# Learning-augmented Online Minimization of Age of Information and Transmission Costs

Zhongdong Liu, Keyuan Zhang, Bin Li, Yin Sun, Y. Thomas Hou, and Bo Ji

***Abstract*—We consider a discrete-time system where a resource-constrained source (e.g., a small sensor) transmits its time-sensitive data to a destination over a time-varying wireless channel. Each transmission incurs a fixed transmission cost (e.g., energy cost), and no transmission results in a staleness cost represented by the *Age-of-Information*. The source must balance the tradeoff between transmission and staleness costs. To address this challenge, we develop a robust online algorithm to minimize the sum of transmission and staleness costs, ensuring a worst-case performance guarantee. While online algorithms are robust, they are usually overly conservative and may have a poor average performance in typical scenarios. In contrast, by leveraging historical data and prediction models, machine learning (ML) algorithms perform well in average cases. However, they typically lack worst-case performance guarantees. To achieve the best of both worlds, we design a learning-augmented online algorithm that exhibits two desired properties: (i) *consistency*: closely approximating the optimal offline algorithm when the ML prediction is accurate and trusted; (ii) *robustness*: ensuring worst-case performance guarantee even ML predictions are inaccurate. Finally, we perform extensive simulations to show that our online algorithm performs well empirically and that our learning-augmented algorithm achieves both consistency and robustness.**

***Index Terms*—Age-of-Information, transmission cost, online algorithm, learning-augmented algorithm.**

## I. Introduction

In recent years, we have witnessed the swift and remarkable development of the Internet of Things (IoT), which connects billions of entities through wireless networks [1]. These entities range from small, resource-constrained sensors (e.g., smart cameras and temperature sensors) to powerful smartphones. Among various IoT applications, one most important category is real-time IoT applications, which require timely information updates from the IoT sensors. For example, in industrial automation systems [2], [3], battery-powered IoT sensors are deployed to provide data for monitoring equipment health and product quality. On the one hand, IoT sensors are usually small and have limited battery capacity, and thus frequent transmissions drain the battery quickly; on the other hand, occasional transmissions render the information at the controller outdated, leading to detrimental decisions. In addition, wireless channels can be unreliable due to potential channel fading, interference, and the saturation of wireless networks if the traffic load generated by numerous sensors is high [4]. Clearly, under unreliable wireless networks, IoT sensors need to transmit wisely to balance the tradeoff between transmission cost (e.g., energy cost) and data freshness. Other applications include smart grids, smart cities, and so on.

To this end, in the first part of this work, we study the tradeoff between transmission cost and data freshness under a time-varying wireless channel. Specifically, we consider a discrete-time system where a device transmits its data to an access point over an ON/OFF wireless channel (i.e., transmissions happen only when the channel is ON). Each transmission incurs a fixed *transmission cost*, while no transmission results in a *staleness cost* represented by the *Age-of-Information (AoI)* [5], which is defined as the time elapsed since the generation time of the freshest delivered packet. To minimize the sum of transmission costs and staleness costs, we develop a robust online algorithm that achieves a competitive ratio (CR) of 3. That is, different from typical studies with stationary network assumptions , the cost of our online algorithm is at most three times larger than that of the optimal offline algorithm under the worst channel state (see the definition of CR in Section III).

While online algorithms exhibit robustness against the worst-case situations, they tend to be overly conservative and may have a poor average performance in real-world scenarios. On the other hand, by exploiting historical data to build prediction models, machine learning (ML) algorithms can excel in average cases. Nonetheless, ML algorithms could be sensitive to disparity in training and testing data due to distribution shifts or adversarial examples, resulting in subpar performance and lacking worst-case performance guarantees.

To that end, we design a novel learning-augmented online algorithm that takes advantage of both online and ML algorithms. Specifically, our learning-augmented online algorithm integrates ML prediction (a sequence of times indicating when to transmit) into our online algorithm and achieves two desired properties: (i) *consistency*: when the ML prediction is accurate and trusted, our learning-augmented algorithm performs closely to the optimal offline algorithm, and (ii) *robustness*: even when the ML prediction is inaccurate, our learning-augmented algorithm still offers a worst-case guarantee.

Our main contributions are as follows.

*First*, we study the tradeoff between transmission cost and data freshness in a time-varying wireless channel by formulating an optimization problem to minimize the sum of transmission and staleness costs under an ON/OFF channel.

*Second*, following the approach in [6], we reformulate our

Zhongdong Liu (zhongdong@vt.edu), Keyuan Zhang (keyuanz@vt.edu), and Bo Ji (boji@vt.edu) are with the Department of Computer Science, Virginia Tech, Blacksburg, VA. Bin Li (binli@psu.edu) is with the Department of Electrical Engineering, Pennsylvania State University, University Park, PA. Yin Sun (yzs0078@auburn.edu) is with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL. Y. Thomas Hou (thou@vt.edu) is with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA.

(non-linear) optimization problem into a linear Transmission Control Protocol (TCP) acknowledgment problem [7] and propose a primal-dual-based online algorithm that achieves a CR of 3. While a similar primal-dual-based online algorithm has been claimed to asymptotically achieve a CR of $e/(e-1)$ [6], there is a technical issue in their analysis (see Remark 1).

*Third*, by incorporating ML predictions into our online algorithm, we design a novel learning-augmented online optimization algorithm that achieves both consistency and robustness. *To the best of our knowledge, this is the first study on AoI that incorporates ML predictions into online optimization to achieve consistency and robustness.*

*Finally*, we perform extensive simulations using both synthetic and real trace data. Our online algorithm exhibits better performance than the theoretical analysis, and our learning-augmented algorithm can achieve consistency and robustness.

Due to space limitations, we omit all the proofs and provide them in our online technical report

## II. Related Work

Since AoI was introduced in [5], it has sparked numerous studies on this topic (see surveys in [8], [9]). Among these AoI studies, two categories are most relevant to our work.

The first category includes studies that consider the joint minimization of AoI and certain costs [10]–[13]. The work of [10] studies the problem of minimizing the average cost of sampling and transmission over an unreliable wireless channel subject to average AoI constraints. In [12], the authors consider a source-monitor pair with stochastic arrival of packets at the source. The source pays a transmission cost to send the packet, and its goal is to minimize the weighted sum of AoI and transmission costs. Here the packet arrival process is assumed to follow certain distributions. Although the assumptions in these studies lead to tractable performance analysis, such assumptions may not hold in practical scenarios.

The second category contains studies that focus on non-stationary settings [6], [14]–[16]. For example, in [14], the authors proposed online algorithms to minimize the AoI of users in a cellular network under adversarial wireless channels. In these AoI works that consider non-stationary settings, the most relevant work to ours is [6], where the authors study the minimization of the sum of download costs and AoI costs under an adversarial wireless channel. A primal-dual-based randomized online algorithm is shown to have an asymptotic CR of $e/(e-1)$. However, there is a technical issue in their analysis (see Remark 1). To address this issue, we propose an online primal-dual-based algorithm that achieves a CR of 3 in the non-asymptotic regime. While the AoI optimization problems under non-stationary settings have been investigated, none of them considers applying ML predictions to improve the average performance of online algorithms.

In recent works [17]–[20], researchers attempt to take advantage of both online algorithms and ML predictions, i.e., to design a learning-augmented online algorithm that achieves consistency and robustness. In the seminal work [17], by incorporating ML predictions into the online Marker algorithm, the authors can achieve consistency and robustness for a caching problem. Following [17], a large body of research works in this direction have emerged. The most related learning-augmented work to ours is [19], where the authors design a primal-dual-based learning-augmented algorithm for the TCP acknowledgment problem. In [19], the uncertainty comes from the packet arrival times, and the controller can make decisions at any time. In our work, however, the uncertainty comes from the channel states, and no data can be transmitted when the channel is OFF. Lacking the freedom to transmit data at any time makes their algorithm inapplicable.

## III. System Model and Problem Formulation

***System Model.*** Consider a status-updating system where a resource-constrained device transmits time-sensitive data to an access point (AP) via an unreliable wireless channel. The system operates in discrete time slots, denoted by $t = 1, 2, \dots, T$, where $T$ is finite and is allowed to be arbitrarily large. The channel state at time $t$ is represented by $s(t) \in \{0, 1\}$, where $s(t) = 1$ means the channel is ON, and the device can access the AP; while $s(t) = 0$ means the channel is OFF, and the device cannot access the AP. The sequence of channel states over the time horizon are $\mathbf{s} = \{s(1), \dots, s(T)\}$.

At the beginning of each slot, the device probes to know the current channel state and then decides whether to transmit its freshest data to the AP or not. Let $d(t) \in \{0, 1\}$ be the transmission decision at slot $t$, where $d(t) = 1$ if the device decides to transmit, i.e., generates a new update and sends it to the AP; and $d(t) = 0$ if not. A scheduling algorithm $\pi$ is denoted by $\pi = \{d^\pi(t)\}_{t=1}^T$, where $d^\pi(t)$ is the transmission decisions made by algorithm $\pi$ at time $t$. For simplicity, we use $\pi = \{d(t)\}_{t=1}^T$ in the rest of the paper. If the device decides to transmit and the channel is ON, it needs to pay a constant *transmission cost* of $c > 1$, and the data at the AP is successfully updated at the end of slot $t$; otherwise, the data at the AP becomes staler.[1] To measure the freshness of data on the AP side, we use a metric called *Age-of-Information (AoI)* [5], which is defined as the time elapsed since the freshest received update was generated. We use $a(t)$ to denote the AoI at time $t$, which evolves as

$$a(t) = \begin{cases} 0, & \text{if } s(t) \cdot d(t) = 1; \\ a(t-1) + 1, & \text{otherwise}, \end{cases} \tag{1}$$

where the AoI drops to 0 if the device transmits at some ON slot, otherwise, the AoI increases by 1.[2] We assume $a(0) = 0$. To reflect the penalty of the device that the AP does not get the update at time $t$, we introduce a *staleness cost*, which is represented by the AoI at time $t$.

***Problem Formulation.*** The total cost of an algorithm $\pi$ is

$$C(\mathbf{s}, \pi) \triangleq \sum_{t=1}^{T} (c \cdot d(t) + a(t)), \tag{2}$$

[1] If the transmission cost $c$ is no larger than 1, which is less than the staleness cost (at least 1), then the optimal policy is to transmit at every ON slot.

[2] Some studies also assume that the AoI drops to be 1, wherein our analysis still holds. We let the AoI drop to 0 to make the discussion concise.

where the first item $c \cdot d(t)$ is the transmission cost at time $t$, and the second item $a(t)$ is the staleness cost at time $t$. In this paper, we focus on the class of *online* scheduling algorithms, under which the information available at time $t$ for making decisions includes the transmission cost $c$, the transmission history $\{d(\tau)\}_{\tau=1}^{t}$, and the channel state pattern $\{s(\tau)\}_{\tau=1}^{t}$, while the time horizon $T$ and the future channel state $\{s(\tau)\}_{\tau=t+1}^{T}$ is unknown. Conversely, an *offline* scheduling algorithm has the information about the connectivity pattern $\mathbf{s}$ (and the time horizon $T$) beforehand.

Our goal is to develop an online algorithm $\pi$ that minimizes the total cost given a channel state pattern $\mathbf{s}$:

$$\min_{d(t)} \quad \sum_{t=1}^{T} (c \cdot d(t) + a(t)) \tag{3a}$$

$$\text{s.t.} \quad d(t) \in \{0,1\} \text{ for } t = 1,2,\ldots,T; \tag{3b}$$

$$a(t) \text{ evolves as Eq. (1) for } t = 1,2,\ldots,T. \tag{3c}$$

In problem (3), the only decision variable is the transmission decision $d(t)$, and the objective function is a non-linear function of $d(t)$ due to the dependence of $a(t)$ on $d(t)$. To see this, consider a simple example where the current time is $t$ and there was no transmission made during $[1, t-1]$. The total cost at time $t$ is $(1 - d(t)) \cdot \sum_{\tau=1}^{t} (1 - d(\tau)) + c \cdot d(t)$, which a quadratic function of the transmission decision $d(t)$. This non-linearity poses a challenge to its efficient solutions. In Section IV-A, following a similar line of analysis as in [6], we reformulate problem (3) to an equivalent TCP acknowledgment (ACK) problem, which is linear and can be solved efficiently (e.g., via the primal-dual approach).

To measure the performance of an online algorithm, we use the metric *competitive ratio* (CR) [21], which is defined as the worst-case ratio of the cost under the online algorithm to the cost of the optimal offline algorithm. Formally, we say that an online algorithm $\pi$ is *$\beta$-competitive* if there exists a constant $\beta \geq 1$ such that for any channel state pattern $\mathbf{s}$,

$$C(\mathbf{s}, \pi) \leq \beta \cdot OPT(\mathbf{s}), \tag{4}$$

where $OPT(\mathbf{s})$ is the cost of the optimal offline algorithm for the given channel state $\mathbf{s}$. We desire to develop an online algorithm with a CR close to 1, which implies that our online algorithm performs closely to the optimal offline algorithm.

## IV. Robust Online Algorithm

In this section, we first reformulate our AoI problem (3) to an equivalent linear TCP ACK problem. Then, this TCP ACK problem is further relaxed to a linear primal-dual-based program. Finally, a 3-competitive online algorithm is developed to solve the linear primal-dual-based program.

### A. Problem Reformulation

Following the approach in [6], we reformulate the non-linear problem (3) to an equivalent linear TCP ACK problem (5) as follows. Consider a TCP ACK problem, where the source reliably generates and delivers one packet to the destination in each slot $t = 1,2,\ldots,T$. Those delivered packets need to be acked (for simplicity, we use "acked" instead of "acknowledged" in the rest of the paper) that they are received by the destination, which requires the destination to send ACK packets (for brevity, we call it ACK) back to the source. We use $d(t) \in \{0,1\}$ to denote the decision made by the destination at slot $t$. Let $z_i(t) \in \{0,1\}$ represent whether packet $i$ (i.e., the packet sent at slot $i$) has been acked by slot $t$ ($i \leq t$), where $z_i(t) = 1$ if packet $i$ is not acked by slot $t$ and $z_i(t) = 0$ otherwise. Once packet $i$ is acked at slot $t$, then it is acked forever after slot $t$, i.e., $z_i(\tau) = 0$ for all $i \leq t$ and $\tau \geq t$. The feedback channel is unreliable and its channel state in slot $t$ is modeled by an ON/OFF binary variable $s(t) \in \{0,1\}$. We use $\mathbf{s} = \{s(1), \ldots, s(T)\}$ to denote the entire feedback channel states. The destination has access to the feedback channel state $s(t)$ at the beginning of each slot $t$. When the feedback channel is ON and the destination decides to send an ACK, all previous packets are acked, i.e., the number of unacked packets becomes 0; otherwise, the number of unacked packets increases by 1. We can see that the dynamic of the number of unacked packets is the same as the AoI dynamic.

We assume that each unacked packet at one slot incurs a holding cost of one unit for the destination. Here, the holding cost of packet $i$ at slot $t$ can be represented by $z_i(t)$, and the total holding cost at slot $t$ is $\sum_{i=1}^{t} z_i(t)$. In addition, we also assume that each ACK has an ACK cost of $c$. The goal of the TCP ACK problem is to develop an online scheduling algorithm $\pi = \{d(t)\}_{t=1}^{T}$ that minimizes the total cost given a feedback channel state pattern $\mathbf{s}$:

$$\min_{d(t), z_i(t)} \quad \sum_{t=1}^{T} \left( c \cdot d(t) + \sum_{i=1}^{t} z_i(t) \right) \tag{5a}$$

$$\text{s.t.} \quad z_i(t) + \sum\nolimits_{\tau=i}^{t} s(\tau) d(\tau) \geq 1$$

$$\text{for } i \leq t \text{ and } t = 1,2,\ldots,T; \tag{5b}$$

$$d(t), z_i(t) \in \{0,1\} \text{ for } i \leq t \text{ and } t = 1,2,\ldots,T, \tag{5c}$$

where the first item $c \cdot d(t)$ in Eq. (5a) is the ACK cost at slot $t$, the second item $\sum_{i=1}^{t} z_i(t)$ in Eq. (5a) is the holding cost at slot $t$. Constraint (5b) states that for packet $i$ at slot $t$, either this packet is not acked (i.e., $z_i(t) = 1$) or an ACK was made since its arrival (i.e., $s(\tau)d(\tau) = 1$ for some $i \leq \tau \leq t$). While Problem (5) is an integer linear problem, we demonstrate its equivalence to Problem (3) in the following.

*Lemma* 1. *Problem* (5) *is equivalent to Problem* (3).

We provide the detailed proof in our technical report[Bo: add a ref. - for now, you can use your github link. If all the proofs are omitted, then just say it once at the end of the introduction - check Keyuan's paper] and give a proof sketch as follows. We can show that: (i) any feasible solution to problem (3) can be converted to a feasible solution to problem (5), and the total costs of these two solutions are the same; (ii) any feasible solution to problem (5) can be converted to a feasible solution to problem (3), and the total cost of the converted solution to problem (3) is no greater than the total cost of the solution to problem (5). This implies that any optimal solution to problem (3) is also an optimal solution to problem (5), and vice versa. Therefore, these two problems are equivalent [22, Sec. 4.1.3].

To obtain a linear program of the integer problem (5), we relax the integer requirement to real numbers:

$$\min_{d(t),z_i(t)} \sum_{t=1}^{T}\left(c\cdot d(t)+\sum_{i=1}^{t} z_i(t)\right) \tag{6a}$$

$$\text{s.t.}\quad z_i(t)+\sum\nolimits_{\tau=i}^{t} s(\tau)d(\tau)\geq 1$$

$$\text{for } i\leq t \text{ and } t=1,2,\ldots,T; \tag{6b}$$

$$d(t),z_i(t)\in[0,1] \text{ for } i\leq t \text{ and } t=1,2,\ldots,T, \tag{6c}$$

which is referred to as the primal program. The corresponding dual problem of problem (6) is as follows:

$$\max_{y_i(t)} \quad \sum_{t=1}^{T}\sum_{i=1}^{t} y_i(t) \tag{7a}$$

$$\text{s.t.}\quad s(t)\sum_{i=1}^{t}\sum_{\tau=t}^{T} y_i(\tau)\leq c \text{ for } t=1,2,\ldots,T; \tag{7b}$$

$$y_i(t)\in[0,1] \text{ for } i\leq t \text{ and } t=1,2,\ldots,T, \tag{7c}$$

which has a dual variables $y_i(t)$ for packet $i$ and time $t\geq i$.

## B. Primal-dual Online Algorithm Description and Analysis

To solve the primal-dual problems (6) and (7), we develop the Primal-dual-based Online Algorithm (PDOA) and present it in Algorithm 1. The input is the channel state pattern s (revealed in an online manner), and the outputs are the primal variables $d(t)$ and $z_i(t)$, and the dual variable $y_i(t)$. Two auxiliary variables $L$ and $M$ are also introduced: $L$ denotes the time when the latest ACK was made and $M$ denotes the ACK marker (PDOA should make an ACK when $M\geq 1$).

PDOA is a threshold-based algorithm. Assuming that the latest ACK was made at slot $L$, when the accumulated holding costs since slot $L+1$ is no smaller than the ACK cost $c$ (i.e., $M\geq 1$), PDOA will make an ACK at the next ON slot $L'$. Here, we call the interval $[L+1,L']$ an ACK interval. We can observe that PDOA updates the primal variables and dual variables only for the packets that are not acked in the current ACK interval $[L+1,L']$. More specifically, assuming that the current slot is slot $t\in[L+1,L']$ that has not been acked by slot $t$: (i) for the primal variable $z_i(t)$, if the threshold is not achieved ($M<1$) or the channel is OFF at slot $t$, PDOA will update $z_i(t)$ to be 1 (in Line 4 or Line 15, respectively) since packet $i$ is not acked by slot $t$; (ii) for the dual variable $y_i(t)$, if packet $i$ is not the last packet in the current ACK interval, PDOA will update $y_i(t)$ to be 1 to maximize the dual objective function; otherwise, PDOA will update $y_i(t)$ to $c-c\cdot M$ to ensure that when the threshold is achieved ($M\geq 1$), the sum of all the dual variables is exactly $c$.

In addition, at the end of slot $t$ (i.e., Lines 19-25), assuming that the most recent ON slot is slot $t^\dagger\in[L,t]$ and the channels are OFF during $[t^\dagger+1,t)$. In the case that slot $t$ is an ON channel, PDOA will skip slot $t$ and go to slot $t$+1, to maximize

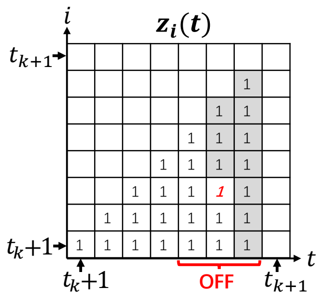
**Algorithm 1:** Primal-dual-based Online Algorithm (PDOA)

```
   Input  : s (revealed in an online manner)
   Output: $d(t), z_i(t), y_i(t)$
   Init.: $d(t), z_i(t), y_i(t), L, M \leftarrow 0$ for all $i$ and $t$
1  for $t = 1$ to $T$ do
       /* Iterate all the packets arriving
          since the latest ACK time $L$.          */
2      for $i = L+1$ to $t$ do
3          if $M < 1$ then        /* Not ready to ACK */
4              $z_i(t) \leftarrow 1$;
5              $M \leftarrow M + 1/c$;
6              $y_i(t) \leftarrow \min\{1, c - c\cdot M\}$;
7          end
8          if $M \geq 1$ then              /* Ready to ACK */
9              if $s(t) = 1$ then          /* ON channel */
10                 $d(t) \leftarrow 1$;
11                 $M \leftarrow 0$;
12                 $L \leftarrow t$;
13                 break and go to the next slot (i.e., $t+1$);
14             else                        /* OFF channel */
15                 $z_i(t) \leftarrow 1$;
16             end
17         end
18     end
       /* At the end of slot $t$, update dual
          variable $y_i(t)$ with $s(i) = 0$ as:    */
19     for $i = t$ decrease to $L+1$ do
20         if $s(i) = 0$ then
21             $y_i(t) \leftarrow 1$;
22         else
23             break and go to the next slot;
24         end
25     end
26 end
```

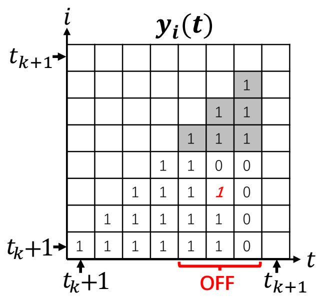
the dual objective function, PDOA updates the dual variables of packet $t^\dagger+1$ to packet $t$ to be 1 since the channels are OFF during $[t^\dagger+1,t)$ and the updating of their dual variables does not violate constraint (7b) (see an illustration in Fig. 1(b)). Note that there may be some OFF channels before slot $t^\dagger$, but PDOA does not update their dual variables to avoid the violation of constraint (7b). For example, assuming that slot $t'$ ($t'<t^\dagger$) is an OFF channel and we let $y_{t'}(t)=1$. Letting $y_{t'}(t)=1$ has no effect on the constraint (7b) at slot $t'$ since we always have $s(t')\sum_{i=1}^{\hat{t}}\sum_{\tau=t'}^{T} y_i(\tau)=0$, but doing this does have effect on the constraint (7b) at slot $t^\dagger$ (i.e., increasing $s(t^\dagger)\sum_{i=1}^{t^\dagger}\sum_{\tau=t^\dagger}^{T} y_i(\tau)$ by 1 because $y_{t'}(t)$ is a part of $s(t^\dagger)\sum_{i=1}^{t^\dagger}\sum_{\tau=t^\dagger}^{T} y_i(\tau)$ as $t'<t^\dagger$ and $t^\dagger\leq t$), possibly making constraint (7b) at slot $t^\dagger$ violated.

*Theorem* 1. *PDOA is 3-competitive.*

The detailed proof will be provided in our technical report and a proof sketch is provided as follows. We first show that given any channel state s, PDOA produces a feasible solution to primal program (6) and dual program (7). Then, we show that in any $k$-th ACK interval, the ratio between the primal objective and the dual objective (denoted by $P(k)$ and $D(k)$,



(a) Primal variables $z_i(t)$ updates.



(b) Dual variables $y_i(t)$ updates.

Fig. 1. The updates of primal variables $z_i(t)$ and dual variables $y_i(t)$ in the $k$-th ACK interval $[t_k+1, t_{k+1}]$, where channels are OFF during $[t_k+5, t_k+7]$. The x-axis represents time and the y-axis represents the packet id. Two ACKs are made at the ON slot $t_k$ and the ON slot $t_{k+1}$, where the ACK cost $c = 18$. The primal variables $z_i(t)$ and dual variables $y_i(t)$ are updated from slot $t_k + 1$ to slot $t_{k+1}$; and in slot $t$, packets are updated from packet $t_k + 1$ to packet $t$. The red bold italic 1 denotes when the ACK marker equals or is larger than 1. In Fig. 1(a), the grey areas denote the updates due to Line 15; In Fig. 1(b), the grey areas denote the updates due to Lines 19-25.

respectively) is at most 3, i.e., $P(k)/D(k) \leq 3$. This implies that the ratio between the total primal objective (denoted by $P$) and the total dual objective (denoted by $D$) is also at most 3, i.e., $P/D \leq 3$. By the weak duality, PDOA is 3-competitive.

*Remark* 1. *An asymptotically $e/(e-1)$-competitive primal-dual-based online algorithm is also proposed in [6]. To maximize the dual objective function, at the end of slot $T$, they update certain dual variables $y_i(t)$ that arrived at OFF slot to be* 1. *In their analysis (specifically, the proof of their Theorem* 7*), they show that because of those dual variables updates at the end of slot $T$, the dual objective value is at least $c$ (i.e., $D \geq c$). In addition, the primal objective value satisfies $P \leq (1 + 1/((1 + 1/c)^{\lfloor c \rfloor} - 1)) \cdot D + (T(T + 1)/2) \cdot (D/c)$. They claim that when $c$ goes to infinity, this bound becomes $P \leq e/(e-1) \cdot D$ as $(T(T+1)/2) \cdot (D/c)$ goes to* 0. *However, this is not true because $D/c \geq 1$ always holds. Therefore, their analysis cannot lead to the conclusion that their algorithm achieves an asymptotically CR of $e/(e-1)$. Furthermore, when $c$ is finite, their CR is a quadratic function of the time horizon $T$, which can be very large when $T$ is long. Instead, in our algorithm, rather than updating these dual variables $y_i(t)$ at the end of slot $T$, we directly update them only in the current ACK interval (i.e., Lines 19-25) and ensure that the dual constraint* (7b) *is satisfied and the dual objective function is as large as possible. This enables us to focus on the analysis of $P(k)/D(k)$ in the current ACK interval, [Bo: finish the sentence - what is the benefit?]*

## V. Learning-Augmented Online Algorithm

Online algorithms are known for their robustness against worst-case scenarios, but they can be overly conservative and may have a poor average performance in typical scenarios. In contrast, ML algorithms leverage historical data to train models that excel in average cases. However, they typically lack worst-case performance guarantees when facing distribution shifts or outliers. To attain the best of both worlds, we design a learning-augmented online algorithm that achieves both consistency and robustness.

### A. Machine Learning Predictions

We consider the case where an ML algorithm provides a prediction $\mathcal{P} \triangleq \{p_1, p_2, \ldots, p_n\}$ that represents the times to transmit an ACK for the destination (i.e., the prediction $\mathcal{P}$ makes a total of $n$ ACKs and sends the $i$-th ACK at slot $p_i$). The prediction $\mathcal{P}$ is unaware of the channel state pattern $\mathbf{s}$ and can be provided either in full in the beginning (i.e., $t = 0$) or be provided one-by-one in each slot. Furthermore, when the prediction $\mathcal{P}$ decides to send an ACK at an OFF slot, we will simply ignore the decision for this particular slot.

Provided with the prediction $\mathcal{P}$, we specify a trust parameter $\lambda \in (0, 1]$ to reflect our confidence in the prediction: a smaller $\lambda$ means higher confidence. The learning-augmented online algorithm takes a prediction $\mathcal{P}$, a trust parameter $\lambda$, and a channel state pattern $\mathbf{s}$ (revealed in an online manner) as inputs, and outputs a solution with a cost of $C(\mathbf{s}, \mathcal{P}, \lambda)$. A learning-augmented algorithm is said $\beta(\lambda)$-*robust* ($\beta(\lambda) \geq 1$) and $\gamma(\lambda)$-*consistent* ($\gamma(\lambda) \geq 1$) if its cost satisfies

$$C(\mathbf{s}, \mathcal{P}, \lambda) \leq \min\{\beta(\lambda) \cdot OPT(\mathbf{s}), \gamma(\lambda) \cdot C(\mathbf{s}, \mathcal{P})\}, \quad (8)$$

where $OPT(\mathbf{s})$ and $C(\mathbf{s}, \mathcal{P})$ is the cost of the optimal offline algorithm and the cost of purely following the prediction $\mathcal{P}$ under the channel state pattern $\mathbf{s}$, respectively.

We aim to design a learning-augmented online algorithm for primal program (6) that exhibits two desired properties (i) *consistency*: when the ML prediction $\mathcal{P}$ is accurate ($C(\mathbf{s}, \mathcal{P}) \approx OPT(\mathbf{s})$) and we trust it, our learning-augmented online algorithm should perform closely to the optimal offline algorithm (i.e., $\gamma(\lambda) \to 1$ as $\lambda \to 0$); and (ii) *robustness*: even if the ML prediction $\mathcal{P}$ is inaccurate, our learning-augmented online algorithm still retains a worst-case guarantee (i.e., $C(\mathbf{s}, \mathcal{P}, \lambda) \leq \beta(\lambda) \cdot OPT(\mathbf{s})$ for any prediction $\mathcal{P}$).

### B. Learning-augmented Online Algorithm Description

We present our Learning-augmented Primal-dual-based Online Algorithm (LAPDOA) in Algorithm 2. LAPDOA behaves similarly to PDOA, but the updates of primal variables and dual variables incorporate the ML prediction $\mathcal{P}$.

In LAPDOA, two additional auxiliary variables $M'$ and $y'$ are used to denote the increment of the ACK marker $M$ and the dual variables $y_i(t)$ in each iteration of update, respectively. Assuming that the current time is $t$, let $\alpha(t)$ denote the next time when the prediction $\mathcal{P}$ sends an ACK (i.e., $\alpha(t) \triangleq \min\{p_i : p_i \geq t\}$ and $\alpha(t) = \infty$ if $t > p_n$). For the updates of primal and dual variables of an unacked packet $i$ at slot $t$, based on the relationship between the current time $t$ and $\alpha(i)$ (which is also the time when the prediction $\mathcal{P}$ makes an ACK for packet $i$ because packet $i$ arrives at slot $i$), we classify them into three types:

- Big updates: those updates make $M' \leftarrow 1/\lambda c$, $y' \leftarrow 1$, and $z_i(t) \leftarrow 1$. The big updates are made when LAPDOA is behind the ACK scheduled by the prediction $\mathcal{P}$ (i.e.,

**Algorithm 2:** Learning-augmented Primal-dual-based Online Algorithm (LAPDOA)

```
Input  : $\mathcal{P}, \lambda$, s (revealed in an online manner)
Output: $d(t), z_i(t), y_i(t)$
Init.: $d(t), z_i(t), y_i(t), L, M \leftarrow 0$ for all $i$ and $t$
1  for $t = 1$ to $T$ do
       /* Iterate all the packets arriving
          since the most recent ACK time $L$. */
2      for $i = L + 1$ to $t$ do
3          if $M < 1$ then        /* Not ready to ACK */
4              if $t \geq \alpha(i)$ then
                   /* Big update: prediction
                      already acked packet $i$     */
5                  $M' \leftarrow 1/\lambda c$, $y' \leftarrow 1$ ;
6              else
                   /* Small update: prediction
                      did not ack packet $i$ yet   */
7                  $M' \leftarrow \lambda/c$, $y' \leftarrow \lambda$ ;
8              end
9              $z_i(t) \leftarrow 1$ ;
10             $M \leftarrow M + M'$;
11             $y_i(t) \leftarrow y'$;
12         end
13         if $M \geq 1$ then           /* Ready to ACK */
14             if $s(t) = 1$ then          /* ON channel */
15                 $d(t) \leftarrow 1$;
16                 $M \leftarrow 0$ ;
17                 $L \leftarrow t$ ;
18                 break and go to the next slot (i.e., $t + 1$);
19             else                       /* OFF channel */
20                 if $z_i(t) \neq 1$ then
                       /* Zero update               */
21                     $z_i(t) \leftarrow 1$;
22                 end
23             end
24         end
25     end
       /* At the end of slot $t$, update dual
          variable $y_i(t)$ with $s(i) = 0$ as:      */
26     for $i = t$ decrease to $L + 1$ do
27         if $s(i) = 0$ then
28             if $y_i(t) = 0$ then
29                 $y_i(t) \leftarrow 1$;
30             end
31         else
32             break and go to the next slot;
33         end
34     end
35 end
```

$t \geq \alpha(i)$), and it tries to catch up the prediction $\mathcal{P}$ by making a big increase in the ACK marker.

- Small updates: those updates make $M' \leftarrow \lambda/c$, $y' \leftarrow \lambda$, and $z_i(t) \leftarrow 1$. The small updates are made when LAPDOA is ahead of the ACK scheduled by the prediction $\mathcal{P}$ (i.e., $t < \alpha(i)$), and LAPDOA tries to slow down its ACK rate by making a small increase in the ACK marker.
- Zero updates: those updates make $M' \leftarrow 0$, $y' \leftarrow 0$, and $z_i(t) \leftarrow 1$. The zero updates are made when LAPDOA is supposed to ACK at some slot $t'$ but finds that slot $t'$ is OFF, and it has to delay its ACK to the next ON slot and pay the holding cost (i.e., $z_i(t) = 1$) along the way.

As we can see, the updates of primal variables and dual variables incorporate the ML prediction $\mathcal{P}$.

## C. Learning-augmented Online Algorithm Analysis

In this subsection, we focus on the consistency and robustness analysis of LAPDOA with $\lambda \in (0, 1]$. The special cases of LAPDOA with $\lambda = 0$ and $\lambda = 1$ correspond to the cases that LAPDOA follows the prediction $\mathcal{P}$ purely and PDOA, respectively. It is noteworthy that by choosing different values of $\lambda$, LAPDOA exhibits a crucial trade-off between consistency and robustness.

*Theorem* 2. *For any channel state pattern* s, *any prediction* $\mathcal{P}$, *any parameter* $\lambda \in (0, 1]$, *and any ACK cost* $c$, *LAPDOA outputs an almost feasible solution (within a factor of* $c/(c+1)$*) with a cost of: when* $\lambda \in (0, 1/c]$,

$$C(\mathbf{s}, \mathcal{P}, \lambda) \leq \min\{(3/\lambda) \cdot ((c+1)/c) \cdot OPT(\mathbf{s}), (1+\lambda)C_H(\mathbf{s}, \mathcal{P}) + C_A(\mathbf{s}, \mathcal{P})\}, \tag{9}$$

*and when* $\lambda \in (1/c, 1]$,

$$C(\mathbf{s}, \mathcal{P}, \lambda) \leq \min\{(3/\lambda) \cdot ((c+1)/c) \cdot OPT(\mathbf{s}), (\lambda+2)C_H(\mathbf{s}, \mathcal{P}) + (1/\lambda + 2) \cdot \lceil \lambda c \rceil \cdot C_A(\mathbf{s}, \mathcal{P})/c\}, \tag{10}$$

*where* $C_A(\mathbf{s}, \mathcal{P})$ *and* $C_H(\mathbf{s}, \mathcal{P})$ *denotes the total ACK costs and total holding costs of prediction* $\mathcal{P}$ *under* s, *respectively.*

Next, we show that LAPDOA has the robustness guarantee in Lemma 2 and the consistency guarantee in Lemma 3. Combining Lemmas 2 and 3, we can conclude Theorem 2.

*Lemma* 2. *(Robustness) For any ON/OFF input instance* s, *any prediction* $\mathcal{P}$, *any parameter* $\lambda \in (0, 1]$, *and any ACK cost* $c$, *LAPDOA outputs a solution which has a cost of*

$$C(\mathbf{s}, \mathcal{P}, \lambda) \leq (3/\lambda) \cdot ((c+1)/c) OPT(\mathbf{s}). \tag{11}$$

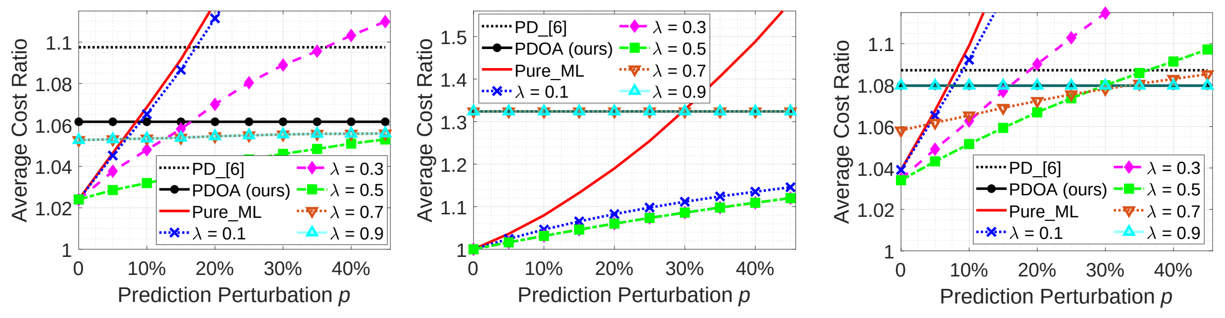
The detailed proof will be provided in our technical report. We provide a proof sketch here. We first show that LAPDOA produces a feasible primal solution and an almost feasible dual solution (with a factor of $c/(c+1)$). Then, we show that in any $k$-th ACK interval, LAPDOA achieves $P(k)/D(k) \leq 3/\lambda$. This implies that LAPDOA also achieves $P/D \leq 3/\lambda$ on the entire instance. Finally, by scaling down all dual variables $y_i(t)$ generated by LAPDOA by a factor of $c/(c+1)$, we obtain a feasible dual solution with a dual objective value of $(c/(c+1)) \cdot D$. By the weak duality, we have $P/OPT \leq P/((c/(c+1)) \cdot D) = (P/D) \cdot ((c+1)/c) \leq (3/\lambda) \cdot ((c+1)/c)$.

*Lemma* 3. *(Consistency) For any channel state pattern* s, *any prediction* $\mathcal{P}$, *any parameter* $\lambda \in (0, 1]$, *and any ACK cost* $c$, *LAPDOA outputs a solution with a cost of: when* $\lambda \in (0, 1/c]$,

$$C(\mathbf{s}, \mathcal{P}, \lambda) \leq (1+\lambda)C_H(\mathbf{s}, \mathcal{P}) + C_A(\mathbf{s}, \mathcal{P}), \tag{12}$$

*and when* $\lambda \in (1/c, 1]$,

$$C(\mathbf{s}, \mathcal{P}, \lambda) \leq (\lambda+2)C_H(\mathbf{s}, \mathcal{P}) + (1/\lambda + 2) \cdot \lceil \lambda c \rceil \cdot C_A(\mathbf{s}, \mathcal{P})/c, \tag{13}$$



(a) Bernoulli process (b) Adversary setting (c) Real trace dataset

Fig. 2. Performance comparison of online and learning-augmented algorithms under different settings.

*where $C_A(\mathbf{s}, \mathcal{P})$ and $C_H(\mathbf{s}, \mathcal{P})$ denotes the total ACK cost and total holding cost of prediction $\mathcal{P}$ under* $\mathbf{s}$*, respectively.*

The detailed proof will be provided in our technical report. We provide a proof sketch as follows. In general, LAPDOA generates three types of updates: big updates, small updates, and zero updates. Our idea is to bound the total cost of each type of update by the cost of the algorithm that purely follows the prediction. In the case of $\lambda \in (0, 1/c]$, we show that the total number of big updates is $C_A(\mathbf{s}, \mathcal{P})/c$, and each big update increases the primal objective value by $c$, so the total cost of big updates in LAPDOA is $C_A(\mathbf{s}, \mathcal{P})$. In addition, the total number of small updates and zero updates can be shown to be bounded by $C_H(\mathbf{s}, \mathcal{P})$, and each small update or zero update incurs a cost at most $1 + \lambda$, thus the total cost of small and zero updates in LAPDOA is at most $(1 + \lambda)C_H(\mathbf{s}, \mathcal{P})$. In summary, the total cost of LAPDOA in the case of $\lambda \in (0, 1/c]$ is bounded by $(1 + \lambda)C_H(\mathbf{s}, \mathcal{P}) + ((c + 1)/c)C_A(\mathbf{s}, \mathcal{P})$. A similar bound can be also obtained for the case of $\lambda \in (1/c, 1]$.

*Remark* 2. *When we trust the ML prediction (i.e., $\lambda \to 0$) and the ML prediction is accurate at the same time ($C(\mathbf{s}, \mathcal{P}) \approx OPT(\mathbf{s})$), our learning-augmented algorithm also performs nearly to the optimal offline algorithm, achieving consistency.*

*Remark* 3. *If the ML prediction is untrustworthy (i.e., $\lambda \to 1$), our learning-augmented algorithm performs closely to our online algorithm (has a CR of* 3*), achieving robustness.*

## VI. Numerical Results

In this section, we perform simulations using both synthetic data and real trace data to show that our online algorithm outperforms greatly the theoretical analysis and that our learning-augmented algorithm achieves consistency and robustness.

***DataSets.*** We consider three representative datasets: (i) the Bernoulli process, where the channel state $\mathbf{s}$ is a Bernoulli process with $\mathbb{P}[s(t) = 1] = 0.5$ and the transmission cost $c = 10$; (ii) the adversary setting, where the transmission cost $c = 4$ and the channel state $\mathbf{s}$ is a repeated pattern of [2×ON, 7×OFF, 1×ON]. That is, the pattern begins with two ON slots, followed by seven OFF slots, and one ON slot in the end. Under this pattern, our online algorithm fails to transmit at the 3rd OFF slot and has to delay to the 10th ON slot. [Bin: It is not clear why the online algorithm does not make transmission decisions in the first two slots. Please illustrate it further.] The total cost of our online algorithm is $(9\cdot10)/2+4 = 49$. While for the optimal offline algorithm, it will transmit at the 2nd ON slot and 10th ON slot, and the total cost is $1+4+(7\cdot8)/2+4 = 37$. This setting leads to a high CR of our online algorithm, which is $49/37 \approx 1.324$; (iii) the real trace dataset [23], where the channel state $\mathbf{s}$ is the channel measurements of a mobile user moving around an AP at different distances (2.5m – 30m) with blockage in between and the transmission cost $c = 10$. For all the datasets, the numerical results are the average of $100$ simulation runs, each with $5000$ time slots.

***ML Prediction.*** Recall that our learning-augmented online algorithm is equipped with an ML prediction $\mathcal{P}$ that provides the transmission decision at each slot. To generate such an ML prediction $\mathcal{P}$, we train a *Long Short-term Memory (LSTM)* network, which has three LSTM layers (each layer has 5 hidden states) followed by one fully connected layer. The input of our LSTM network in each time slot is the current channel state and all previous outputs, and the output is the transmission probability in the current time slot. In the training process, for the first two synthetic datasets, we manually generate 100 sequences (each sequence has 5000 time slots) as the training datasets; for the real trace dataset, we use the first 5100 time slots and split them into 100 sequences by the sliding window technique, and use those sequences as our training datasets. The optimal offline transmission decisions of the training datasets can be obtained via dynamic programming. We use the mean absolute error as the loss function and employ the Adam optimizer to train the weights. In the end, to convert the output of our LSTM network (i.e., transmission probability) to the exact transmission decisions, we take the accumulated transmission probability since the last transmission as the transmission decision at the current slot (i.e., when the accumulated transmission probability is no smaller than 1, we will make a transmission at the next ON slot). This conversion is widly used in the literature [6], [21].

To generate predictions with varying qualities, we first train the ML models without any perturbation. Then during the test, we perturb the input of the network by flipping the real channel state (i.e., flip ON channel to OFF channel or flip OFF channel to ON channel) with some probability $p \in [0, 1]$. This represents the scenario where the prediction is biased due to the distribution shift between training and testing data. Similar techniques are commonly adapted in the literature [19], [20]

***Performance Evaluation.*** We evaluate four algorithms: (i) *PD_[6]*, the primal-dual-based online algorithm proposed in [6], (ii) *PDOA*, our proposed primal-dual-based online algorithm, (iii) *Pure_ML*, the algorithm that purely follows the ML prediction, and (iv) our learning-augmented online algorithm LAPDOA with different trust parameters (i.e., $\lambda \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$). Recall that a smaller $\lambda$ indicates higher confidence in the prediction. In Fig. 2, we show the average cost ratio (the ratio between the cost of online algorithm and the cost of the optimal offline algorithm) of these algorithms. First, since our online algorithm PDOA does not rely on the predictions, its performance is robust over all prediction perturbations and greatly outperforms the theoretical analysis (with a CR of 3). Second, when the prediction perturbation is small (i.e., the prediction is accurate), our learning-augmented algorithm that trusts the prediction (i.e., $\lambda \in \{0.1, 0.3\}$) perform closely to the offline optimum, achieving consistency. Third, when the prediction perturbation is large (i.e., the prediction is inaccurate), our learning-augmented algorithm can still have a good performance guarantee by choosing a small trust parameter (i.e., $\lambda \in \{0.7, 0.9\}$), achieving robustness.

## VII. Conclusion

In this paper, we studied the minimization of data freshness and transmission costs under a time-varying wireless channel. After reformulating our original problem to a TCP ACK problem, we developed a 3-competitive primal-dual-based online algorithm. Realizing the pros and cons of online algorithms and ML algorithms, we designed a learning-augmented online algorithm that takes advantage of both approaches and achieves consistency and robustness. Finally, simulation results validate the superiority of our online algorithm and highlight the consistency and robustness achieved by our learning-augmented algorithm. For future work, one interesting direction would be to consider how to adaptively select the trust parameter $\lambda$ to achieve the best performance.

## References

[1] S. Li, L. D. Xu, and S. Zhao, "The internet of things: a survey," *Information systems frontiers*, vol. 17, pp. 243–259, 2015.

[2] F. Wu, C. Rüdiger, and M. R. Yuce, "Real-time performance of a self-powered environmental iot sensor network system," *Sensors*, vol. 17, no. 2, p. 282, 2017.

[3] X. Cao, J. Wang, Y. Cheng, and J. Jin, "Optimal sleep scheduling for energy-efficient aoi optimization in industrial internet of things," *IEEE Internet of Things Journal*, vol. 10, no. 11, pp. 9662–9674, 2023.

[4] B. Yu, Y. Cai, X. Diao, and K. Cheng, "Adaptive packet length adjustment for minimizing age of information over fading channels," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2023.

[5] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 2731–2735.

[6] Y.-H. Tseng and Y.-P. Hsu, "Online energy-efficient scheduling for timely information downloads in mobile networks," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 1022–1026.

[7] A. R. Karlin, C. Kenyon, and D. Randall, "Dynamic tcp acknowledgement and other stories about e/(e-1)," in *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, ser. STOC '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 502–509.

[8] Y. Sun, I. Kadota, R. Talak, and E. Modiano, *Age of information: A new metric for information freshness*. Springer Nature, 2022.

[9] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.

[10] E. Fountoulakis, N. Pappas, M. Codreanu, and A. Ephremides, "Optimal sampling cost in wireless networks with age of information constraints," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 918–923.

[11] Z. Liu, B. Li, Z. Zheng, Y. T. Hou, and B. Ji, "Towards optimal tradeoff between data freshness and update cost in information-update systems," *IEEE Internet of Things Journal*, pp. 1–1, 2023.

[12] K. Saurav and R. Vaze, "Minimizing the sum of age of information and transmission cost under stochastic arrival model," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[13] A. M. Bedewy, Y. Sun, S. Kompella, and N. B. Shroff, "Optimal sampling and scheduling for timely status updates in multi-source networks," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 4019–4034, 2021.

[14] A. Sinha and R. Bhattacharjee, "Optimizing age-of-information in adversarial and stochastic environments," *IEEE Transactions on Information Theory*, vol. 68, no. 10, pp. 6860–6880, 2022.

[15] S. Banerjee and S. Ulukus, "Age of information in the presence of an adversary," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–8.

[16] S. Li, C. Li, Y. Huang, B. A. Jalaian, Y. T. Hou, and W. Lou, "Enhancing resilience in mobile edge computing under processing uncertainty," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 3, pp. 659–674, 2023.

[17] T. Lykouris and S. Vassilvitskii, "Competitive caching with machine learned advice," *J. ACM*, vol. 68, no. 4, jul 2021.

[18] M. Purohit, Z. Svitkina, and R. Kumar, "Improving online algorithms via ml predictions," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[19] E. Bamas, A. Maggiori, and O. Svensson, "The primal-dual method for learning augmented algorithms," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 20 083–20 094.

[20] D. Rutten, N. Christianson, D. Mukherjee, and A. Wierman, "Smoothed online optimization with unreliable predictions," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 7, no. 1, mar 2023. [Online]. Available: https://doi.org/10.1145/3579442

[21] N. Buchbinder, J. S. Naor *et al.*, "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.

[22] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[23] I. K. Jain, R. Subbaraman, T. H. Sadarahalli, X. Shao, H.-W. Lin, and D. Bharadia, "Mmobile: Building a mmwave testbed to evaluate and address mobility effects," in *Proceedings of the 4th ACM Workshop on Millimeter-Wave Networks and Sensing Systems*, ser. mmNets'20. New York, NY, USA: Association for Computing Machinery, 2020.

## Appendix

### A. Proof of Lemma 1

*Proof.* We prove that problem (3) is equivalent to problem (5) by showing that any optimal solution to problem (3) can be converted to an optimal solution to problem (5), and vice versa [22, Sec. 4.1.3].

We first show that any feasible solution to problem (3) can be converted to a feasible solution to problem (5). Given a channel state pattern $\mathbf{s}$, we assume that the solution $\pi = \{t_1, t_2, \ldots, t_n\}$ is a feasible solution to problem (3), where this solution makes the $i$-th transmission at the ON slot $t_i$ (i.e., $s(t_i)d(t_i) = 1$), and the total number of transmission is $n$. We can compute the total cost of the solution $\pi$ to problem (3) as $C_3(s, \pi) = cn + (t_1 - 1)t_1/2 + \sum_{i=2}^{n}(t_i - t_{i-1} - 1)(t_i - t_{i-1})/2 + (T - t_n - 1)(T - t_n)/2$, where $cn$ is the total transmission cost and the rest is the total staleness cost. Based on the solution $\pi$ to problem (3), we construct a solution $\pi'$ to problem (5) in the following way: (i) solution $\pi'$ sends the ACKs at the same time when solution $\pi$ transmits, i.e., solution $\pi'$ sends ACK at a sequence of slot $\{t_1, t_2, \ldots, t_n\}$; (ii) based on the ACK decisions in step (i), for any slot $t$ and packet $i \le t$, solution $\pi'$ lets $z_i(t) = 0$ if $\sum_{\tau=i}^{t} s(\tau)d(\tau) \ge 1$ and $z_i(t) = 1$ if $\sum_{\tau=i}^{t} s(\tau)d(\tau) = 0$. We denote the solution $\pi'$ to problem (5) by $\pi' = \{\{t_1, t_2, \ldots, t_n\}, \{\{z_i(t)\}_{i=1}^{t}\}_{t=1}^{T}\}$. We can easily verify that solution $\pi'$ is a feasible solution to problem (5) because both constraints (5b) and (5c) are satisfied. Furthermore, according to the construction of $z_i(t)$ in step (ii), we know that once the ACK is made at some ON slot $t \in \{t_1, t_2, \ldots, t_n\}$, then all arrived packets (packet 1 to packet $t$) are acked forever after slot $t$, i.e., $z_i(\tau) = 0$ for all $i \le t$ and all $\tau \ge t$. This indicates that we can compute the total holding cost of the solution $\pi'$ as

$$\begin{aligned}
&\sum_{t=1}^{T}\sum_{i=1}^{t} z_i(t)\\
=&\underbrace{\sum_{t=1}^{t_1-1}\sum_{i=1}^{t} z_i(t) + \sum_{t=t_1}^{T}\sum_{i=1}^{t_1} z_i(t)}_{\text{Total holding cost of packet 1 to packet } t_1}\\
&+\underbrace{\sum_{t=t_1+1}^{t_2-1}\sum_{i=t_1+1}^{t} z_i(t) + \sum_{t=t_2}^{T}\sum_{i=t_1+1}^{t_2} z_i(t)}_{\text{Total holding cost of packet } (t_1+1) \text{ to packet } t_2}\\
&+\cdots+\underbrace{\sum_{t=t_n+1}^{T}\sum_{i=t_n+1}^{t} z_i(t)}_{\text{Total holding cost of packet } (t_n+1) \text{ to packet } T}\\
\overset{(a)}{=}&\underbrace{\sum_{t=1}^{t_1-1}\sum_{i=1}^{t} 1 + \sum_{t=t_1}^{T}\sum_{i=1}^{t_1} 0}_{\text{Total holding cost of packet 1 to packet } t_1}\\
&+\underbrace{\sum_{t=t_1+1}^{t_2-1}\sum_{i=t_1+1}^{t} 1 + \sum_{t=t_2}^{T}\sum_{i=t_1+1}^{t_2} 0}_{\text{Total holding cost of packet } (t_1+1) \text{ to packet } t_2}\\
&+\cdots+\underbrace{\sum_{t=t_n+1}^{T}\sum_{i=t_n+1}^{t} 1}_{\text{Total holding cost of packet } (t_n+1) \text{ to packet } T}\\
=&(t_1-1)t_1/2 + \sum_{i=2}^{n}(t_i - t_{i-1} - 1)(t_i - t_{i-1})/2\\
&+(T - t_n - 1)(T - t_n)/2,
\end{aligned} \tag{14}$$

where in (a), $z_i(t) = 1$ is because packet $i$ is not acked by slot $t$ and $z_i(t) = 0$ otherwise. In addition, the total ACK cost of the solution $\pi'$ is $cn$. Therefore, the total cost of the solution $\pi'$ to problem (5) is $C_5(s, \pi') = cn + (t_1 - 1)t_1/2 + \sum_{i=2}^{n}(t_i - t_{i-1} - 1)(t_i - t_{i-1})/2 + (T - t_n - 1)(T - t_n)/2$, which is the same as the total cost of solution $\pi$ to problem (3). In conclusion, any feasible solution $\pi$ to problem (3) can be converted to a feasible solution $\pi'$ to problem (5), and those two solutions have the same total cost, i.e., $C_3(s, \pi) = C_5(s, \pi')$.

Next, we show that any feasible solution to problem (5) can be converted to a feasible solution to problem (3). Given a channel state pattern $\mathbf{s}$, we assume that the solution $\pi = \{\{t_1, t_2, \ldots, t_n\}, \{\{z_i(t)\}_{i=1}^{t}\}_{t=1}^{T}\}$ is a feasible solution to problem (5), where this solution makes the $i$-th ACK at the ON slot $t_i$ (i.e., $s(t_i)d(t_i) = 1$). Though the solution $\pi$ is a feasible solution to problem (5), it is possible that this solution makes unnecessary cost of $z_i(t)$ (i.e., letting $z_i(t) = 1$ even though $\sum_{\tau=i}^{t} s(\tau)d(\tau) \ge 1$). In this case, we can always find another feasible solution $\hat{\pi} = \{\{t_1, t_2, \ldots, t_n\}, \{\{\hat{z}_i(t)\}_{i=1}^{t}\}_{t=1}^{T}\}$ to problem (5) that makes the same ACK decisions as the solution $\pi$ but never makes the unnecessary cost of $\hat{z}_i(t)$ (i.e., letting $\hat{z}_i(t) = 1$ only when $\sum_{\tau=i}^{t} s(\tau)d(\tau) = 0$ and letting $\hat{z}_i(t) = 0$ only when $\sum_{\tau=i}^{t} s(\tau)d(\tau) \ge 1$), and their total cost in problem (5) satisfies $C_5(\mathbf{s}, \hat{\pi}) \le C_5(\mathbf{s}, \pi)$. Similar to the previous analysis (i.e., Eq. (14)), the total cost of the solution $\hat{\pi}$ is $C_5(s, \hat{\pi}) = cn + (t_1 - 1)t_1/2 + \sum_{i=2}^{n}(t_i - t_{i-1} - 1)(t_i - t_{i-1})/2 + (T - t_n - 1)(T - t_n)/2$. Based on the feasible solution $\hat{\pi}$ to problem (5), we can construct a solution $\pi'$ to problem (3) in the following way:

(i) solution $\pi'$ transmits at the same time when solution $\pi$ sends the ACKs, i.e., solution $\pi'$ transmits at a sequence of slot $\{t_1, t_2, \dots, t_n\}$. We denote the solution $\pi'$ to problem (3) by $\pi' = \{t_1, t_2, \dots, t_n\}$. We can easily check the solution $\pi'$ is a feasible solution to problem (3) since constraint (3b) is satisfied. In addition, we can compute the total cost of the solution $\pi'$ to problem (3) as $C_3(s, \pi') = cn + (t_1 - 1)t_1/2 + \sum_{i=2}^{n} (t_i - t_{i-1} - 1)(t_i - t_{i-1})/2 + (T - t_n - 1)(T - t_n)/2$, which is the same as the total cost of solution $\hat{\pi}$ to problem (5). In conclusion, any feasible solution $\pi$ to problem (5) can be converted to a feasible solution $\pi'$ to problem (3), and their total cost satisfies $C_5(s, \pi) \geq C_3(s, \pi')$.

Finally, we show that for any optimal solution of problem (3), it can be converted to an optimal solution to problem (5), and vice versa. Assuming that the solution $\pi_*$ is an optimal solution to problem (3). From the above analysis, we can construct a feasible solution $\pi'_*$ to problem (5), and their total cost satisfies $C_3(\mathbf{s}, \pi_*) = C_5(\mathbf{s}, \pi'_*)$. We claim that $\pi'_*$ is also an optimal solution to problem (5). Otherwise, there must be an optimal solution $\pi''_*$ to problem (5) such that $\pi''_* \neq \pi'_*$ and $C_5(\mathbf{s}, \pi'_*) > C_5(\mathbf{s}, \pi''_*)$. Again, from the previous analysis, we know that the optimal solution $\pi''_*$ to problem (5) can be converted to a feasible solution $\pi^\dagger_*$ to problem (3), and their total cost satisfies $C_5(\mathbf{s}, \pi''_*) \geq C_3(\mathbf{s}, \pi^\dagger_*)$. However, this indicates the solution $\pi_*$ is not an optimal solution to problem (3) since $C_3(\mathbf{s}, \pi^\dagger_*) < C_3(\mathbf{s}, \pi_*)$, which contradicts with our assumption. Therefore, the solution $\pi_*$ is also an optimal solution to problem (5). Similarly, we can show that any optimal solution to problem (5) is also an optimal solution to problem (3). This completes the proof. $\square$

### *B. Proof of Theorem 1*

We first demonstrate that PDOA produces a feasible solution to primal program (6) and dual program (7) in Lemma 4. Then, we explain the usefulness of the primal-dual program [21] for competitive analysis. Finally, we establish that our online primal-dual-based PDOA is 3-competitive.

To begin with, we first introduce two key observations of PDOA that will be widely used in the proofs.

*Observation* 1. *Assuming that PDOA made the latest ACK at some ON slot L and the current time slot is t (t > L), then at slot t, before the threshold is achieved (*$M < 1$*), PDOA updates the primal variables and the dual variables of packet* $(L + 1)$ *to packet t (i.e., letting* $z_i(t) = 1$ *and* $y_i(t) = 1$ *(or* $c - c \cdot M$*) where* $i \in [L + 1, t]$*); however, once the threshold is achieved (*$M \geq 1$*), then PDOA only updates the primal variables of the unacked packets due to the OFF slot (i.e., letting* $z_i(t) = 1$ *where* $i \in [L + 1, t]$ *and* $s(t) = 0$*).*

*Observation* 2. *Once PDOA makes an ACK at some ON slot t, all the packets arriving no later than slot t (packet* 1 *to packet t) are acked forever after slot t, and their primal variables and dual variables will never be changed after slot t, i.e.,* $z_i(\tau) = 0$ *and* $y_i(\tau) = 0$ *for all* $i \leq t$ *and all* $\tau \geq t$.

With Observations 1 and 2, we ready to show the feasibility of the solution produced by PDOA.

*Lemma* 4. *PDOA produces a feasible solution to both primal program* (6) *and dual program* (7)*.*

*Proof.* The primal constraint (6c) and the dual constraint (7c) are clearly satisfied. For the primal constraint (6b), it is easy to verify that for the $i$-th packet at slot $t$ ($i \leq t$), if PDOA made an ACK during $[i, t]$, then constraint (6b) is satisfied; otherwise, since the $i$-th packet is not acked by slot $t$, PDOA will update $z_i(t)$ to be 1, so constraint (6b) is also satisfied. When the channels are OFF, the dual constraints (7b) are automatically satisfied. Now, consider an ON slot $t$ and its dual constraint (7b) $\sum_{i=1}^{t} \sum_{\tau=t}^{T} y_i(\tau) \leq c$. This constraint requires that for all the packets arriving no later than slot $t$ (packet 1 to packet $t$), the sum of their dual variables beyond slot $t$ should not exceed $c$. Assuming that this ON slot $t$ falls into the $k$-th ACK interval $[t_k + 1, t_{k+1}]$ of PDOA, i.e., $t_k + 1 \leq t \leq t_{k+1}$, where PDOA makes two ACKs at the ON slot $t_{k+1}$ ($t_{k+1} > t_k + 1$) and the ON slot $t_k$ (when the ON slot $t$ falls into the last ACK interval $[t_K + 1, T]$, where PDOA makes the last ACK at the ON slot $t_K$, our following analysis can be easily extended to this case). According to Observations 1 and 2, packet 1 to packet $t_k$ are not updated after slot $t_k$, and packet $(t_k + 1)$ to packet $t$ are not updated after slot $t_{k+1}$, then we have

$$\begin{aligned}
&\sum_{i=1}^{t} \sum_{\tau=t}^{T} y_i(\tau) \\
=&\sum_{i=1}^{t_k} \sum_{\tau=t}^{T} y_i(\tau) + \sum_{i=t_k+1}^{t} \sum_{\tau=t}^{t_{k+1}} y_i(\tau) \\
&+ \sum_{i=t_k+1}^{t} \sum_{\tau=t_{k+1}+1}^{T} y_i(\tau) \\
=&0 + \sum_{i=t_k+1}^{t} \sum_{\tau=t}^{t_{k+1}} y_i(\tau) + 0 \\
=&\sum_{i=t_k+1}^{t} \sum_{\tau=t}^{t_{k+1}} y_i(\tau).
\end{aligned} \tag{15}$$

Next, we discuss the value of $\sum_{i=t_k+1}^{t} \sum_{\tau=t}^{t_{k+1}} y_i(\tau)$ in two cases: (i) $t = t_k + 1$, and (ii) $t_k + 1 < t \leq t_{k+1}$.

(i) $t = t_k + 1$. In this case, we have $\sum_{i=t_k+1}^{t} \sum_{\tau=t}^{t_{k+1}} y_i(\tau) = \sum_{\tau=t_k+1}^{t_{k+1}} y_{t_k+1}(\tau)$. We assume that the threshold is achieved after the updating packet $j$ ($t_k + 1 \leq j \leq t_{k+1}$) at slot $t^\dagger$ ($t_k + 2 \leq t^\dagger \leq t_{k+1}$), i.e., $\sum_{\tau=t_k+1}^{t^\dagger-1} \sum_{i=t_k+1}^{\tau} y_i(\tau) + \sum_{i=t_k+1}^{j} y_i(t^\dagger) = c$. Then

$$\begin{aligned}
&\sum_{\tau=t_k+1}^{t_{k+1}} y_{t_k+1}(\tau) \\
=&\sum_{\tau=t_k+1}^{t^\dagger} y_{t_k+1}(\tau) + \sum_{\tau=t^\dagger+1}^{t_{k+1}} y_{t_k+1}(\tau) \\
\overset{(a)}{=}&\sum_{\tau=t_k+1}^{t^\dagger} y_{t_k+1}(\tau) + 0 \\
\leq&\sum_{\tau=t_k+1}^{t^\dagger-1} \sum_{i=t_k+1}^{\tau} y_i(\tau) + \sum_{i=t_k+1}^{j} y_i(t^\dagger) \\
=&c,
\end{aligned} \tag{16}$$

where $(a)$ is because the threshold is achieved at slot $t^\dagger$ and packet $(t_k + 1)$ is never updated after slot $t^\dagger$.

(ii) $t_k + 1 < t \leq t_{k+1}$. We assume that during the interval $[t, t_{k+1}]$, all packets arriving between $[t_k + 1, t]$ (packet

$t_k+1$ to packet $t$) make a total number $m$ updates, i.e., $\sum_{i=t_k+1}^{t}\sum_{\tau=t}^{t_{k+1}} y_i(\tau)=m$. If the ACK maker $M$ is no smaller than 1 before $m$ achieves $\lceil c\rceil-1$ (i.e., $m<\lceil c\rceil-1$), then all the dual variables of packet $t_k+1$ to packet $t$ are not updated according to Observation 1, and we have $\sum_{i=t_k+1}^{t}\sum_{\tau=t}^{t_{k+1}} y_i(\tau)=m<\lceil c\rceil-1<(c+1)-1=c$. Now consider the case where the ACK maker $M$ is smaller than 1 before $m$ achieves $\lceil c\rceil-1$. When $m=\lceil c\rceil-1$, the increment of the ACK maker $M$ due to those $m$ updates (denoted by $M'$) is $M'=(\lceil c\rceil-1)/c$. At this point, the ACK maker $M$ becomes $M=N'+M'\geq 1/c+M'=\lceil c\rceil/c\geq 1$, where $N'$ is the increment of the ACK maker $M$ due to packet $(t_k+1)$ to packet $t-1$ ($N'$ is at least $1/c$ since packet $(t_k+1)$ is not acked at slot $(t_k+1)$, which increases $N'$ by $1/c$). Given that the ACK maker $M$ now is no smaller than 1, all the dual variables of packet $t_k+1$ to packet $t$ are not updated according to Observation 1. Therefore, we have $\sum_{i=t_k+1}^{t}\sum_{\tau=t}^{t_{k+1}} y_i(\tau)=m=\lceil c\rceil-1<(c+1)-1=c$.
In summary, we have $\sum_{i=1}^{t}\sum_{\tau=t}^{T} y_i(\tau)\leq c$ in both cases, thus the dual constraint (7b) is satisfied. □

The primal-dual program allows us to analyze the CR of our online algorithm without knowing the optimal offline solution. As shown in Lemma 4, for a given channel state $\mathbf{s}$, our online algorithm outputs an integer feasible solution (denoted by $\pi$) to both primal program (6) and dual program (7). We use $P(\mathbf{s},\pi)$ and $D(\mathbf{s},\pi)$ to denote the primal objective value and the dual objective value under $\pi$, respectively. In addition, the integer solution $\pi$ is also a feasible solution to problem (5), and we use $C_5(\mathbf{s},\pi)$ to denote the objective value of problem (5) under $\pi$. Because primal program (6) and problem (5) have the same objective function, we have $C_5(\mathbf{s},\pi)=P(\mathbf{s},\pi)$. The CR of our online algorithm $\pi$ for primal program (6) satisfies

$$\underbrace{\frac{C_5(\mathbf{s},\pi)}{OPT_5(\mathbf{s})}}_{\text{CR of problem (5)}}\leq\underbrace{\frac{P(\mathbf{s},\pi)}{OPT_6(\mathbf{s})}}_{\text{CR of primal program (6)}}\leq\frac{P(\mathbf{s},\pi)}{D(\mathbf{s},\pi)},\tag{17}$$

where $OPT_5(\mathbf{s})$ and $OPT_6(\mathbf{s})$ is the cost of the optimal offline algorithm for problem (5) and primal problem (6), respectively. Here we have $OPT_5(\mathbf{s})\geq OPT_6(\mathbf{s})$ because the search space of the optimal solution in primal problem (6) is larger than that in problem (5). The second inequality comes from the weak duality [21]. Furthermore, if we can show that there exists a constant $\beta$ such that $P(\mathbf{s},\pi)/D(\mathbf{s},\pi)\leq\beta$ holds for any channel state $\mathbf{s}$, then our online algorithm is $\beta$-competitive for primal program (6) and problem (5).

For notational simplicity, let $P$ and $D$ be the value of the objective function of the primal and the dual solutions produced by PDOA under a given channel state $\mathbf{s}$, respectively. In the following, we show that $P/D\leq 3$. We assume that PDOA makes a sequence of ACKs $\pi=\{t_1,t_2,\ldots,t_K\}$, where PDOA makes the $i$-th ACK at the ON slot $t_i$ (i.e., $s(t_i)d(t_i)=1$). Our goal is to show that for any $k$-th ($k\in[0,K]$) ACK interval $[t_k+1,t_{k+1}]$ (where the first ACK interval is $[1,t_1]$ when $k=0$ and the last ACK interval is $[t_K+1,T]$ when $k=K$), the ratio between the primal objective value and the dual objective value in this $k$-th ACK interval (denoted by $P(k)$ and $D(k)$, respectively) is at most 3, i.e., $P(k)/D(k)\leq 3$. According to Observation 2, $P(k)$ and $D(k)$ are never changed when this ACK interval ends at slot $t_{k+1}$. This implies that PDOA also achieves $P/D\leq 3$ on the entire instance $\mathbf{s}$.

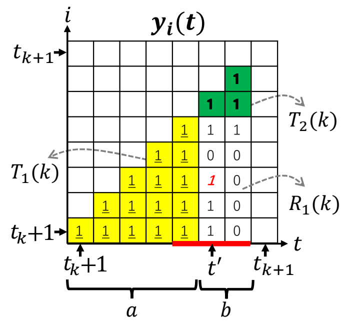


Fig. 3. An illustration of $T_1(k)$, $T_2(k)$, and $R_1(k)$. $T_1(k)$ is an equilateral triangle made of 1 (the underlined 1's with yellow background), $T_2(k)$ is an equilateral triangle made of 1 (the bold 1's with green background), and $R_1(k)$ is a rectangle made of 1 and 0 (the regular 1's and 0's without background).

We first discuss the relation between $P(k)$ and $D(k)$ in the first $K$ ACK interval $[t_k+1,t_{k+1}]$ (i.e., $k\in[0,K-1]$, where there is always an ACK made at slot $t_{k+1}$), and then discuss the relation between $P(K)$ and $D(K)$ in the last ACK interval $[t_K+1,T]$ (i.e., $k=K$, where it is possible that no ACK is made at slot $T$) in the end.

Consider any $k$-th ($k\in[0,K-1]$) ACK interval $[t_k+1,t_{k+1}]$ (denoted by $I_k$), where PDOA makes two ACKs at the ON slots $t_k$ and $t_{k+1}$, respectively. There are two cases when making an ACK at $t_{k+1}$: 1) the ACK marker $M$ equals or is larger than 1 at $t_{k+1}$; 2) the ACK marker $M$ equals or is larger than 1 at some OFF slot $t'$ ($t'<t_{k+1}$) and $t_{k+1}$ is the very first ON slot after $t'$ (the channels are OFF during $[t',t_{k+1}-1]$). An illustration of Case 2 is provided in Fig. 1. We emphasize that according to Observation 2, we only need to consider the primal variable update and dual variable updates of packet $(t_k+1)$ to packet $t_{k+1}$, since all previous packets (packet 1 to packet $t_k$) are never updated after slot $t_k$.

Case 1): The ACK marker $M$ equals or is larger than 1 at $t_{k+1}$. In this case, Lines 3-7 in PDOA are repeated $c$ times, and the total holding cost in $I_k$ is $\sum_{t=t_k+1}^{t_{k+1}}\sum_{i=t_k+1}^{t} z_i(t)=\lceil c\rceil$ and the total ACK cost in $I_k$ is $\sum_{t=t_k+1}^{t_{k+1}} c\cdot d(t)=c\cdot d(t_{k+1})=c$. Thus, the primal objective is $P(k)=\lceil c\rceil+c$. Similarly, the dual objective is the sum of the dual variables $y_i(t)$ in $I_k$, which is $D(k)=\sum_{t=t_k+1}^{t_{k+1}}\sum_{i=t_k+1}^{t} y_i(t)=c$. Therefore, we have $P(k)/D(k)=(\lceil c\rceil+c)/c\leq(c+1+c)/c<3$.

Case 2): The ACK marker $M$ equals or is larger than 1 at some OFF slot $t'$ ($t'<t_{k+1}$) and $t_{k+1}$ is the very first ON slot after $t'$. We use $C_A(k)$ to denote the total ACK cost and use $C_H(k)$ to denote the total holding cost in $I_k$. Here

$P(k) = C_A(k) + C_H(k)$. We have

$$\begin{aligned} P(k)/D(k) &= (C_A(k) + C_H(k))/D(k) \\ &\overset{(a)}{=} c/D(k) + C_H(k)/D(k) \\ &\overset{(b)}{\leq} c/c + C_H(k)/D(k) \\ &\overset{(c)}{\leq} c/c + 2\Delta D(k)/D(k) \\ &= 3, \end{aligned} \tag{18}$$

where $(a)$ is because PDOA makes only one ACK at slot $t_{k+1}$ during $I_k$, i.e., $C_A(k) = \sum_{t=t_k+1}^{t_{k+1}} c \cdot d(t) = c \cdot d(t_{k+1}) = c$; $(b)$ is due to the dual objective value $D(k)$ is at least $c$ (i.e., when the ACK markter $M$ equals or is larger than 1, $D(k)$ equals $c$, and $D(k)$ can be larger than $c$ due to the additional updates of the dual variables in Lines 19-25); and we prove $(c)$ as follows. First, we can compute the total holding cost in $I_k$ as

$$\begin{aligned} &C_H(k) \\ =&\sum_{\tau=t_k+1}^{t_{k+1}} \sum_{i=t_k+1}^{\tau} z_i(\tau) \\ =&\sum_{\tau=t_k+1}^{t_{k+1}-1} \sum_{i=t_k+1}^{\tau} z_i(\tau) + \sum_{i=t_k+1}^{t_{k+1}} z_i(t_{k+1}) \\ \overset{(d)}{=}&\sum_{\tau=t_k+1}^{t_{k+1}-1} \sum_{i=t_k+1}^{\tau} z_i(\tau) + 0 \\ =&\sum_{\tau=t_k+1}^{t_{k+1}-1} \sum_{i=t_k+1}^{\tau} z_i(\tau) \\ \overset{(e)}{=}&\sum_{\tau=t_k+1}^{t_{k+1}-1} \sum_{i=t_k+1}^{\tau} 1 \\ =&(t_{k+1} - t_k - 1)(t_{k+1} - t_k)/2, \end{aligned} \tag{19}$$

where in $(d)$, $z_i(t_{k+1}) = 0$ for any $i \in [t_k + 1, t_{k+1}]$ is because all the packets in $I_k$ are acked at slot $t_{k+1}$; and in $(e)$, $z_i(\tau) = 1$ for any $\tau \in [t_k+1, t_{k+1}-1]$ and $i \in [t_k+1, \tau]$ is because the packets in $I_k$ are not acked until slot $t_{k+1}$, and each of them needs to pay a holding cost, i.e., $z_i(\tau) = 1$. Similarly, the dual objective value in $I_k$ can be computed as $D(k) = \sum_{\tau=t_k+1}^{t_{k+1}-1} \sum_{i=t_k+1}^{\tau} y_i(\tau)$. We split $D(k)$ into three parts: the triangle $T_1(k) = \sum_{\tau=t_k+1}^{t'-1} \sum_{i=t_k+1}^{\tau} y_i(\tau)$, the triangle $T_2(k) = \sum_{\tau=t'}^{t_{k+1}-1} \sum_{i=t'}^{\tau} y_i(\tau)$, and the rectangle $R_1(k) = \sum_{\tau=t'}^{t_{k+1}-1} \sum_{i=t_k+1}^{t'-1} y_i(\tau)$ (see an illustration in Fig. 3). Here $D(k) = T_1(k) + T_2(k) + R_1(k)$.

Our goal is to show that $C_H(k) \leq 2(T_1(k) + T_2(k)) \leq 2D(k)$. Here, we can compute $T_1(k) = \sum_{\tau=t_k+1}^{t'-1} \sum_{i=t_k+1}^{\tau} y_i(\tau) = \sum_{\tau=t_k+1}^{t'-1} \sum_{i=t_k+1}^{\tau} 1 = (t' - t_k - 1)(t' - t_k)/2$, where $y_i(\tau) = 1$ for any $\tau \in [t_k + 1, t' - 1]$ and $i \in [t_k + 1, \tau]$ comes from Lines 3-7 in PDOA since the ACK marker $M$ equals or is larger than 1 until $t'$. In addition, we can compute $T_2(k) = \sum_{\tau=t'}^{t_{k+1}-1} \sum_{i=t'}^{\tau} y_i(\tau) = \sum_{\tau=t'}^{t_{k+1}-1} \sum_{i=t'}^{\tau} 1 = (t_{k+1} - t')(t_{k+1} - t' + 1)/2$, where $y_i(\tau) = 1$ for any $\tau \in [t', t_{k+1} - 1]$ and $i \in [t', \tau]$ comes from Lines 19-25 in PDOA since the channels are OFF during $[t', t_{k+1} - 1]$. Let $a$ be the length of $[t_k + 1, t' - 1]$ (i.e., $a = t' - t_k - 1$) and $b$ be the length of $[t', t_{k+1} - 1]$ (i.e., $b = t_{k+1} - t'$). Now we have $T_1(k) = a(a + 1)/2$, $T_2(k) = b(b + 1)/2$, and $C_H(k) = (a + b)(a + b + 1)/2$. Clearly, we have

$$\begin{aligned} &2D(k) - C_H(k) \\ &= 2(T_1(k) + T_2(k) + R_1(k)) - C_H(k) \\ &\geq 2(T_1(k) + T_2(k)) - C_H(k) \\ &= 2 \cdot [a(a + 1)/2 + b(b + 1)/2] - (a + b)(a + b + 1)/2 \\ &= [(a - b)^2 + a + b]/2 \\ &\geq 0, \end{aligned} \tag{20}$$

which completes $(c)$ in Eq. (18).

In the end, we consider the last time interval $[t_K + 1, T]$ (denoted by $I_K$). If the last slot is an ON slot and PDOA makes the last ACK exactly at the last slot, i.e., $t_K = T$, then our previous analysis in Cases 1 and 2 still holds. Next, we consider the scenario where $t_K < T$. There are two cases at slot $T$: 1) $T$ is the slot before the ACK marker $M$ equals or is larger than 1; 2) $T$ is the slot when or after the ACK marker $M$ equals or is larger than 1. In both cases, there is no ACK made during $I_K$.

Case 1): $T$ is the slot before the ACK marker $M$ equals or is larger than 1. In this case, the total holding cost and the total ACK cost in $I_K$ are $(T-t_K)(T-t_K+1)/2$ and 0, respectively. Thus, the primal objective is $P(K) = (T - t_K)(T - t_K + 1)/2 + 0 \cdot c = (T - t_K)(T - t_K + 1)/2$. The dual objective is the sum of total dual variables $y_i(t)$ in $I_{K+1}$, which is $(T-t_K)(T-t_K+1)/2$, i.e., $D(K) = (T-t_K)(T-t_K+1)/2$. Therefore, we have $P(K)/D(K) = 1$.

Case 2): $T$ is the slot when or after the ACK marker $M$ equals or is larger than 1. We assume that the ACK marker $M$ equals or is larger than 1 at some slot $t'$ ($t_K < t' \leq T$). We claim that the channels are OFF during the interval $[t', T]$. Otherwise, an ACK will be made during $[t', T]$, which contradicts the fact that there is no ACK made during $I_K$. We use $C_A(K)$ to denote the total ACK cost and use $C_H(K)$ to denote the total holding cost. Here $P(K) = C_A(K) + C_H(K) = 0 + C_H(K)$, where $C_A(K) = 0$ because there is no ACK made during $I_K$. We have

$$\begin{aligned} P(K)/D(K) &= (C_A(K) + C_H(K))/D(K) \\ &= 0 + C_H(K)/D(K) \\ &\overset{(a)}{\leq} 2\Delta D(K)/D(K) \\ &= 2, \end{aligned} \tag{21}$$

where the analysis in $(a)$ is the same as that for $(c)$ in Eq. (18).

In summary, given any channel state $\mathbf{s}$, PDOA achieves $P(k)/D(k) \leq 3$ for any ACK interval $I_k$ ($k \in [0, K]$), and thus PDOA achieves $P/D \leq 3$ on the entire instance $\mathbf{s}$. By the weak duality, PDOA is 3-competitive.

## C. *Proof of Lemma 2*

The proof outline is as follows: We first show that LAPDOA produces a feasible primal solution and an almost feasible dual solution (with a factor of $c/(c + 1)$) in Lemma 5. Then, we show that in any $k$-th ACK interval, the ratio between the primal objective and the dual objective in this $k$-th ACK interval (denoted by $P(k)$ and $D(k)$, respectively) is at most $3/\lambda$, i.e., $P(k)/D(k) \leq 3/\lambda$. This implies that the ratio between the total primal objective (denoted by $P$) and the total dual objective (denoted by $D$) is also at most $3/\lambda$, i.e., $P/D \leq 3/\lambda$. Scaling down all dual variables $y_i(t)$ generated by LAPDOA by a multiplicative factor of $c/(c+1)$, we obtain a feasible dual solution with a dual objective value of $(c/(c+1)) \cdot D$. By the weak duality, we have $P/OPT \leq P/((c/(c+1)) \cdot D) = ((c+1)/c) \cdot P/D \leq ((c+1)/c) \cdot 3/\lambda$, which completes the proof.

To begin with, we introduce two key observations of LAPDOA that will be widely used in the following proofs.

*Observation* 3. *Assuming that LAPDOA made the latest ACK at some ON slot $L$ and the current time slot is $t$ ($t > L$), then at slot $t$, before the threshold is achieved ($M < 1$), LAPDOA updates the primal variables and the dual variables of packet $(L+1)$ to packet $t$ (i.e., for any $i \in [L+1, t]$, letting $z_i(t) = 1$ and $y_i(t) = 1$ if there is a big update, or letting $z_i(t) = 1$ and $y_i(t) = \lambda$ if there is a small update); however, once the threshold is achieved ($M \geq 1$), then LAPDOA only updates the primal variables of the unacked packets due to the OFF slot (i.e., letting $z_i(t) = 1$ where $i \in [L+1, t]$ and $s(t) = 0$).*

*Observation* 4. *Once LAPDOA makes an ACK at some ON slot $t$, all the packets arriving no later than slot $t$ (packet 1 to packet $t$) are acked forever after slot $t$, and their primal variables and dual variables will never be changed after slot $t$, i.e., $z_i(\tau) = 0$ and $y_i(\tau) = 0$ for all $i \leq t$ and all $\tau \geq t$.*

Then, we show that LAPDOA gives an almost feasible solution in Lemma 5.

*Lemma* 5. *LAPDOA produces a feasible primal solution. In addition, let $\mathbf{y} \triangleq \{\{y_i(t)\}_{i=1}^{t}\}_{t=1}^{T}$ be the dual solution produced by LAPDOA, then $(c/(c+1))\mathbf{y}$ is a feasible dual solution.*

*Proof.* We omit the proof for primal constraints (6b)-(6c) and dual constraint (7c) because they are similar to the proof in Lemma 4 and provide the proof for dual constraint (7b). Consider an ON slot $t$ and its dual constraint (7b) $\sum_{i=1}^{t} \sum_{\tau=t}^{T} y_i(\tau) \leq c$. Recall that this dual constraint requires that for all the packets arriving no later than slot $t$, the sum of their dual variables beyond slot $t$ should not exceed $c$. We assume that this ON slot $t$ falls into the $k$-th ACK interval $[t_k + 1, t_{k+1}]$, i.e., $t_k + 1 \leq t \leq t_{k+1}$, where LAPDOA makes two ACKs at the ON slots $t_k$ and $t_{k+1}$ (in a special case that the ON slot $t$ falls into the last interval $[t_K + 1, T]$, i.e., $t_K + 1 \leq t \leq T$, where LAPDOA makes the last ACK at the ON slot $t_K$, our following analysis can be extended to this case). According to Observations 3 and 4, packet 1 to packet $t_k$ are not updated after slot $t_k$, and packet $(t_k + 1)$ to packet $t$ are not updated after slot $t_{k+1}$, similar to Eq. (15), we have $\sum_{i=1}^{t} \sum_{\tau=t}^{T} y_i(\tau) = \sum_{i=t_k+1}^{t} \sum_{\tau=t}^{t_{k+1}} y_i(\tau)$. Furthermore, we assume that during the interval $[t, t_{k+1}]$, all packets arriving between $[t_k + 1, t]$ (packet $t_k + 1$ to packet $t$) make $m$ big updates and $n$ small updates (some zero updates can also be made but we ignore them since they cannot increase the dual variables). In other words, we have $\sum_{i=t_k+1}^{t} \sum_{\tau=t}^{t_{k+1}} y_i(\tau) = 1 \cdot m + \lambda \cdot n = m + \lambda n$. We claim that if $m + \lambda n \geq c$, then the increment of ACK maker $M$ due to those $m$ big and $n$ small updates (denoted by $M(m, n)$) will be larger than or equal to 1. This is true since we have $M(m, n) = m \cdot (1/\lambda c) + n \cdot (\lambda/c) = m/\lambda c + \lambda n/c \geq m/c + \lambda n/c = (m + \lambda n)/c \geq c/c = 1$. This claim, in turn, implies that $\sum_{i=t_k+1}^{t} \sum_{\tau=t}^{t_{k+1}} y_i(\tau) = m + \lambda n < c + 1$. To see this, consider the edge case where there are $m'$ big updates and $n'$ small updates made by all packets arriving between $[t_k + 1, t]$ since slot $t$, and they satisfy: (i) their sum of dual variables is smaller than $c$, i.e., $m' + \lambda n' < c$; (ii) with one more update (either big or small), the sum of their dual variables is no less than $c$, i.e., either $(m' + 1) + \lambda n' \geq c$ or $m' + \lambda(n' + 1) \geq c$ holds. From condition (ii) and our claim we know that with the arrival of one more update, the ACK maker $M$ will be larger than or equal to be 1 (because we have $M \geq M(m, n) \geq 1$) at some slot $t'$ ($t' \leq t_{k+1}$). When this happens, the sum of dual variables is at most $\max\{(m' + 1) + \lambda n', m' + \lambda(n' + 1)\} = (m' + 1) + \lambda n' = (m' + \lambda n') + 1 < c + 1$, and those dual variables will never be updated after slot $t'$ according to Observation 3. Therefore, we have $\sum_{i=1}^{t} \sum_{\tau=t}^{T} y_i(\tau) = \sum_{i=t_k+1}^{t} \sum_{\tau=t}^{t_{k+1}} y_i(\tau) < c + 1$. Now scaling down the dual solution $\mathbf{y}$ by a factor of $c/(c+1)$, we obtain a feasible dual solution $(c/(c+1))\mathbf{y}$. $\square$

In the following, we assume that $d(t)$ is updated to the ACK marker $M$ ($M \geq 1$) rather than 1 in Line 15, which possibly makes LAPDOA perform worse (i.e., has a larger total cost since the one-time ACK cost now is $c \cdot M$, which is larger than or equal to $c \cdot 1$). We show that our CR analysis holds for this worse setting (i.e., $d(t) = M$), and thus our CR analysis also holds for LAPDOA. The benefit of considering this worse setting is that this allows us to allocate the ACK costs to large and small updates. Specifically, under the worse setting, suppose that LAPDOA makes an ACK at slot $t_k$, and after $m$ ($m \geq 0$) big updates and $n$ ($n \geq 0$) small updates, LAPDOA is ready to make another ACK at some slot $t_{k+1}$. At this point, the ACK marker is $M = m/(\lambda c) + n\lambda/c$, and the ACK cost is $c \cdot M = m/\lambda + n\lambda$. However, instead of calculating the ACK cost at slot $t_{k+1}$, we can distribute the ACK cost to the updates in $[t_k + 1, t_{k+1}]$, that is, each big update gets an ACK cost of $1/\lambda$ and each small update gets an ACK cost of $\lambda$. The total ACK cost of those $m$ big updates and $n$ small updates is still $m/\lambda + n\lambda$. Doing this does not change the ACK cost, but now every big update or small update has a contribution to the ACK cost, which helps our analysis in the following when we compute the primal increment (i.e., the sum of ACK cost and holding cost) of each update.

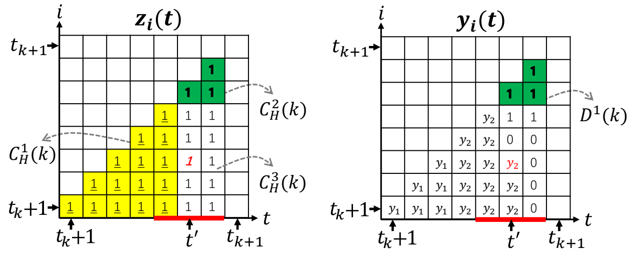
We assume that LAPDOA makes a sequence of ACKs $\pi = \{t_1, t_2, \ldots, t_K\}$, where LAPDOA makes the $i$-th ACK at the ON slot $t_i$ (i.e., $s(t_i)d(t_i) = 1$). Our goal is to show that for any $k$-th ($k \in [0, K]$) ACK interval $[t_k + 1, t_{k+1}]$ (where the first ACK interval is $[1, t_1]$ when $k = 0$ and the last ACK interval is $[t_K+1, T]$ when $k = K$), the ratio between the primal objective value and the dual objective value in this $k$-th ACK interval is at most 3, i.e., $P(k)/D(k) \le 3/\lambda$. According to Observation 4, when this ACK interval ends at slot $t_{k+1}$, $P(k)$ and $D(k)$ are never changed. This implies that LAPDOA also achieves $P/D \le 3/\lambda$ on the entire instance $\mathbf{s}$.

We first discuss the relation between $P(k)$ and $D(k)$ in the first $K$ ACK interval $[t_k+1, t_{k+1}]$ (i.e., $k \in [0, K-1]$, where there is always an ACK made at slot $t_{k+1}$), and then discuss the relation between $P(K)$ and $D(K)$ in the last ACK interval $[t_K+1, T]$ (i.e., $k = K$, where it is possible that no ACK is made at slot $T$) in the end.

Consider the $k$-th ($k \in [0, K-1]$) ACK interval $[t_k+1, t_{k+1}]$ (denoted by $I_k$), where LAPDOA makes two ACKs at the ON slots $t_k$ and $t_{k+1}$ (the first ACK interval is the 0-th ACK interval $[t_0+1, t_1]$, where $t_0 = 0$ and LAPDOA only makes one ACK at slot $t_1$), respectively. There are two cases when we make an ACK at slot $t_{k+1}$: 1) the ACK marker $M$ is equal to or larger than 1 at $t_{k+1}$; 2) the ACK marker $M$ equals or is larger than 1 at some OFF slot $t'$ ($t' < t_{k+1}$) and $t_{k+1}$ is the very first ON slot after slot $t'$ (in this case, the channels are OFF during $[t', t_{k+1}-1]$). We analyze the performance of LAPDOA in these two cases of $t_{k+1}$.

Case 1): The ACK marker $M$ is equal to or larger than 1 at $t_{k+1}$. In this case, we do not have zero updates in $I_k$. Let $\Delta P$ and $\Delta D$ denote the increment of the primal objective value and the increment of the dual objective value when we make an update, respectively. In the case of a small update, we have $\Delta P = \lambda + 1$ and $\Delta D = \lambda$, that is, $\Delta P/\Delta D = 1 + 1/\lambda$. In the case of a big update, $\Delta P = 1/\lambda + 1$ and $\Delta D = 1$, and we still have $\Delta P/\Delta D = 1 + 1/\lambda$. Obviously, for this $k$-th ACK interval, we have $P(k)/D(k) = 1 + 1/\lambda$.

Case 2): The ACK marker $M$ equals or is larger than 1 at some OFF slot $t'$ ($t' < t_{k+1}$) and $t_{k+1}$ is the very first ON slot after slot $t'$. An illustration is shown in Fig. **??**. We use $C_A(k)$ and $C_H(k)$ to denote the ACK costs and holding costs in $I_k$, respectively. Here $P(k) = C_A(k) + C_H(k)$. In addition, we assume that there are $m$ ($m \ge 0$) big updates and $n$ ($n \ge 0$) small updates in $I_k$ (some zero updates can also be made but we ignore them since they cannot increase the ACK cost and the dual variable). Given that each big update has an ACK cost of $1/\lambda$ and increases the dual variable by 1, and each small update has an ACK cost of $\lambda$ and increases the dual variable by $\lambda$, we have $C_A(k) = m/\lambda + \lambda n$ and $D(k) \ge m + \lambda n$ (i.e., $D(k)$ can be larger than $m + \lambda n$ due to the additional updates of dual variables in Lines 26-34). Now we can compute

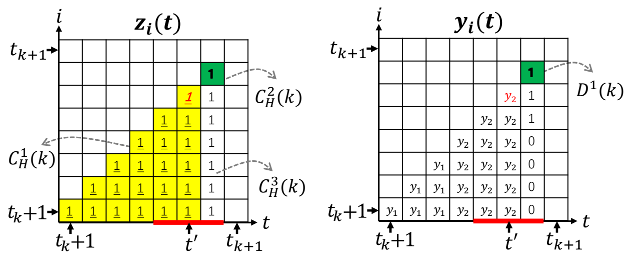


(a) Primal variables $z_i(t)$ updates. (b) Dual variables $y_i(t)$ updates.

Fig. 4. An illustration of $C_H^1(k), C_H^2(k), C_H^3(k)$ and $D^1(k)$ when $j \ne t'$. $C_H^1(k)$ is an equilateral triangle made of 1 (the underlined 1's with yellow background in Fig. 4(a)), $C_H^2(k)$ is an equilateral triangle made of 1 (the bold 1's with green background in Fig. 4(a)), $C_H^3(k)$ is a rectangle made of 1 (the regular 1's without background in Fig. 4(a)), and $D^1(k)$ is an equilateral triangle made of 1 (the bold 1's with green background in Fig. 4(b)).

$$\begin{aligned}
\frac{P(k)}{D(k)} &= \frac{C_A(k) + C_H(k)}{D(k)} \\
&= \frac{m/\lambda + \lambda n + C_H(k)}{D(k)} \\
&\le \frac{m/\lambda + \lambda n}{m + \lambda n} + \frac{C_H(k)}{D(k)} \\
&\le \frac{1}{\lambda} + \frac{C_H(k)}{D(k)} \\
&\overset{(a)}{\le} 1/\lambda + 2/\lambda \\
&= 3/\lambda,
\end{aligned} \tag{22}$$

where $(a)$ is proven in the following. Similar to the analysis of Case-(2) in the proof of Theorem 1, we can first compute the total holding cost in $I_k$ as $C_H(k) = \sum_{\tau=t_k+1}^{t_{k+1}-1} \sum_{i=t_k+1}^{\tau} z_i(\tau) = \sum_{\tau=t_k+1}^{t_{k+1}-1} \sum_{i=t_k+1}^{\tau} 1 = (t_{k+1} - t_k - 1)(t_{k+1} - t_k)/2$, where $z_i(\tau) = 1$ for any $\tau \in [t_k+1, t_{k+1}-1]$ and $i \in [t_k+1, \tau]$ is because the packets in $I_k$ are not acked until slot $t_{k+1}$, and they need to pay a holding cost, i.e., $z_i(\tau) = 1$. Next, we split $C_H(k)$ into three parts under two different cases. Assuming that the ACK marker $M$ is equal to or larger than 1 after the updating of $j$-th packet at slot $t'$. There are two cases for packet $j$: 1) packet $j$ is not packet $t'$ ($j \ne t'$), and 2) packet $j$ is packet $t'$ ($j = t'$). In the first case ($j \ne t'$), we can split $C_H(k)$ into $C_H^1(k) = \sum_{\tau=t_k+1}^{t'-1} \sum_{i=t_k+1}^{\tau} z_i(\tau)$, $C_H^2(k) = \sum_{\tau=t'}^{t_{k+1}-1} \sum_{i=t'}^{\tau} z_i(\tau)$, and $C_H^3(k) = \sum_{\tau=t'}^{t_{k+1}-1} \sum_{i=t_k+1}^{t'-1} z_i(\tau)$ (see an illustration in Fig. 4). In addition, when $j \ne t'$, we know that the total number of big updates and small updates satisfies $m + n = \sum_{\tau=t_k+1}^{t'-1} \sum_{i=t_k+1}^{\tau} z_i(\tau) + \sum_{i=t_k+1}^{j} z_i(t') \ge \sum_{\tau=t_k+1}^{t'-1} \sum_{i=t_k+1}^{\tau} z_i(\tau) = C_H^1(k)$. In the second case ($j = t'$), we can split $C_H(k)$ into $C_H^1(k) = \sum_{\tau=t_k+1}^{t'} \sum_{i=t_k+1}^{\tau} z_i(\tau)$, $C_H^2(k) = \sum_{\tau=t'+1}^{t_{k+1}-1} \sum_{i=t'+1}^{\tau} z_i(\tau)$, and $C_H^3(k) = \sum_{\tau=t'+1}^{t_{k+1}-1} \sum_{i=t_k+1}^{t'} z_i(\tau)$ (see an illustration in Fig. 5). Furthermore, when $j = t'$, we know that the total number of big updates and small updates satisfies $m + n = \sum_{\tau=t_k+1}^{t'} \sum_{i=t_k+1}^{\tau} z_i(\tau) = C_H^1(k)$. In the following, we only focus on the analysis of $D(k)$ in the first case since the analysis in the second case is very similar.



(a) Primal variables $z_i(t)$ updates. (b) Dual variables $y_i(t)$ updates.

Fig. 5. An illustration of $C_H^1(k), C_H^2(k), C_H^3(k)$ and $D^1(k)$ when $j = t'$. $C_H^1(k)$ is an equilateral triangle made of 1 (the underlined 1's with yellow background in Fig. 5(a)), $C_H^2(k)$ is an equilateral triangle made of 1 (the bold 1's with green background in Fig. 5(a)), $C_H^3(k)$ is a rectangle made of 1 (the regular 1's without background in Fig. 5(a)), and $D^1(k)$ is an equilateral triangle made of 1 (the bold 1's with green background in Fig. 5(b)).

According to Lines 26-34, for the dual variables, we have an equilateral triangle made of 1, which has the same shape as $C_H^2(k)$, we denote it by $D^1(k)$ (see an illustration in Fig. 4). We can calculate that $D^1(k) = \sum_{\tau=t'}^{t_{k+1}-1}\sum_{i=t'}^{\tau} y_i(\tau) = \sum_{\tau=t'}^{t_{k+1}-1}\sum_{i=t'}^{\tau} 1 = C_H^2(k)$. Now we can compute

$$\begin{aligned} C_H(k) &= C_H^1(k) + C_H^2(k) + C_H^3(k) \\ &\overset{(a)}{\leq} 2(C_H^1(k) + C_H^2(k)) \\ &\overset{(b)}{\leq} 2((m+n) + C_H^2(k)) \\ &= 2((m+n) + D^1(k)) \\ &\leq 2(\frac{m+\lambda n}{\lambda} + \frac{D^1(k)}{\lambda}) \\ &= 2/\lambda \cdot (m + \lambda n + D^1(k)) \\ &\overset{(c)}{\leq} 2/\lambda \cdot D(k), \end{aligned} \tag{23}$$

where $(a)$ can be proven using the same techniques in Eq. (20), $(b)$ is because $C_H^1(k)$ is at least $m+n$ as we analyzed above, and $(c)$ is because $D(k)$ is least $m + \lambda n + D^1(k)$ (i.e., $D(k)$ can be larger than $m + \lambda n + D^1(k)$ due to Lines 26-34). This completes (a) in Eq. (22).

In the end, we consider the last time interval $[t_K+1, T]$ (denoted by $I_K$). If the last slot is an ON slot and LAPDOA makes the last ACK exactly at the last slot, i.e., $t_K = T$, then our previous analysis in Cases 1 and 2 still holds. Next, we consider the scenario where $t_K < T$. There are two cases at slot $T$: 1) $T$ is the slot before the ACK marker $M$ equals or is larger than 1; 2) $T$ is the slot when or after the ACK marker $M$ equals or is larger than 1. In both cases, there is no ACK made during $I_K$. We use $P(K)$ and $D(K)$ to denote the primal objective value and the dual objective value in $I_K$, respectively.

Case 1): $T$ is the slot before the ACK marker $M$ equals or is larger than 1. In this case, we do not have zero updates in $I_K$. Let $\Delta P$ and $\Delta D$ denote the increment of the primal objective value and the increment of the dual objective value when we make an update, respectively. In the case of a small update, we have $\Delta P = c{\cdot}0+1 = 1$ and $\Delta D = \lambda$, that is, $\Delta P/\Delta D = 1/\lambda$. In the case of a big update, $\Delta P = c \cdot 0 + 1 = 1$ and $\Delta D = 1$, so we have $\Delta P/\Delta D = 1$. Obviously, for this $K$-th ACK interval, we have $P(K)/D(K) \leq 1/\lambda$.

Case 2): $T$ is the slot when or after the ACK marker $M$ equals or is larger than 1. We assume that the ACK marker $M$ equals or is larger than 1 at slot $t'$ ($t' > t_K$). According to the definition of Case 2, we have $T \geq t'$. We claim that the channels are OFF during the interval $[t', T]$. Otherwise, an ACK will be made during $[t', T]$, which contradicts the fact that there is no ACK during $I_K$. We use $C_A(K)$ to denote the total ACK cost and use $C_H(K)$ to denote the total holding cost in $I_K$. Here $P(K) = C_A(K) + C_H(K) = 0 + C_H(K)$, where $C_A(K) = 0$ because there is no ACK made during $I_K$. We have

$$\begin{aligned} P(K)/D(K) &= (C_A(K) + C_H(K))/D(K) \\ &= 0 + C_H(K)/D(K) \\ &\overset{(a)}{\leq} 2/\lambda, \end{aligned} \tag{24}$$

where the analysis in $(a)$ is the same as the (a) in Eq. (22).

In summary, LAPDOA achieves $P(k)/D(k) \leq 3/\lambda$ for any ACK interval, and thus LAPDOA also achieves $P/D \leq 3/\lambda$ on the entire instance.

## D. *Proof of Lemma 3*

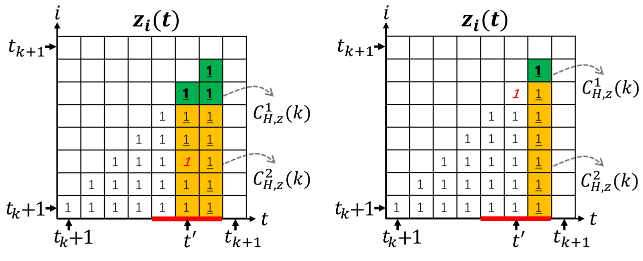
*Proof.* In this proof, similar to the proof of Lemma 2, we still assume that $d(t)$ is updated to the ACK marker $M$ ($M \geq 1$) rather than 1 in Line 15 (except the analysis of big updates in the case of $\lambda \in (0, 1/c]$, where we still update $d(t)$ to be 1). Therefore, for the big updates and small updates in any ACK interval, each big update can be charged with an ACK cost of $1/\lambda$, and each small update can be charged with an ACK cost of $\lambda$. In particular, for the big updates and small updates in the last time interval $[t_K, T]$ (we assume that LAPDOA makes the last ACK at slot $t_K$), though there is no ACK made during $[t_K, T]$, we still charge each big update and each small update with an ACK cost of $1/\lambda$ and $\lambda$, respectively. Doing this can possibly increase the total cost of LAPDOA, but we can show that the upper bound we derived still holds in this case.

We first consider $\lambda \in (0, 1/c]$. In this case, LAPDOA has three types of updates: big updates, small updates, and zero updates. Consider the total cost of big updates first. Once the prediction $\mathcal{P}$ makes an ACK at some ON slot $t$, LAPDOA will make a big update immediately, the ACK marker becomes $M = 1/\lambda c = 1/c \cdot 1/\lambda \geq 1/c \cdot c = 1$, and thus LAPDOA will also make an ACK at the beginning of slot $t$. Since the prediction $\mathcal{P}$ has a total number of $C_A(\mathrm{s}, \mathcal{P})/c$ ACKs, then LAPDOA has also $C_A(\mathrm{s}, \mathcal{P})/c$ big updates. Each big update leads to an ACK, which results in an ACK cost of $c$. Therefore, the total cost of the big updates in LAPDOA is $C_A(\mathrm{s}, \mathcal{P})/c \cdot c = C_A(\mathrm{s}, \mathcal{P})$. By the definition of small update and zero update, each small update or zero update in LAPDOA corresponds to one packet in the prediction $\mathcal{P}$ that has not been acked yet, which requires the prediction $\mathcal{P}$ to pay a holding cost of 1, so the total number of small updates and zero udpates is at most $C_H(\mathrm{s}, \mathcal{P})/1 = C_H(\mathrm{s}, \mathcal{P})$. For each of the small updates, the increase in the primal is $\Delta P = \lambda + 1$, and for each of the zero updates, the increase in the primal is $\Delta P = 0 + 1 = 1$, so the total cost of small updates and zero updates is at most $(1+\lambda)C_H(\mathbf{s},\mathcal{P})$. This concludes Eq. (12).

Next, we analyze $\lambda \in (1/c, 1]$. We consider two cases: 1) the channels are ON all the time, and 2) there are some OFF channels. We show that Eq. (13) holds for both cases.

Case 1): The channels are ON all the time. In this case, LAPDOA will generate only two types of updates: small updates and big updates. By the definition of small updates, for any of them, there is one corresponding packet in the prediction $\mathcal{P}$ that has not been acked yet, which requires the prediction $\mathcal{P}$ to pay a holding cost of 1 for this packet, so the total number of the small updates is at most $C_H(\mathbf{s},\mathcal{P})/1 = C_H(\mathbf{s},\mathcal{P})$. Each small update contributes $(\lambda+1)$ to the primal objective value, thus the total cost of small updates is at most $(1+\lambda)C_H(\mathbf{s},\mathcal{P})$. Next, we analyze the total cost of big updates. We claim that for any ACK made by the prediction $\mathcal{P}$, LAPDOA makes at most $\lceil \lambda c \rceil$ big updates for this ACK. To see this, assuming that prediction $\mathcal{P}$ makes an ACK at slot $t$, and consider all the big updates due to this ACK (i.e., those big updates produced by the packets in LAPDOA that have not been acked yet and arrives before or at slot $t$). After at most $\lceil \lambda c \rceil$ such big updates, the ACK marker will become $M = \lceil \lambda c \rceil \cdot 1/\lambda c \geq 1$ at some slot $t' \geq t$. Once the ACK marker $M$ equals or is larger than 1, no more big updates will be made for the ACK made by prediction $\mathcal{P}$ at slot $t$. Given that prediction $\mathcal{P}$ makes $C_A(\mathbf{s},\mathcal{P})/c$ ACKs, then LAPDOA makes at most $\lceil \lambda c \rceil \cdot C_A(\mathbf{s},\mathcal{P})/c$ big updates. For each big update, the increment in the primal objective value is $\Delta P = 1/\lambda + 1$. Therefore, the total cost of big updates is at most $(1/\lambda+1)\cdot\lceil \lambda c \rceil \cdot C_A(\mathbf{s},\mathcal{P})/c$. In summary, when the channels are always ON, the total cost of LAPDOA is upper bounded by $C(\mathbf{s},\mathcal{P},\lambda) \leq (\lambda+1)C_H(\mathbf{s},\mathcal{P}) + (1/\lambda+1)\lceil \lambda c \rceil C_A(\mathbf{s},\mathcal{P})/c$, which is smaller than the bound in Eq. (13).

Case 2): There are some OFF channels. In this case, LAPDOA will generate three types of updates: small updates, big updates, and zero updates. Note that these zero updates only increase the holding costs. We use $C_{A,s}$ and $C_{H,s}$ to denote the ACK cost and the holding cost of all the small updates, respectively; use $C_{A,b}$ and $C_{H,b}$ to denote the ACK cost and the holding cost of all the big updates, respectively; and $C_{H,z}$ to denote the holding cost of all the zero updates. Obviously, we have $C(\mathbf{s},\mathcal{P},\lambda) = C_{A,b} + C_{H,b} + C_{A,s} + C_{H,s} + C_{H,z}$. Furthermore, similar to the analysis in Case 1, for the big updates, we can obtain $C_{A,b} + C_{H,b} \leq (1/\lambda+1)\cdot\lceil \lambda c \rceil \cdot C_A(\mathbf{s},\mathcal{P})/c$; and for the small updates, we have $C_{A,s} + C_{H,s} \leq (\lambda+1)C_H(\mathbf{s},\mathcal{P})$. To analyze the holding costs of zero updates $C_{H,z}$, our idea is to bound it by the holding cost of small updates and big updates, which can be further bounded by the total cost of prediction. To this end, we assume that LAPDOA makes a sequence of ACKs $\pi = \{t_1, t_2, \ldots, t_K\}$, where LAPDOA makes the $i$-th ACK at the ON slot $t_i$ (i.e., $s(t_i)d(t_i) = 1$). Our goal is to show that in any $k$-th ($k \in [0, K]$) ACK interval $[t_k+1, t_{k+1}]$ (where the first ACK interval is $[1, t_1]$ when $k = 0$ and the last ACK interval is $[t_K+1, T]$ when $k = K$), the holding costs of zero updates can be bounded by the holding cost of small updates and big updates.



(a) when $j \neq t'$ (b) when $j = t'$

Fig. 6. An illustration of $C_{H,z}(k)$, $C^1_{H,z}(k)$, and $C^2_{H,z}(k)$ in two different cases ($j \neq t'$ and $j = t'$). $C_{H,z}(k)$ is the sum of the underlined 1's, $C^1_{H,z}(k)$ is an equilateral triangle made of 1 (the bold underlined 1's with green background), and $C^2_{H,z}(k)$ is a rectangle made of 1 (the sum of some regular 1's and some underlined 1's with orange background).

We consider the $k$-th ($k \in [0, K-1]$) ACK interval $[t_k+1, t_{k+1}]$ (denoted by $I_k$), where the LAPDOA makes two ACKs at the ON slots $t_k$ and $t_{k+1}$, respectively. Still, there are two cases when we make an ACK at $t_{k+1}$: 1) the ACK marker $M$ is equal to or larger than 1 at $t_{k+1}$ and $t_{k+1}$ is an ON slot; 2) the ACK marker $M$ equals or is larger than 1 at some OFF slot $t'$ ($t' < t_{k+1}$) and $t_{k+1}$ is the very first ON slot after slot $t'$. Note that though the following analysis is for the general ACK interval $[t_k+1, t_{k+1}]$ ($k \in [0, K-1]$), they can easily extended the last ACK interval $[t_K+1, T]$.

1) The ACK marker $M$ is equal to or larger than 1 at the ON slot $t_{k+1}$. In this case, there is no zero update, and the holding cost of zero updates is 0.
2) The ACK marker $M$ equals or is larger than 1 at some OFF slot $t'$ ($t' < t_{k+1}$) and $t_{k+1}$ is the very first ON slot after slot $t'$ (i.e., the channels are OFF during $[t', t_{k+1}-1]$). Assuming that the ACK marker $M$ is equal to or larger than 1 after the updating of the $j$-th packet at slot $t'$. In this case, for the holding costs of zero updates in $I_k$ (denoted by $C_{H,z}(k)$), we can compute it under two different cases based on packet $j$: 1) packet $j$ is not packet $t'$ ($j \neq t'$), and 2) packet $j$ is packet $t'$ ($j = t'$). In the first case ($j \neq t'$), we can denote $C_{H,z}(k)$ as $C_{H,z}(k) = \sum_{i=j+1}^{t'} z_i(t') + \sum_{\tau=t'+1}^{t_{k+1}-1}\sum_{i=t_k+1}^{\tau} z_i(\tau)$ (see an illustration in Fig. 6(a)); and in the second case ($j = t'$), we can denote $C_{H,z}(k)$ as $C_{H,z}(k) = \sum_{\tau=t'+1}^{t_{k+1}-1}\sum_{i=t_k+1}^{\tau} z_i(\tau)$ (see an illustration in Fig. 6(b)). In the following, we only focus on the analysis of $C_{H,z}(k)$ in the first case since the analysis in the second case is very similar. Next, we bound $C_{H,z}(k)$ by two areas: $C^1_{H,z}(k) \triangleq \sum_{\tau=t'}^{t_{k+1}-1}\sum_{i=t'}^{\tau} z_i(\tau)$ and $C^2_{H,z}(k) \triangleq \sum_{\tau=t'}^{t_{k+1}-1}\sum_{i=t_k+1}^{t'-1} z_i(\tau)$ (see an illustration in Fig. 6(a)). Clearly, we have $C_{H,z}(k) = \sum_{i=j+1}^{t'} z_i(t') + \sum_{\tau=t'+1}^{t_{k+1}-1}\sum_{i=t_k+1}^{\tau} z_i(\tau) \leq \sum_{\tau=t'}^{t_{k+1}-1}\sum_{i=t'}^{\tau} z_i(\tau) + \sum_{\tau=t'}^{t_{k+1}-1}\sum_{i=t_k+1}^{t'-1} z_i(\tau) = C^1_{H,z}(k) + C^2_{H,z}(k)$. We use $C^1_{H,z}$ to denote the sum of $C^1_{H,z}(k)$ over all the ACK intervals $I_k$ ($k \in [0, K]$), i.e., $C^1_{H,z} \triangleq \sum_{k=0}^{K} C^1_{H,z}(k)$. For any of the zero updates in $C^1_{H,z}$, there is one

corresponding packet in the prediction $\mathcal{P}$ that has not been acked yet since the channels are OFF during some $[t', t_{k+1} - 1]$, which requires the prediction $\mathcal{P}$ to pay a holding cost of 1 for this packet. Similarly, for any of the small updates, by the definition of small updates, there is one corresponding packet in the prediction $\mathcal{P}$ that has not been acked yet, which requires the prediction $\mathcal{P}$ to pay a holding cost of 1 for this packet. Therefore, the total number of the zero updates in $C^1_{H,z}$ and the small updates is at most $C_H(\mathbf{s}, \mathcal{P})/1 = C_H(\mathbf{s}, \mathcal{P})$, and each of such update has a total cost at most $(1 + \lambda)$, which indicates that the total cost of the zero updates in $C^1_{H,z}$ and the small updates is at most $(1 + \lambda)C_H(\mathbf{s}, \mathcal{P})$. That is,

$$C_{A,s} + C_{H,s} + C^1_{H,z} \leq (1 + \lambda)C_H(\mathbf{s}, \mathcal{P}). \tag{25}$$

For the holding cost $C^2_{H,z}(k)$, same as the analysis in $(a)$ of Eq. (23), it is upper bounded by the sum of the holding cost of big updates and small updates in $I_k$ and the holding cost of the zero updates in $C^1_{H,z}(k)$, i.e., $C^2_{H,z}(k) \leq C_{H,s}(k) + C_{H,b}(k) + C^1_{H,z}(k)$. More generally, let $C^2_{H,z}$ denote the sum of $C^2_{H,z}(k)$ over all the ACK intervals $I_k$ $(k \in [0, K])$, i.e., $C^2_{H,z} \triangleq \sum_{k=0}^{K} C^2_{H,z}(k)$. Then we have

$$\begin{aligned}
C^2_{H,z} &\leq C_{H,s} + C_{H,b} + C^1_{H,z} \\
&= (C_{H,s} + C^1_{H,z}) + C_{H,b} \\
&\overset{(a)}{\leq} C_H(\mathbf{s}, \mathcal{P}) + C_{H,b} \\
&\overset{(b)}{\leq} C_H(\mathbf{s}, \mathcal{P}) + \lceil \lambda c \rceil \times \frac{C_A(\mathbf{s}, \mathcal{P})}{c},
\end{aligned}$$

where $(a)$ is because as we showed before, the total number of the zero updates in $C^1_{H,z}$ and the small updates is at most $C_H(\mathbf{s}, \mathcal{P})$, and each of small updates or zero updates increases the holding costs of LAPDOA by 1, so the total holding cost of them is at most $C_H(\mathbf{s}, \mathcal{P})$; $(b)$ is due to the total number of big updates is at most $\lceil \lambda c \rceil \times C_A(\mathbf{s}, \mathcal{P})/c$, and each of big updates increases the holding costs by 1, so we have $C_{H,b} \leq \lceil \lambda c \rceil \times C_A(\mathbf{s}, \mathcal{P})/c$.

Finally, the total cost of LAPDOA in Case 2 is

$$\begin{aligned}
&C(\mathbf{s}, \mathcal{P}, \lambda) && (26)\\
&= C_{A,b} + C_{H,b} + C_{A,s} + C_{H,s} + C_{H,z} && (27)\\
&\leq C_{A,b} + C_{H,b} + C_{A,s} + C_{H,s} + C^1_{H,z} + C^2_{H,z} && (28)\\
&= (C_{A,b} + C_{H,b}) + (C_{A,s} + C_{H,s} + C^1_{H,z}) + C^2_{H,z} && (29)\\
&\leq (1/\lambda + 1) \times \lceil \lambda c \rceil \times \frac{C_A(\mathbf{s}, \mathcal{P})}{c} + (1 + \lambda)C_H(\mathbf{s}, \mathcal{P}) && (30)\\
&\quad + C_H(\mathbf{s}, \mathcal{P}) + \lceil \lambda c \rceil \times \frac{C_A(\mathbf{s}, \mathcal{P})}{c} && (31)\\
&= (1/\lambda + 2) \times \lceil \lambda c \rceil \times \frac{C_A(\mathbf{s}, \mathcal{P})}{c} + (2 + \lambda)C_H(\mathbf{s}, \mathcal{P}). && (32)
\end{aligned}$$

Finally, combining the results in Case 1 and Case 2, we see that Eq. (13) holds. □