How Useful Is Delayed Feedback in AoI Minimization — A Study on Systems With Queues in Both Forward and Backward Directions

Chih-Chun Wang, Email: chihw@purdue.edu
Elmore Family School of ECE, Purdue University, USA

***Abstract*—One canonical example of Age-Of-Information (AoI) minimization is the *update-through-queues* models. Existing results fall into two categories: The open-loop setting for which the sender is oblivious of the actual packet departure time, versus the closed-loop setting for which the decision is based on *instantaneous* Acknowledgement (ACK). Neither setting perfectly reflects modern networked systems, which almost always rely on *feedback that experiences some delay.* Motivated by this observation, this work subjects the ACK traffic to an independent queue so that the closed-loop decision is made based on delayed feedback. Near-optimal schedulers have been devised, which smoothly transition from the instantaneous-ACK to the open-loop schemes depending on how long the feedback delay is. The results thus quantify the benefits of delayed feedback for AoI minimization in the update-through-queues systems.**

I. Introduction

Supporting low-latency applications is a top mission of modern communication networks, with 5G NR aiming for 1ms latency and even shorter delay (100μs) for 6G and beyond. One example application is remote control in cyber-physical systems (CPS). For example, [1] studies *linear quadratic Gaussian* control systems [2] with random delay, and shows that the control performance deteriorates exponentially fast with respect to the *Age of* (the measurement) *Information* (AoI) at the controller. The intuition is that any control action at time t based on measurements that are Δ-time old inevitably leaves the state disturbance accumulated during time interval $(t-\Delta, t]$ unchecked, which incurs exponential cost $e^{c\Delta}$ since the system state drifts exponentially in time if unchecked.

With strong relationships between AoI and underlying system performance [3]–[5], AoI minimization has attracted significant research attention. One earliest canonical example is *information update through queues* [6]–[17]. Specifically, source s sends update packets through a queue to destination d. The AoI at d is defined as

$$\Delta(t) \triangleq t - \max\{S_i : \forall i \text{ s.t. } D_i < t\} \tag{1}$$

where S_i is the *send time* of the i-th packet P_i (the time of injecting P_i into the queue) and D_i is the *delivery time* (the time P_i departs the queue). The objective is to design $\{S_i : i\}$ that minimizes the *average AoI*.

This work was supported in parts by NSF Grants CCF-1618475, CCF-1816013, CCF-2008527, CNS-2107363, and also by National Spectrum Consortium (NSC) under grant W15QKN-15-9-1004.

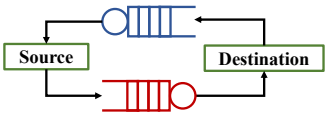

Fig. 1. System with Queues in Both Forward and Backward Directions.

Existing results of this model generally fall into two categories: Open loop versus closed loop. In the open loop settings [6], [7], [13], [14], [16], the sender is oblivious of the actual packet departure time. Performance analysis has been conducted for different service policies, e.g., Last-Come-First-Serve (LCFS), and the scheduling scheme generally follows a stationary randomized design. In the closed-loop settings [8]–[12], [15], [17] s has instantaneous ACK of the packet departure time. Optimal $\{S_i : \forall i\}$ are analyzed for different variations, including AoI penalty functions [8], [10], transmission cost [11], [12], and provably optimal distribution-oblivious online algorithm [10].

Nonetheless, modern network protocols almost always rely on feedback that experiences some (random) delay, especially in a remote estimation/control environment. It remains unclear whether one should employ a scheme designed for instantaneous ACK while knowing the feedback being used is actually stale, or one should take an open-loop approach that discards the delayed feedback completely. Intuitively, even though delayed feedback is not as valuable as instantaneous ACK, it should still contain some information that can assist scheduling. The question to answer, though, is how to design *schemes that take full advantage of the delayed feedback.*

With this motivation, this work subjects the ACK traffic to an independent queue so that the closed-loop decision is based on delayed ACK. See Fig. 1. The main contributions are:

(1) With delayed feedback, *source s sometimes has to make a decision before the arrival of ACK since ACK may arrive too late due to feedback delay*. However, it was not clear how to define the "information" when s has *not* received the ACK. This work rigorously formulates the problem via the *stopping-time* terminology, which includes both the instantaneous-ACK [8] and open-loop settings [6] as special cases.

(2) We explicitly design a new near-optimal achievability scheme and a *genie-aided* converse result. Jointly, these new bounds tightly bracket the optimal AoI with delayed feedback. The results characterize a smooth transition such that the shorter the feedback delay, the closer the optimal performance

is to the closed-loop setting; and the longer the feedback delay, the performance becomes similar to the open-loop setting.

(3) Our results quantify the benefits of closed-loop over open-loop schemes, a critical piece of information for system designers. E.g., numerical evaluation shows that if the forward and feedback queues have comparable service time, then the benefits of delayed feedback almost vanish completely and we can use the (simpler) open-loop approach to achieve near-optimal performance. On the other hand, if the feedback delay is roughly half of the forward delay, then significant gain can still be achieved under a closed-loop design.

II. Problem Formulation

This work assumes *slotted time axis*, i.e., the injection and departure times of both queues in Fig. 1 are integers. Such an assumption is not restrictive since most numerical methods do quantize the time axis [5]. At time 0, both queues are empty. For any $i \geq 1$, source s would inject packet P_i to the forward queue at the *send time* S_i. P_i will leave the forward queue and arrive at destination d at the *delivery time* D_i. Once delivered, the ACK packet of P_i, denoted by Ack_i, is immediately injected to the backward queue (thus at time D_i). Ack_i will leave the backward queue at the *ACK time* A_i. Once arrived, Ack_i will inform s the exact delivery time D_i.

For each packet P_i (and its corresponding Ack_i) we denote the i.i.d. service times of the forward and backward queues by $Y_i \sim \mathbb{P}_Y$ and $Z_i \sim \mathbb{P}_Z$, respectively. $\mathbb{P}_Y$ and $\mathbb{P}_Z$ can be arbitrary distributions with bounded supports $[1, y_{\max}]$ and $[0, z_{\max}]$, respectively. The assumption of $Y_i \geq 1$ is to avoid the complication of *instantaneous forward delivery*. Initialize $S_0 = D_0 = A_0 = 0$. Under the FIFO-queue model, we have

$$D_i = \max(S_i, D_{i-1}) + Y_i \tag{2}$$

$$A_i = \max(D_i, A_{i-1}) + Z_i \tag{3}$$

For any $i \geq 1$, define a random process $\mathsf{ack.del}_i(t) \triangleq D_i \cdot 1_{\{A_i \leq t\}}$, which jumps from 0 to D_i at ACK time A_i and stays at D_i afterward, i.e., $\mathsf{ack.del}_i(t)$ is the *acknowledged-delivery-time at time* t. Define $\mathbb{F}^{(i)} \triangleq \{\mathcal{F}_t^{(i)} : t \in [1, \infty)\}$ as the filtration generated by random processes $\{\mathsf{ack.del}_j(t) : j \in [1, i-1]\}$. I.e., σ-algebra $\mathcal{F}_t^{(i)}$ contains all the information available to s when making the S_i decision at time t.

This work studies the following AoI minimization problem:

$$\mathsf{avg.aoi}^* \triangleq \min_{\{S_i : i \geq 1\}} \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \Delta(t) \tag{4}$$

$$\text{subject to} \quad \forall i \in [1, \infty),\ S_{i-1} \leq S_i \text{ and} \tag{5}$$

$$S_i \text{ is a } \textit{stopping time} \text{ w.r.t. } \mathbb{F}^{(i)} \tag{6}$$

where the AoI $\Delta(t)$ is defined in (1); and (5) ensures that the packet index i is the chronological order of transmission, i.e., P_{i-1} is, by definition, sent at an earlier time than P_i.

Our model is general. For example, we can choose $\mathbb{P}_Z$ to be instantaneous ACK $\mathbb{P}(Z_i = 0) = 1$ [8], to be deterministic but non-zero, to be (truncated) log-normal distribution, or to be $\mathbb{P}(Z_i = z_{\max}) = 1$ for a large $z_{\max}$ that mimics the open-loop setting in which feedback never arrives.

We now present a straightforward lemma.

Lemma 1: $\exists B < \infty$, the value of which depends on $y_{\max}$ and $z_{\max}$, such that the optimal scheme must satisfy $S_i \leq S_{i-1} + B$.

The intuition behind is as follows. With bounded support $y_{\max}$ and $z_{\max}$, if we wait long enough, Ack_{i-1} will return to s and the previous packet P_{i-1} will be fully ACK'ed. Since [8] shows that under some mild assumptions the optimal waiting time is bounded in a fully ACK'ed system, the optimal waiting time must also be bounded for the delayed feedback system.

A. Four existing upper and lower bounds of $\mathsf{avg.aoi}^*$

Zero-Wait-After-ACK (ZWAA) is a scheme for which s sends P_i immediately after receiving Ack_{i-1}, i.e., $S_i = A_{i-1}$. By analyzing its performance, we derive an upper bound

$$\mathsf{zwaa} = \mathbb{E}(Y) + 0.5 + \frac{\mathbb{E}(Y^2) + 2\mathbb{E}(Y)\mathbb{E}(Z) + \mathbb{E}(Z^2)}{2 \cdot (\mathbb{E}(Y) + \mathbb{E}(Z))} \tag{7}$$

Best-After-ACK (BAA) [10] adds an additional constraint $S_i \geq A_{i-1}$ to (4)–(6) and then solves the optimal value of the restricted problem. The result is an upper bound, which we denote by baa. By definition, we always have $\mathsf{baa} \leq \mathsf{zwaa}$.

Optimal periodic (Opt.Per) is an open-loop scheme which chooses $S_i = \lfloor (i-1) \cdot c \rfloor$ where $c > 0$ is the real-valued period being used. We numerically select the optimal c^* that leads to the smallest $\mathsf{avg.aoi}$ in Monte-Carlo simulation. The result is an upper bound, which we denote by $\mathsf{opt.per}$.

Instantaneous ACK (Inst.ACK) hardwires $\mathbb{P}(Z_i = 0) = 1$ and uses [8] to compute the optimal value. The result is a lower bound of $\mathsf{avg.aoi}^*$, which we denote by $\mathsf{inst.ack}$.

As will be shown later, none of zwaa, baa, $\mathsf{opt.per}$, and $\mathsf{inst.ack}$ is tight for $\mathsf{avg.aoi}^*$ in a general setting.

III. Main Result #1: A New Lower Bound

Our lower bound is derived by analyzing the performance of a genie-aided scheme. Specifically, for any $i \geq 1$, at time $t = \max(S_{i-1}, D_{i-2})$ the genie will temporarily take over[1] the backward queue and deliver all ACKs in the backward queue, including Ack_{i-2}, to source s *instantaneously*. Note that the genie does not alter the forward queue in any way. We now argue that the optimal genie-aided scheme must satisfy

Rule 1: During time $t \in (S_{i-1}, \max(S_{i-1}, D_{i-2})]$, source s waits and does not generate/send the current packet P_i.

The reason is as follows. If $A_{i-2} \leq S_{i-1} - 1$, then s knows the value of D_{i-2} at time A_{i-2}. If s has not received Ack_{i-2} by time $S_{i-1} - 1$, then our special genie will deliver Ack_{i-2} to s at time $\max(S_{i-1}, D_{i-2})$. In either case, s knows the value of $\max(S_{i-1}, D_{i-2})$ either by the normal delivery of Ack_{i-2} or by the genie-assisted delivery at time $\max(S_{i-1}, D_{i-2})$.

By (2), the previous packet P_{i-1} will only be delivered *after* time $\max(S_{i-1}, D_{i-2})$. Equipped with the knowledge

[1] In our model, both the forward and backward FIFO queues are beyond the control of the sender, the same as in [3]–[5], [8]–[12]. However, when deriving an *impossibility result* (a lower bound), we utilize a genie who is not limited by this constraint and can directly manipulate/circumvent the backward queue. Its performance thus serves as a lower bound of our original problem.

of $\max(S_{i-1}, D_{i-2})$, s should always delay the transmission of P_i to be after $\max(S_{i-1}, D_{i-2})$ and make sure P_i is not stuck behind P_{i-1}, which is bad for AoI minimization.

Note that when s is deciding whether to send the current P_i, *it keeps accumulating more and more knowledge via observing* $\{\mathsf{ack.del}_j(t) : j \in [1, i-1]\}$ *over time.* Further dissecting this knowledge accumulation enables us to prove that the optimal scheme must have the following additional structure.

Consider two "waiting time functions" $\phi_{\mathsf{na}} : [0, y_{\max}] \mapsto [0, B]$ and $\phi_{\mathsf{a}} : [1, 2y_{\max} + z_{\max}] \mapsto [0, B]$.

Rule 2: At time $t = \max(S_{i-1}, D_{i-2})$, source s computes the value of $x^*_{\mathsf{na}} \triangleq \phi_{\mathsf{na}}\left((D_{i-2} - S_{i-1})^+\right)$ where $(\cdot)^+ \triangleq \max(\cdot, 0)$. If Ack_{i-1} has not arrived by time $\max(S_{i-1}, D_{i-2}) + x^*_{\mathsf{na}}$, then s will send P_i at that time. Namely, x^*_{na} is the additional waiting time after $\max(S_{i-1}, D_{i-2})$ if Ack_{i-1} has not arrived by then.

Rule 3: If Ack_{i-1} has arrived at an earlier time than $\max(S_{i-1}, D_{i-2}) + x^*_{\mathsf{na}}$, i.e., $A_{i-1} \le \max(S_{i-1}, D_{i-2}) + x^*_{\mathsf{na}}$, then at time $t = A_{i-1}$, source s computes $x^*_{\mathsf{a}} \triangleq \phi_{\mathsf{a}}(A_{i-1} - S_{i-1})$ and will send P_i at time $A_{i-1} + x^*_{\mathsf{a}}$. Namely, x^*_{a} is the additional waiting time after A_{i-1}.

In sum, we use Rule 2 initially, but would switch to Rule 3 at time A_{i-1} if the *precomputed send-time* $\max(S_{i-1}, D_{i-2}) + x^*_{\mathsf{na}}$ has not been "committed" by the time Ack_{i-1} arrives.

We now explain why the optimal scheme must satisfy Rules 2 and 3. At time $\max(S_{i-1}, D_{i-2})$, source s is still waiting for Ack_{i-1} due to (2). If Ack_{i-1} had not arrived for some interval, then no additional "variable" is revealed to s during that interval. Therefore, s *can anticipate the situation and pre-compute the decision* S_i *at time as early as* $t = \max(S_{i-1}, D_{i-2})$*, assuming* Ack_{i-1} *arrives later than that decision.* See Rule 2.

Furthermore, at time $\max(S_{i-1}, D_{i-2})$, packet P_{i-1} has just started to be processed in the forward queue, the "state" of the system is thus how much the AoI has grown when P_{i-1} was idled in the forward queue, which is $\max(S_{i-1}, D_{i-2}) - S_{i-1} = (D_{i-2} - S_{i-1})^+$. That is why we compute x^*_{na} by $\phi_{\mathsf{na}}\left((D_{i-2} - S_{i-1})^+\right)$ in Rule 2.

Similarly for Rule 3, if Ack_{i-1} were delivered before time $\max(S_{i-1}, D_{i-2}) + x^*_{\mathsf{na}}$, then at time A_{i-1}, *source* s *would know with 100% certainty that both the forward and backward queues are empty at that moment,* a new piece of information "revealed" to s at time A_{i-1}. As a result, s switches to a new waiting time decision x^*_{a}. The system state at time A_{i-1} is again how much the AoI has grown, which is $A_{i-1} - S_{i-1}$, and we use it to compute x^*_{a} by $\phi_{\mathsf{a}}(A_{i-1} - S_{i-1})$.

The ranges of ϕ_{na} and ϕ_{a} are both $[0, B]$ because of Lemma 1. The domain of ϕ_{na} is $[0, y_{\max}]$. The reason is that Rule 1 implies that $S_{i-1} \ge \max(S_{i-2}, D_{i-3})$. By (2) we have $D_{i-2} \le S_{i-1} + y_{\max}$ and thus $(D_{i-2} - S_{i-1})^+ \in [0, y_{\max}]$. Similar reasons can show that $A_{i-1} - S_{i-1} \in [1, 2y_{\max} + z_{\max}]$, which is the domain of ϕ_{a}.

Using Rules 1 to 3, one can convert (4)–(6) to an average cost per stage (ACPS) problem of semi-Markov decision process (semi-MDP) [18]. Specifically, we define two functions to be used shortly:

$$m^+_Y(x) \triangleq \mathbb{E}(Y) + \mathbb{E}\left((Y - x)^+ \middle| Y + Z > x\right) \tag{8}$$

$$\gamma(\delta, y) \triangleq \frac{\delta^2}{2} + \delta \cdot (y + 0.5) \tag{9}$$

and use $\bar{a} = A_{i-1} - S_{i-1}$ and $\bar{d} = (D_{i-2} - S_{i-1})^+$ as shorthand for the state values used in the waiting time functions ϕ_{a} and ϕ_{na}, respectively. We denote the *value functions* of the semi-MDP by $f_{\mathsf{a}}(\bar{a})$ and $f_{\mathsf{na}}(\bar{d})$, where the former (resp. the latter) corresponds to Rule 3 (resp. Rule 2). Then the Bellman equations become: $\forall \bar{a} \in [1, 2y_{\max} + z_{\max}]$ we have

$$f_{\mathsf{a}}(\bar{a}) = \min_{x \in [0, B]} \gamma(\bar{a} + x, \mathbb{E}(Y)) - \mathsf{v} \cdot (\bar{a} + x) + f_{\mathsf{na}}(0) \tag{10}$$

and $\forall \bar{d} \in [0, y_{\max}]$ we have

$$f_{\mathsf{na}}(\bar{d}) = \min_{x \in [0, B]} \Bigg\{ \sum_{k=1}^{x} \mathbb{P}(Y + Z = k) \cdot f_{\mathsf{a}}(\bar{d} + k) \tag{11}$$

$$+ \mathbb{P}(Y + Z > x) \cdot \Big(\gamma\big(\bar{d} + x, m^+_Y(x)\big) - \mathsf{v} \cdot (\bar{d} + x) \tag{12}$$

$$+ \sum_{y=1}^{y_{\max}} \mathbb{P}(Y = y | Y + Z > x) \cdot f_{\mathsf{na}}\left((y - x)^+\right) \Big) \Bigg\} \tag{13}$$

For the readers who are familiar with the AoI derivations in [8], [10], the function $\gamma(\delta, y)$ in (9) is the AoI cost of letting the *total waiting time* to be δ when the forward delay is y. Recall that f_{a} is the value function after receiving Ack_{i-1} at time $t = A_{i-1}$. In this case, any additional waiting time will result in the total waiting time being $A_{i-1} - S_{i-1} + x = \bar{a} + x$. The resulting AoI cost is thus $\gamma(\bar{a} + x, \mathbb{E}(Y))$ since the forward queue is empty at time A_{i-1} and the new packet P_i will take, in average, $\mathbb{E}(Y)$ to go through the empty forward queue. This leads to the first half of the expression of $f_{\mathsf{a}}(\bar{a})$ in (10).

The term "$-\mathsf{v} \cdot (\bar{a} + x)$" in (10) is a generalization of the average-cost adjustment term of ACPS-MDP to its counterpart for *ACPS-semi-MDP.* Finally, since $S_i = A_{i-1} + x \ge D_{i-1}$, when transmitting the next packet P_{i+1}, we will face a new $\bar{d} = (D_{i-1} - S_i)^+ = 0$. That is why in (10) the "next state value" is always $f_{\mathsf{na}}(0)$. The argmin x^* value in (10) gives us the optimal waiting time function $\phi_{\mathsf{a}}(\bar{a})$.

Now consider Rule 2. Suppose we choose to waiting x slots after time $\max(S_{i-1}, D_{i-2})$. We first consider the event that $A_{i-1} \le \max(S_{i-1}, D_{i-2}) + x$. Because of the genie, we have $A_{i-1} = \max(S_{i-1}, D_{i-2}) + Y_{i-1} + Z_{i-1}$. The event is thus equivalent to $Y_{i-1} + Z_{i-1} = k$ for some $k \le x$, i.e., the events described in (11). Under these events, the scheme will switch to the new policy ϕ_{a}. Because $A_{i-1} = \max(S_{i-1}, D_{i-2}) + k$, we have $\bar{a} = A_{i-1} - S_{i-1} = \bar{d} + k$, which is why we use $f_{\mathsf{a}}(\bar{a} + k)$ in (11) as the next state value.

The terms in (12) and (13) consider the event that s sends P_i at time $\max(S_{i-1}, D_{i-2}) + x$ before the arrival of Ack_{i-1}. The AoI cost in (12) is $\gamma(\bar{d} + x, m^+_Y(x))$. Comparing it to (10), the difference is that *without the arrival of* Ack_{i-1}*, when sending* P_i *at time* $\max(S_{i-1}, D_{i-2}) + x$*, source* s *cannot be 100% certain that the forward queue is empty. There is a chance*

that P_{i-1} may "block" P_i and the expected delay of P_i is thus enlarged from $\mathbb{E}(Y)$ to $m_Y^+(x)$ in (8). Therefore we use $m_Y^+(x)$ in the AoI cost term $\gamma(\cdot)$ of (12).

The term "$-\mathsf{v}\cdot(\bar{d}+x)$" in (12) is again the average-cost adjustment term for *ACPS-semi-MDP*. (13) computes the next state values. Specifically, when transmitting the next packet P_{i+1}, the new state value $\bar{d}$ for P_{i+1} becomes

$$\begin{aligned}\bar{d} &= (D_{i-1}-S_i)^+ \\ &= \big((\max(S_{i-1},D_{i-2})+Y_{i-1})-(\max(S_{i-1},D_{i-2})+x)\big)^+\end{aligned}$$

After simplification, we have $\bar{d}=(Y_{i-1}-x)^+$ and thus the next state values specified in (13). The argmin x^* value in (11)–(13) gives us the optimal waiting time function $\phi_{\mathsf{na}}(\bar{d})$.

We use *value iteration* to find a scalar v and functions $f_{\mathsf{a}}(\bar{a})$ and $f_{\mathsf{na}}(\bar{d})$ that satisfy (10)–(13) with the ground state value hardwired to $f_{\mathsf{na}}(0)=0$. The final v value is the optimal AoI of the genie-aided scheme, thus a new lower bound lb_{new}.

IV. Main Result #2: A New Upper Bound

Our upper bound is derived by adding the constraint:

$$S_i \geq A_{i-2}, \quad \forall i \geq 1. \tag{14}$$

to (4)–(6) and then characterizing the optimal value of the restricted problem. Compared to the BAA scheme in Sec. II-A, our scheme can send P_i *before* A_{i-1} if desired, but must be *after* A_{i-2}. We argue that the optimal scheme of the restricted problem (4)–(6) plus (14) must respect the following rules:

Rule 1: During time $t\in(S_{i-1},\max(S_{i-1},A_{i-2})]$, by (14) source s waits and must not send the current packet P_i.

Consider two "waiting time functions" $\theta_{\mathsf{na}}:[0,D]^2\mapsto[0,B]$ and $\theta_{\mathsf{a}}:[1,D]\mapsto[0,B]$. Both the domains and ranges are finite with D and B being explicit functions of $y_{\max}$ and $z_{\max}$. We omit their expressions due to space constraints.

Recall that we use $\bar{a}\triangleq A_{i-1}-S_{i-1}$ as shorthand, see the discussion after (9). Define two additional ones:

$$\tilde{a}\triangleq(A_{i-2}-S_{i-1})^+ \text{ and } \quad \tilde{d}\triangleq\tilde{a}-(D_{i-2}-S_{i-1})^+. \tag{15}$$

Rule 2: At time $t=\max(S_{i-1},A_{i-2})$, source s computes $x^*_{\mathsf{na}}\triangleq\theta_{\mathsf{na}}\left(\tilde{a},\tilde{d}\right)$. If Ack_{i-1} has not arrived by time $\max(S_{i-1},A_{i-2})+x^*_{\mathsf{na}}$, then s will send P_i at that time, i.e., x^*_{na} is the additional waiting time after $\max(S_{i-1},A_{i-2})$ if Ack_{i-1} has not arrived by then.

Rule 3: If Ack_{i-1} has arrived at an earlier time than $\max(S_{i-1},A_{i-2})+x^*_{\mathsf{na}}$, then at time $t=A_{i-1}$, s computes $x^*_{\mathsf{a}}\triangleq\theta_{\mathsf{a}}(\bar{a})$ and will send P_i at time $A_{i-1}+x^*_{\mathsf{a}}$.

Rules 1 to 3 have the same structure as the genie-aided scheme in Sec. III. The main difference lies in Rule 2, for which the state value now consists of a pair $(\tilde{a},\tilde{d})$. The $\tilde{a}$ value is how much the AoI has grown at time $\max(S_{i-1},A_{i-2})$, an important piece of information when minimizing AoI.

Note that at time $t=\max(S_{i-1},A_{i-2})$, using Ack_{i-2}, source s knows with 100% certainty the value of D_{i-2}, the time when P_{i-2} left the forward queue. Therefore, the past $\tilde{a}$ slots (counted from the injection of P_{i-1} to the current time $\max(S_{i-1},A_{i-2})$) can be divided into two segments: Segment 1: The first $(D_{i-2}-S_{i-1})^+$ slots during which the forward queue was still busy processing P_{i-2} and thus *cannot* process P_{i-1}; and Segment 2: The remaining $\tilde{d}$ slots, see definition (15), during which the forward queue started to serve P_{i-1}.

As a result, *the longer the Segment 2 is, the more time the forward queue has devoted to serving P_{i-1}, the more likely that P_{i-1} has been delivered to d (though no arrival of Ack_{i-1} yet), and the more likely that new packet P_i will face an empty queue and thus a shorter delay.* The value $\tilde{d}$ is thus another critical information for s when deciding the send time S_i. Some careful analysis shows that the $(\tilde{a},\tilde{d})$ pair is indeed the necessary and sufficient state values for the semi-MDP problem. That is why we have $\theta_{\mathsf{na}}(\tilde{a},\tilde{d})$ in Rule 2.

Define the following function:

$$\tilde{m}_Y^+(x,\delta)\triangleq\mathbb{E}(Y)+\mathbb{E}\left((Y-x-\delta)^+|(Y-\delta)^++Z>x\right) \tag{16}$$

Using Rules 1 to 3, the Bellman equations can be written as follows. $\forall\bar{a}\in[1,D]$ we have

$$g_{\mathsf{a}}(\bar{a})=\min_{x\in[0,B]}\gamma(\bar{a}+x,\mathbb{E}(Y))-\mathsf{v}\cdot(\bar{a}+x)+g_{\mathsf{na}}(0,0) \tag{17}$$

and $\forall(\tilde{a},\tilde{d})\in[0,D]^2$ we have

$$g_{\mathsf{na}}(\tilde{a},\tilde{d})=\min_{x\in[0,B]}\Bigg\{\sum_{k=0}^{x}\mathbb{P}\Big((Y-\tilde{d})^++Z=k\Big)\cdot g_{\mathsf{a}}(\tilde{a}+k) \tag{18}$$

$$+\mathbb{P}\left((Y-\tilde{d})^++Z>x\right)\cdot \left(\gamma\left(\tilde{a}+x,\tilde{m}_Y^+(x,\tilde{d})\right)-\mathsf{v}\cdot(\tilde{a}+x)\right. \tag{19}$$

$$\left.+\sum_{y,z}\mathbb{P}\left(Y=y,Z=z\big|(Y-\tilde{d})^++Z>x\right)\cdot g_{\mathsf{na}}\left(\tilde{a}_{\text{next}},\tilde{d}_{\text{next}}\right)\right)\Bigg\} \tag{20}$$

where $\tilde{a}_{\text{next}}\triangleq\left((y-\tilde{d})^++z-x\right)^+$ and $\tilde{d}_{\text{next}}\triangleq\tilde{a}_{\text{next}}-(y-\tilde{d}-x)^+$. Eq. (17) is similar to (10), consisting of the AoI cost term $\gamma(\tilde{a}+x,\mathbb{E}(Y))$, the ACPS adjustment term $-\mathsf{v}(\tilde{a}+x)$, and the next state value $g_{\mathsf{na}}(0,0)$, which is derived by noticing that since $S_i\geq A_{i-1}$ in Rule 3, we have $\tilde{a}=(A_{i-1}-S_i)^+=0=\tilde{d}=(D_{i-1}-S_i)^+$ for the next packet P_{i+1}.

Eqs. (18)–(20) follow the same structure as in (11)–(13). Specifically, (18) depicts the event that Ack_{i-1} arrives before the send time decision $\tilde{a}+x$. Eqs. (19)–(20) describe the event that Ack_{i-1} arrives after time $\tilde{a}+x$. In particular, the AoI cost term in (19) uses $\tilde{m}_Y^+(x,\tilde{d})$ in (16), the *enlarged* expected delay of P_i passing through the forward queue due to the possibility being blocked by P_{i-1}. Eq. (20) analyzes the next states when transmitting the next packet P_{i+1}. The derivation is very similar to that of (11)–(13) and we thus omit the details.

We use value iteration to find a scalar v and functions $g_{\mathsf{a}}(\bar{a})$ and $g_{\mathsf{na}}(\tilde{a},\tilde{d})$ that satisfy (17)–(20) and $g_{\mathsf{na}}(0,0)=0$. The final v value is the optimal AoI of the new achievability scheme, which we denote by ub_{new}. The argmin x^* values in

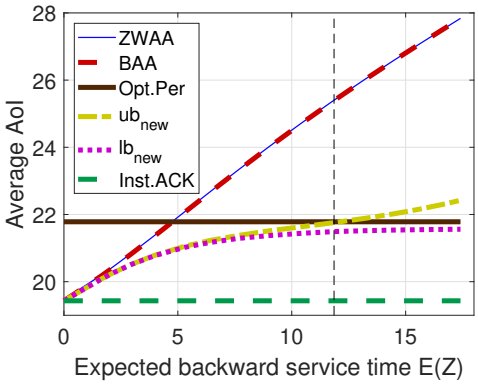

Fig. 2. $(\mathsf{lb}_{\mathsf{new}}, \mathsf{ub}_{\mathsf{new}})$ versus existing results — log-normal $\mathbb{P}_Y$

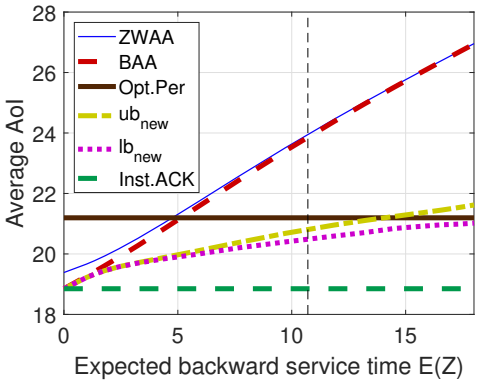

Fig. 3. $(\mathsf{lb}_{\mathsf{new}}, \mathsf{ub}_{\mathsf{new}})$ versus existing results — composite log-normal $\mathbb{P}_Y$.

(17)–(20) give the optimal waiting time functions $\theta_{\mathsf{a}}(\bar{a})$ and $\theta_{\mathsf{na}}(\tilde{a}, \tilde{d})$. Once $\theta_{\mathsf{a}}(\bar{a})$ and $\theta_{\mathsf{na}}(\tilde{a}, \tilde{d})$ are computed, the scheme can be easily implemented following Rules 1 to 3. As will be seen later, our achievability scheme exhibits near-optimal performance and could have significant impact in practice.

V. Numerical Evaluation

For any given $[M_L, M_U]$, μ, and σ^2 values, we say a random variable Q is integer-quantized, $[M_L, M_U]$-truncated, log-normal with parameters (μ, σ^2) if $\forall q \in [M_L, M_U]$,

$$\mathbb{P}(Q = q) \propto \mathbb{P}(W \in (q - 0.5, q + 0.5])$$

where W is log-normal with parameters (μ, σ^2). That is, we first truncate the values outside $[M_L, M_U]$ and then proportionally scale it so that the total probability is 1.

Fig. 2 plots $(\mathsf{lb}_{\mathsf{new}}, \mathsf{ub}_{\mathsf{new}})$ versus existing bounds zwaa, baa, opt.per, and inst.ack, for which we assume Y_i (resp. Z_i) is integer-quantized, $[1, 24]$-truncated (resp. $[0, 24]$-truncated), log-normal with parameters (μ_Y, σ_Y^2) (resp. (μ_Z, σ_Z^2)). The truncation intervals are slightly different since we assume $Y_i \geq 1$ and $Z_i \geq 0$ in our setting, see Sec. II. We fix $(\mu_Y, \sigma_Y^2) = (2.5, 0.6^2)$ and set $\sigma_Z^2 = 0.6^2$ while varying the values of μ_Z to change the expected backward delay. A thin vertical line $\mathbb{E}(Y) = 11.86$ is drawn in Fig. 2 to indicate the average service time of the *forward queue*.

As can be seen, none of existing upper bounds zwaa, baa, opt.per and lower bound inst.ack is tight for the general cases, while $\mathsf{lb}_{\mathsf{new}}$ and $\mathsf{ub}_{\mathsf{new}}$ closely follow each other. In fact, the smaller the $\mathbb{E}(Z)$, the smaller the gap ratio $\frac{\mathsf{ub}_{\mathsf{new}} - \mathsf{lb}_{\mathsf{new}}}{\mathsf{lb}_{\mathsf{new}}}$. Specifically, it is less than 0.28% when $\mathbb{E}(Z) \leq 6.40$ and it grows to 1.24% when $\mathbb{E}(Z) = \mathbb{E}(Y) = 11.86$. The bounds do diverge for $\mathbb{E}(Z) \geq \mathbb{E}(Y)$. Those situations are less interesting in practice since each feedback usually consists of a small(er) packet that experiences shorter delay than the forward traffic. If desired, we can sharpen the upper bound by $\overline{\mathsf{ub}}_{\mathsf{new}} \triangleq \min(\mathsf{ub}_{\mathsf{new}}, \mathsf{opt.per})$. The gap ratio $\frac{\overline{\mathsf{ub}}_{\mathsf{new}} - \mathsf{lb}_{\mathsf{new}}}{\mathsf{lb}_{\mathsf{new}}}$ is less than 1.36% for all $\mathbb{E}(Z)$. The pair $(\mathsf{lb}_{\mathsf{new}}, \overline{\mathsf{ub}}_{\mathsf{new}})$ thus tightly brackets the true avg.aoi* of (4)–(6) for all scenarios.

The very tight performance is because *our achievability scheme uses the conditional probabilities in (19) and (20) to accurately represent "the information when* Ack_{i-1} *has not arrived". Therefore, the scheme can send the current packet* P_i *at the best possible time even before* s *has received* Ack_{i-1}.

Note that the gap between opt.per versus $\mathsf{lb}_{\mathsf{new}}$ (or $\mathsf{ub}_{\mathsf{new}}$) diminishes gradually as the feedback delay $\mathbb{E}(Z)$ grows. When $\mathbb{E}(Z) = \mathbb{E}(Y) = 11.86$, the gap ratio between $\mathsf{lb}_{\mathsf{new}}$ and opt.per has diminished to 1.36%. That is, even with the best possible closed-loop design, one can improve upon the simpler Opt.Per scheme by at most 1.36%. Fig. 2 provides a useful guideline on when one should consider switching to open loop designs under delayed feedback scenarios.

Fig. 3 repeats the same experiment except that we let $\mathbb{P}_Y$ be a $(0.5, 0.5)$ mixture of two integer-quantized $[1, 24]$-truncated log-normals with parameters $(\mu_{Y_1}, \sigma_{Y_1}^2) = (2.9, 0.2^2)$ and $(\mu_{Y_2}, \sigma_{Y_2}^2) = (1.0, 0.7^2)$, respectively. That is, $\mathbb{P}_Y$ is bimodal composite-log-normal. The thin vertical line indicates the new $\mathbb{E}(Y) = 10.71$. The gap ratio between $(\mathsf{lb}_{\mathsf{new}}, \mathsf{ub}_{\mathsf{new}})$ is less than 0.54% when $\mathbb{E}(Z) \leq 5.83$ and grows to 1.6% when $\mathbb{E}(Z) = \mathbb{E}(Y) = 10.71$. The largest gap ratio between $\mathsf{lb}_{\mathsf{new}}$ and $\overline{\mathsf{ub}}_{\mathsf{new}} \triangleq \min(\mathsf{ub}_{\mathsf{new}}, \mathsf{opt.per})$ is 1.9% for all $\mathbb{E}(Z)$.

Under the instantaneous ACK setting, the gap between zwaa and baa is larger if $\mathbb{P}(Y)$ happens to be bimodal, see the diverging gap between zwaa and baa in Fig. 3 when $\mathbb{E}(Z) = 0$. This is also why we are interested in bimodal $\mathbb{P}_Y$ in Fig. 3. Nonetheless, the gap between zwaa and baa diminishes quickly with feedback delay. I.e., there is little room for AoI improvement if we always wait for Ack_{i-1} before sending P_i. In contrast, the gap between zwaa and $\mathsf{ub}_{\mathsf{new}}$ continues to widen when $\mathbb{E}(Z)$ grows. Namely, the AoI improvement of our new achievability scheme over the naive zero-wait policy keeps getting bigger since our new scheme uses the delayed feedback in a probabilistically near-optimal way.

Jointly, Figs. 2 and 3 show that our bounds are numerically tight for two very different distributions, e.g., unimodal versus bimodal. In our other not-reported experiments, the tightness persists for uniform and geometric delay distributions as well.

VI. Conclusion

We have studied the AoI minimization problem with queues in both the forward/feedback directions. Near-optimal schedulers have been devised. The results have quantified the usefulness of delayed feedback in queue-based AoI minimization.

REFERENCES

[1] J. Zhang and C.-C. Wang, "On the rate-cost of Gaussian linear control systems with random communication delays," in *Proc. IEEE Int'l Symp. Inform. Theory*. Vail, USA, June 2018.

[2] J. Doyle, "Guaranteed margins for LQG regulators," *IEEE Trans. Autom. Control*, vol. 23, no. 4, pp. 756–757, August 1978.

[3] Y. Sun, Y. Polyanskiy, and E. Uysal, "Sampling of the Wiener process for remote estimation over a channel with random delay," *IEEE Trans. Inf. Theory*, vol. 66, no. 2, pp. 1118–1135, February 2019.

[4] T. Ornee and Y. Sun, "Sampling and remote estimation for the Ornstein-Uhlenbeck process through queues: Age of information and beyond," *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 1962–1975, October 2021.

[5] Y. Sun and B. Cyr, "Sampling for data freshness optimization: Non-linear age functions," *J. Commun. Networks*, vol. 21, no. 3, pp. 204–219, June 2019.

[6] L. Huang and E. Modiano, "Optimizing age-of-information in a multi-class queueing system," in *Proc. IEEE Int'l Symp. Inform. Theory*. Hong Kong, China, June 2015.

[7] V. Tripathi, R. Talak, and E. Modiano, "Age of information for discrete time queues," in *arXiv:1901.10463*, 2019.

[8] Y. Sun, E. Uysal-Biyikoglu, R. Yates, C. Koksal, and N. Shroff, "Update or wait: How to keep your data fresh," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7492–7508, December 2017.

[9] C.-H. Tsai and C.-C. Wang, "Unifying AoI minimization and remote estimation — optimal sensor/controller coordination with random two-way delay," in *Proc. 39th IEEE Conference on Computer Communications (INFOCOM)*. Toronto, Canada, July 2020, 10 pages.

[10] ——, "Age-of-information revisited: Two-way delay and distribution-oblivious online algorithm," in *Proc. IEEE Int'l Symp. Inform. Theory*. Los Angeles, USA, June 2020, 6 pages.

[11] ——, "Unifying AoI minimization and remote estimation – optimal sensor/controller coordination with random two-way delay," *IEEE/ACM Trans. Netw.*, September 2021, https://doi.org/10.1109/TNET.2021.3111495.

[12] ——, "Jointly minimizing AoI penalty and network cost among coexisting source-destination pairs," in *Proc. IEEE Int'l Symp. Inform. Theory*. Melbourne, Australia, July 2021, 6 pages.

[13] R. Yates and S. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1807–1826, March 2019.

[14] S. Kaul, R. Yates, and M. Gruteser, "Status updates through queues," in *Proc. 46th Conf. Inform. Sciences and Systems*. Princeton, NJ, USA, March 2012.

[15] C. Kam, S. Kompella, and A. Ephremides, "Learning to sample a signal through an unknown system for minimum AoI," in *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019.

[16] A. Bedewy, Y.Sun, and N. Shroff, "Minimizing the age of information through queues," *IEEE Trans. Inf. Theory*, vol. 65, no. 8, pp. 5215–5232, August 2019.

[17] A. Bedewy, Y.Sun, S. Kompella, and N. Shroff, "Optimal sampling and scheduling for timely status updates in multi-source networks," *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 4019–4034, June 2021.

[18] D. Bertsekas, *Dynamic Programming and Optimal Control*, 4th ed. New Hampshire, USA: Athena Scientific, 2017.