its source node. To solve the optimization problem in line 7, we use dynamic programming

similar to Algorithm 2. The difference is that when multiple schedules achieves the optimum, we choose one of them to be k^* uniformly at random.

Algorithm 4 PDD: Schedule Suggestion at Source Nodes

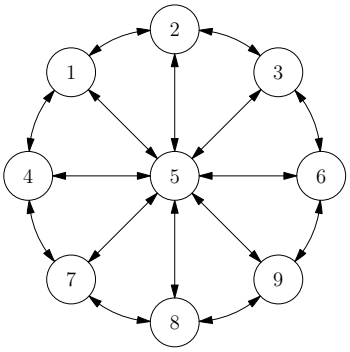
for each arriving packet m **do**
  $t_{\mathrm{a}} \leftarrow a_m$ // packet arrival time
  $t_{\mathrm{b}} \leftarrow$ last broadcast time
  $\tilde{\beta}_{lt} \leftarrow$ last broadcasted exponential load
  $\beta_{lt} \leftarrow$ most up-to-update values, $\forall t \geq t_{\mathrm{a}}, l : s_m \in l$
  $\beta_{lt} \leftarrow \tilde{\beta}_{l,t-t_{\mathrm{a}}+t_{\mathrm{b}}}, \forall t \geq t_{\mathrm{a}}, l : s_m \notin l$
  $k^* \leftarrow \mathrm{argmin}_k \sum_{(l,t)\in k} \beta_{lt}$ // suggested schedule
  if $\sum_{(l,t)\in k^*} \beta_{lt} < 1$ **then**
    Put k^* in the header of packet m.
    Forward the packet to the first link in k^*.
  else
    Drop packet m.
  end if
end for

Since the actual values of β_{lt} can be different from those last broadcasted, there is no guarantee that a packet can be delivered in time using the valid schedule k^* even if $\sum_{(l,t)\in k^*} \beta_{lt} < 1$. Therefore, when a node determines a valid schedule k^* for a packet, the valid schedule k^* is treated only as a suggestion for links in k^*. Specifically, if k^* contains an entry (l^*, t^*), then the link l^* interprets k^* as a requirement that l^* needs to forward the packet to the next link by t^*, or drops the packet. When l^* obtains the packet, it still has the freedom to choose when to forward the packet, as long as the packet is forwarded by the time t^*.

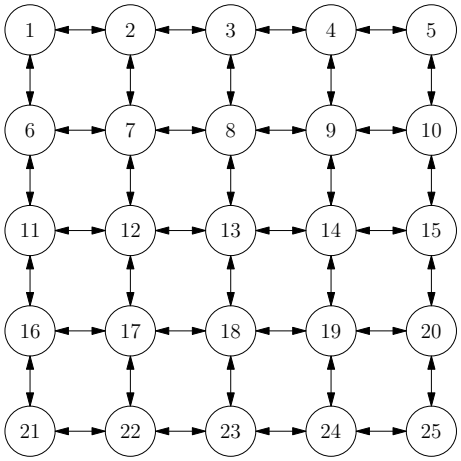
Next, we discuss how each link determines the actual time to transmit each packet. Obviously, each link l^* knows its own β_{l^*t}. From the design of PD, we can see that PD prefers to transmit packets when β_{lt} is small. Our proposed policy is based on this principle. When a link l^* receives a packet, it finds the entry (l^*, t^*) from the valid schedule k^* specified in the header of the packet. Link l^* then finds a time t_{s} between the current time and t^* that has the smallest β_{l^*t}, and transmits the packet at time t_{s}. Algorithm 5 summarizes the policy for packet transmission on each link.

Algorithm 5 PDD: Packet Transmission on Each Link

for each incoming packet m on link l^* **do**
  $k^* \leftarrow$ suggested schedule in the packet header
  $t^* \leftarrow t$ such that $(l^*, t) \in k^*$ // local deadline
  $t_0 \leftarrow$ current time
  $t_{\mathrm{s}} \leftarrow \mathrm{argmin}_{t_0 \leq t \leq t^*} \beta_{l^*,t}$
  if $\beta_{l^*,t_{\mathrm{s}}} < 1$ **then**
    $\beta_{l^*t_{\mathrm{s}}} \leftarrow \beta_{l^*,t_{\mathrm{s}}} \left(1 + \frac{1}{C_{l^*}}\right) + \frac{1}{(d_{l^*}-1)C_{l^*}}$
    Transmit packet m on link l^* at time t_{s}.
  else
    Drop packet m.
  end if
end for



(a) A small network



(b) A 5×5 grid network

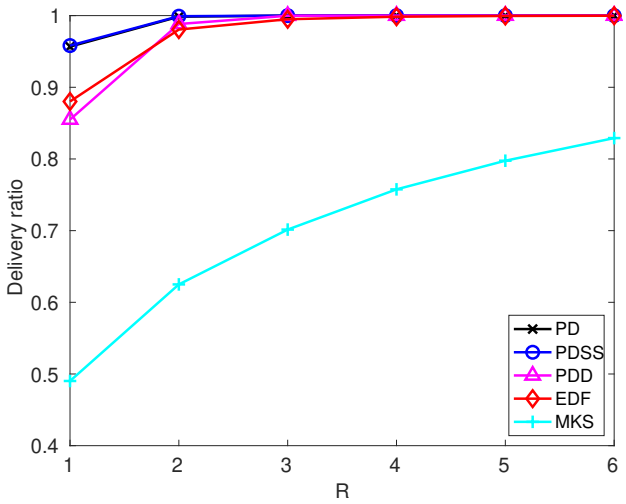
Fig. 5. Network topologies in simulations.

VIII. Simulations

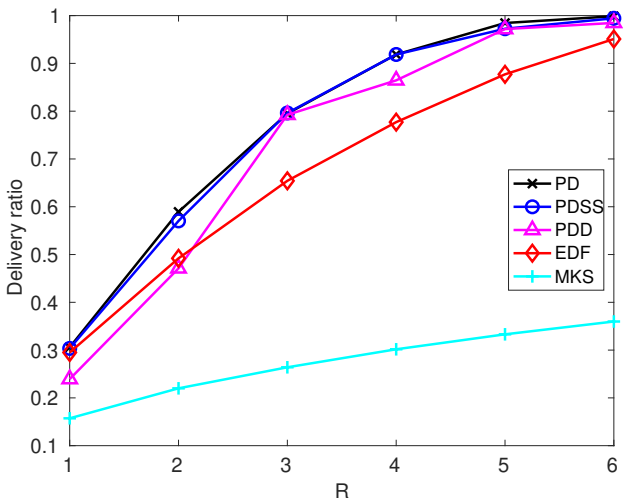
In this section, we evaluate the performance of our policies by simulations. We compare our policies with the EDF policy and the MKS policy [2]. Both EDF and MKS focus on packet scheduling, and are applicable only when the route of the packet is given. For these two policies, we assume that each packet is routed through the shortest path.

We consider the combinations of two network topologies, two link capacity settings, and two traffic patterns in our simulations. This gives us eight combinations in total. We first introduce the two network topologies. The first one is a small network as shown in Fig. 5a. The network has 9 nodes from node 1 to node 9. Neighboring nodes have bidirectional links between them, and each pair of nodes can communicate within two hops. This topology can be useful in smart home environment where all wireless devices can talk to a central hub (Node 5), and nearby wireless devices can also communicate directly. The second one is a 5×5 grid network which is also used by [2]. Fig. 5b depicts the grid network. There are 80 directional links in this network. The longest path between all pairs of nodes has a length of 24.

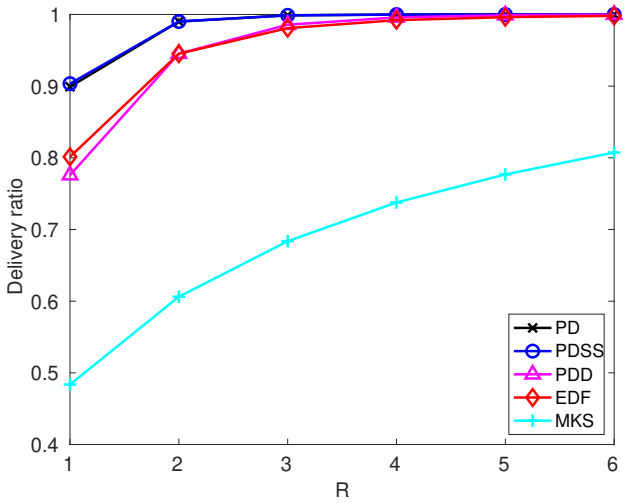
Given a network topology, we consider two link capacity settings. One is "homogeneous link capacity", where each link in the network has a capacity of two units when $R = 1$. The other is "heterogeneous link capacities", where we choose the link capacities to be integers uniformly at random from 1 to



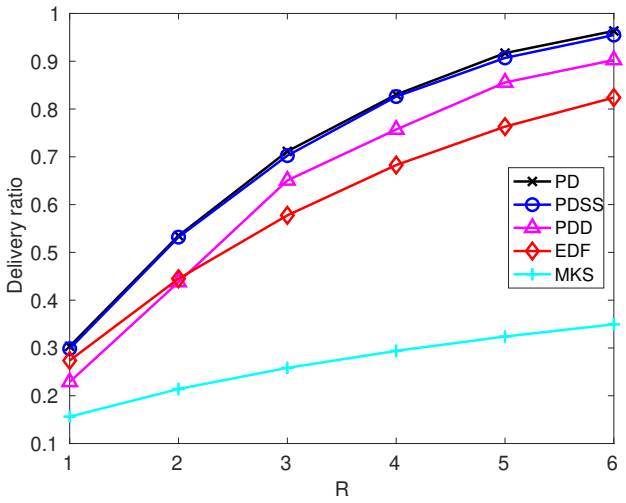
(a) Homogeneous link capacity, light traffic



(b) Homogeneous link capacity, heavy traffic



(c) Heterogeneous link capacities, light traffic



(d) Heterogeneous link capacities, heavy traffic

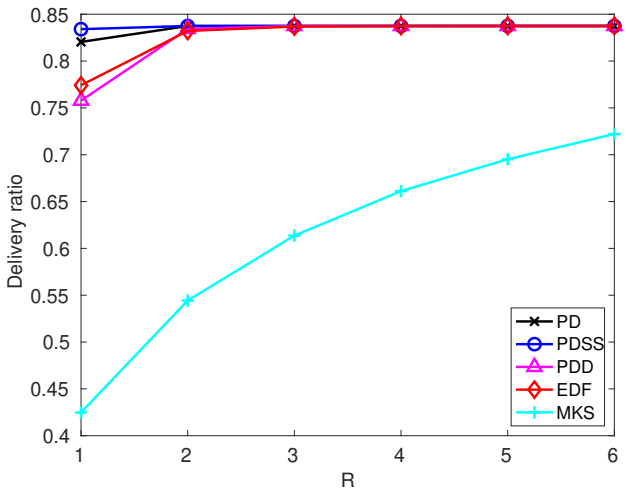
Fig. 6. Delivery ratio comparison for the small network.

3 when $R = 1$.

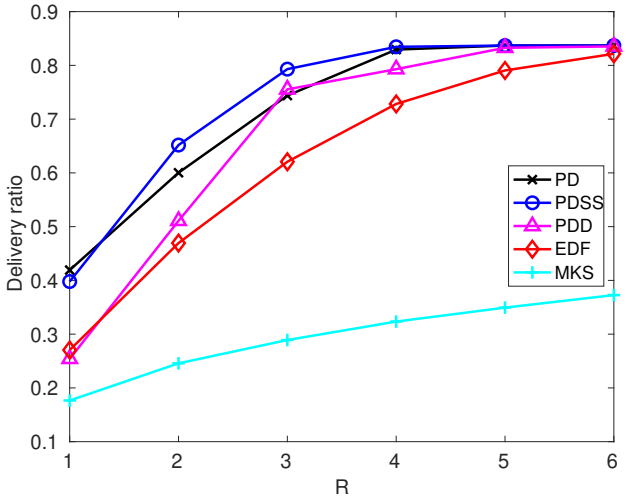
Given a network topology and link capacity setting, we consider two traffic patterns. The first one corresponds to light traffic. The inter-arrival time between packets are chosen to be 0 with probability 0.95 and 1 with probability 0.05. On average, there are about 19.8 packets arriving in each slot. The second pattern has heavy traffic. At the beginning of each time slot, the number of packets arriving the system is chosen uniformly at random from 100 to 200. For both traffic patterns, there are 10^4 packets arriving at the system. The source nodes and destination nodes are both chosen from all nodes in the network with equal probability, and for each packet the destination node is not allowed to be the same as the source node. We choose the relative deadline of each packet, the time between its arrival and deadline, to be an integer uniformly at random from 2 to 6 for the small network and from 2 to 10 for the 5×5 grid network. Simulations end after the expiry of all packets. The average simulation duration varies from 73 slots (small network, heavy traffic) to 506 slots (5×5 grid network, light traffic). We let PDD broadcast exponential load values every 10 slots so that there are at least seven broadcast cycles in a typical simulation.

Since the heterogeneous link capacities and packet arrivals are generated randomly, we report the average performance of each algorithm over 100 runs given a particular combination of network topology, link capacity setting, and traffic pattern.[4] We measure the performance of an algorithm by its delivery ratio. Fig. 6 and Fig. 7 compare the delivery ratios of different algorithms against different values of R for each of the eight combinations of network topology, link capacity setting, and traffic pattern respectively. We can see that our PD and PDSS algorithms outperform the two baseline algorithms in all figures. PD and PDSS typically have similar performance and in some cases PDSS is slightly better. Our distributed algorithm PDD is better than the two baseline algorithms in most cases, and the gap is larger under heavy traffic. When R is small, PDD can be slightly worse than EDF but is still much better than MKS. We also note that both baseline policies are

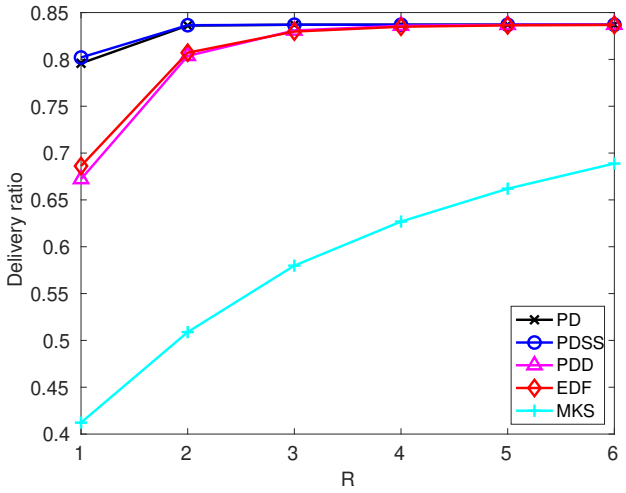
[4]Due to excessive memory usage and running time, we report the performance of MKS based on the first 10^3 packets of each packet arrival sequence.



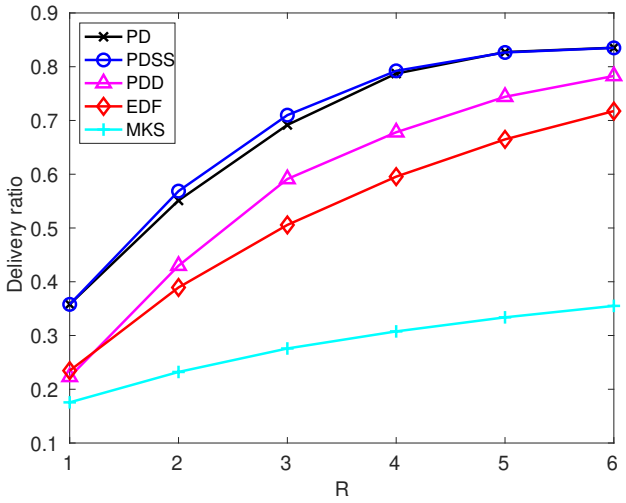
(a) Homogeneous link capacity, light traffic



(b) Homogeneous link capacity, heavy traffic



(c) Heterogeneous link capacities, light traffic



(d) Heterogeneous link capacities, heavy traffic

Fig. 7. Delivery ratio comparison for the 5×5 grid network.

centralized policies, while PDD is a distributed protocol. MKS performs poorly in our simulations because the condition in Theorem 3 in [2] is not met and the control parameter μ therein is not fine tuned.

By comparing these figures, we can also observe that generally the performance under light traffic is better than that under heavy traffic, and the performance under homogeneous link capacity is better than that under heterogeneous link capacities. These results confirm the intuition that heavy traffic and heterogeneous link capacities degrade the network performance. Besides, note that in the small network, the best algorithms can deliver almost all packets with large capacity redundancy R, while in the larger 5×5 grid network, the performance saturates at a delivery ratio of over 80%. The reason is that the relative deadline can be smaller than the distance between the source and the destination in the 5×5 grid network, making it impossible to deliver some packets in time.

Furthermore, we focus on PDD and study the impact of broadcast frequency over its delivery ratio performance. Fig. 8 shows the simulation results with the small network, homogeneous link capacity, and heavy traffic, where we report the average performance of PDD over 100 runs for each R and broadcast period T_b. We can observe the performance degradation with infrequent broadcasts (i.e. large broadcast period T_b). However, the degradation is not significant and becomes negligible when R is large.

IX. Conclusion

In this paper, we have presented our study on online routing and scheduling with capacity redundancy for timely delivery guarantees in multihop networks. We have answered the question that how much capacity redundancy is needed for online algorithms to guarantee certain timely delivery requirements. We have proposed an online algorithm PD for routing and scheduling as packets arrive in the network. The algorithm is proved to be $(R, 1 + \frac{L}{e^R - 1})$-competitive, where L is the length of the longest path. We have also showed that the complexity of PD is $O(ET)$, where E is the total number of links and T is the maximum relative deadline.

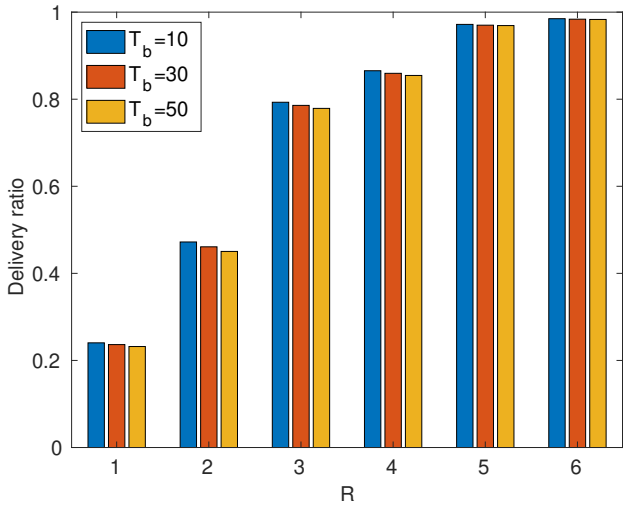


Fig. 8. Delivery ratio comparison of different PDD broadcast periods T_b. As an example, $T_b = 10$ means PDD broadcasts every 10 slots.

Next, we have showed that any online algorithm cannot be better than $(R, 1 + \frac{L-2e^R}{(L+1)e^R-L})$-competitive. When both L and the required deliver ratio are large, our PD algorithm requires at most twice as much capacity as the lower bound. In addition, we have proposed another online algorithm PDSS, which is proved to be $(1, O(\log L))$-competitive, and thus order optimal when $R = 1$. Furthermore, we have proposed a heuristic distributed algorithm PDD that only requires infrequent broadcast of load information. Simulation results have demonstrated that our algorithms outperform EDF and MKS scheduling algorithms with shortest path routing in various network settings.

There remain many interesting open problems for future research. The PDSS algorithm is proved to be order optimal only when $R = 1$. However, simulation results suggest that PDSS still performs very well when $R > 1$. It would be interesting to study online algorithms that are order optimal for all R. It is also of great interest to study the competitiveness of distributed algorithms.

REFERENCES

[1] H. Deng and I.-H. Hou, "On the capacity requirement for arbitrary end-to-end deadline and reliability guarantees in multi-hop networks," in *Proc. 2017 ACM SIGMETRICS Abstracts*. ACM Press, 2017.

[2] Z. Mao, C. E. Koksal, and N. B. Shroff, "Optimal online scheduling with arbitrary hard deadlines in multihop communication networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 177–189, Feb 2016, errata: http://newslab.ece.ohio-state.edu/research/resources/Mao_errata.pdf.

[3] N. Buchbinder and J. S. Naor, "The design of competitive online algorithms via a primal-dual approach," *Foundations and Trends in Theoretical Computer Science*, vol. 3, no. 2–3, pp. 93–263, 2009.

[4] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973.

[5] 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," in *Proc. 12th Real-Time Systems Symposium*, Dec. 1991, pp. 106–115.

[6] S. A. Goldman, J. Parwatikar, and S. Suri, "Online scheduling with hard deadlines," *Journal of Algorithms*, vol. 34, no. 2, pp. 370–389, 2000.

[7] M. H. Goldwasser and B. Kerbikov, "Admission control with immediate notification," *J. of Scheduling*, vol. 6, no. 3, pp. 269–285, May 2003.

[8] M. H. Goldwasser, "Patience is a virtue: The effect of slack on competitiveness for admission control," *Journal of Scheduling*, vol. 6, no. 2, pp. 183–211, 2003.

[9] M. H. Goldwasser and M. Pedigo, "Online nonpreemptive scheduling of equal-length jobs on two identical machines," *ACM Trans. Algorithms*, vol. 5, no. 1, pp. 2:1–2:18, Dec. 2008.

[10] J. Ding and G. Zhang, *Online Scheduling with Hard Deadlines on Parallel Machines*. Springer Berlin Heidelberg, 2006, pp. 32–42.

[11] J. Ding, T. Ebenlendr, J. Sgall, and G. Zhang, *Online Scheduling of Equal-Length Jobs on Parallel Machines*. Springer Berlin Heidelberg, 2007, pp. 427–438.

[12] T. Ebenlendr and J. Sgall, "A lower bound for scheduling of unit jobs with immediate decision on parallel machines," in *Approximation and Online Algorithms*. Springer Berlin Heidelberg, 2009, pp. 43–52.

[13] M. Andrews and L. Zhang, "Packet routing with arbitrary end-to-end delay requirements," in *Proc. 31st Annu. ACM Symp. Theory of Computing (STOC '99)*. ACM, 1999, pp. 557–565.

[14] P. P. Bhattacharya, L. Tassiulas, and A. Ephremides, "Optimal scheduling with deadline constraints in tree networks," *IEEE Trans. Autom. Control*, vol. 42, no. 12, pp. 1703–1705, Dec. 1997.

[15] I.-H. Hou, "Packet scheduling for real-time surveillance in multihop wireless sensor networks with lossy channels," *IEEE Trans. Wireless Commun.*, vol. 14, no. 2, pp. 1071–1079, Feb. 2015.

[16] R. Li and A. Eryilmaz, "Scheduling for end-to-end deadline-constrained traffic with reliability requirements in multihop networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1649–1662, Oct. 2012.

[17] R. Singh and P. R. Kumar, "Decentralized throughput maximizing policies for deadline-constrained wireless networks," in *2015 54th IEEE Conf. Decision and Control (CDC)*, Dec. 2015, pp. 3759–3766.

[18] ——, "Throughput optimal decentralized scheduling of multi-hop networks with end-to-end deadline constraints: Unreliable links," *IEEE Trans. Autom. Control*, to be published.

[19] H. Li, Y. Cheng, C. Zhou, and W. Zhuang, "Minimizing end-to-end delay: A novel routing metric for multi-radio wireless mesh networks," in *IEEE INFOCOM 2009*, Apr. 2009, pp. 46–54.

[20] X. Liu and L. Ying, "Spatial-temporal routing for supporting end-to-end hard deadlines in multi-hop networks," in *2016 Annu. Conf. Information Science and Systems (CISS)*, March 2016, pp. 262–267.

[21] Q. Wang, P. Fan, D. O. Wu, and K. B. Letaief, "End-to-end delay constrained routing and scheduling for wireless sensor networks," in *2011 IEEE Int. Conf. Communications (ICC)*, Jun. 2011.

[22] W. Aiello, R. Ostrovsky, E. Kushilevitz, and A. Rosén, "Dynamic routing on networks with fixed-size buffers," in *Proc. 14th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA '03)*. Society for Industrial and Applied Mathematics, 2003, pp. 771–780.

[23] S. Angelov, S. Khanna, and K. Kunal, "The network as a storage device: Dynamic routing with bounded buffers," *Algorithmica*, vol. 55, no. 1, pp. 71–94, Sep. 2009.

[24] G. Even and M. Medina, "Online packet-routing in grids with bounded buffers," *Algorithmica*, vol. 78, no. 3, pp. 819–868, 2017.

[25] G. Even, M. Medina, and B. Patt-Shamir, "Better deterministic online packet routing on grids," in *Proc. 28th ACM Symp. Parallelism in Algorithms and Architectures (SPAA '15)*. ACM, 2015, pp. 284–293.

[26] V. Rodoplu, S. Vadvalkar, A. A. Gohari, and J. J. Shynk, "Empirical modeling and estimation of end-to-end VoIP delay over mobile multi-hop wireless networks," in *2010 IEEE Global Telecommunications Conf. (GLOBECOM 2010)*, Dec. 2010, pp. 1–6.

[27] K. Sanada, N. Komuro, and H. Sekiya, "End-to-end throughput and delay analysis for IEEE 802.11 string topology multi-hop network using Markov-chain model," in *2015 IEEE 26th Annu. Int. Symp. Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Aug. 2015, pp. 1697–1701.

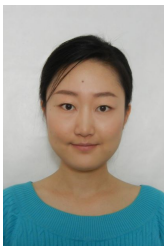
[28] W. Jiao, M. Sheng, K. S. Lui, and Y. Shi, "End-to-end delay distribution analysis for stochastic admission control in multi-hop wireless networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 3, pp. 1308–1320, Mar. 2014.

[29] N. Buchbinder and J. Naor, "Improved bounds for online routing and packing via a primal-dual approach," in *2006 47th Annu. IEEE Symp. Foundations of Computer Science (FOCS '06)*. IEEE, 2006.

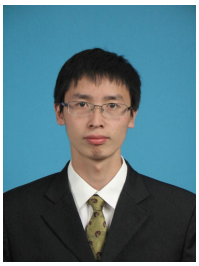
[30] H. Deng and I.-H. Hou, "Online job allocation with hard allocation ratio requirement," in *IEEE INFOCOM 2016*, 2016.

[31] ——, "Optimal capacity provisioning for online job allocation with hard allocation ratio requirement," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 724–736, Apr. 2018.

[32] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "The price of being near-sighted," in *Proc. 17th Annu. ACM-SIAM Symp. Discrete Algorithm (SODA '06)*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2006, pp. 980–989.



Han Deng received the B.S. degree in information engineering from the Beijing Institute of Technology, Beijing, China, in 2009, the M.S. degree in electrical and computer engineering from Oakland University, MI, USA, in 2012, and the Ph.D. degree in computer engineering from Texas A&M University in 2017. She is currently a Post-Doctoral Fellow with the Houston Methodist Research Institute, USA. Her research interests are in optimization and machine learning.



Tao Zhao received his B.Eng. and M.Sc. degrees from Tsinghua University, Beijing, China, in 2012 and 2015, respectively. He is currently a PhD student in Department of Electrical & Computer Engineering, Texas A&M University, College Station, Texas, United States. His current research interests include wireless networks, cloud-based systems, and networked systems. He received the Best Student Paper Award in WiOpt 2017.

I-Hong Hou (S'10–M'12) received the B.S. in Electrical Engineering from National Taiwan University in 2004, and his M.S. and Ph.D. in Computer Science from University of Illinois, Urbana-Champaign in 2008 and 2011, respectively.

In 2012, he joined the department of Electrical and Computer Engineering at the Texas A&M University, where he is currently an Associate Professor. His research interests include wireless networks, wireless sensor networks, real-time systems, distributed systems, and vehicular ad hoc networks.

Dr. Hou received the Best Paper Award in ACM MobiHoc 2017, the Best Student Paper Award in WiOpt 2017, and the C.W. Gear Outstanding Graduate Student Award from the University of Illinois at Urbana-Champaign.