Waiting but not Aging: Optimizing Information Freshness Under the Pull Model

Fengjiao Li, *Student Member, IEEE,* Yu Sang, Zhongdong Liu, *Student Member, IEEE,* Bin Li, *Senior Member, IEEE,* Huasen Wu, *Member, IEEE,* and Bo Ji, *Senior Member, IEEE*

Abstract—The Age-of-Information is an important metric for investigating the timeliness performance in information-update systems. In this paper, we study the AoI minimization problem under a new Pull model with replication schemes, where a user proactively sends a replicated request to multiple servers to "pull" the information of interest.Interestingly, we find that under this new Pull model, replication schemes capture a novel tradeoff between different values of the AoI across the servers (due to the random updating processes) and different response times across the servers, which can be exploited to minimize the expected AoI at the user's side. Specifically, assuming Poisson updating process for the servers and exponentially distributed response time, we derive a closed-form formula for computing the expected AoI and obtain the optimal number of responses to wait for to minimize the expected AoI. Then, we extend our analysis to the setting where the user aims to maximize the AoI-based utility, which represents the user's satisfaction level with respect to freshness of the received information. Furthermore, we consider a more realistic scenario where the user has no prior knowledge of the system. In this case, we reformulate the utility maximization problem as a stochastic Multi-Armed Bandit problem with side observations and leverage a special linear structure of side observations to design learning algorithms with improved performance guarantees. Finally, we conduct extensive simulations to elucidate our theoretical results and compare the performance of different algorithms. Our findings reveal that under the Pull model, waiting does not necessarily lead to aging; waiting for more than one response can often significantly reduce the AoI and improve the AoI-based utility in most scenarios.

Index Terms—Age-of-Information, pull model, replication, utility maximization, Multi-armed bandit, learning.

I. INTRODUCTION

The last decades have witnessed the prevalence of smart devices and significant advances in ubiquitous computing and the Internet of things. This trend is forecast to continue in the years to come [2]. The development of this trend has spawned a plethora of real-time services that require timely information/status updates. One practically important example of such services is vehicular networks and intelligent transportation systems [3], [4], where accurate status information (position, speed, acceleration, tire pressure, etc.) of a vehicle needs to be shared with other nearby vehicles and road-side facilities in a timely manner in order to avoid collisions

and ensure substantially improved road safety. More such examples include sensor networks for environmental/health monitoring [5], [6], wireless channel feedback [7], news feeds, weather updates, online social networks, flight aggregators (e.g., Google Flights), and stock quote services.

For systems providing such real-time services, those commonly used performance metrics, such as throughput and delay, exhibit significant limitations in measuring the system performance [8]. Instead, *the timeliness of information updates becomes a major concern*. To that end, a new metric called the *Age-of-Information (AoI)* has been proposed as an important metric for studying the timeliness performance [3]. The AoI is defined as the time elapsed since the most recent update occurred (see Eq. (1) for a formal definition). Using this new AoI metric, the work of [8] employs a simple system model to analyze and optimize the timeliness performance of an information-update system. This seminal work has recently aroused dramatic interests from the research community and has inspired a series of interesting studies on AoI analysis and optimization (see [9], [10] and references therein).

While all prior studies consider a *Push* model, concerning about when and how to "push" (i.e., generate and transmit) the updated information to the user, in this paper we introduce a new *Pull* model, under which a user sends requests to the servers to proactively "pull" the information of interest. This Pull model is more relevant for many important applications where the user's interest is in the freshness of information at the point when the user requests it rather than in continuously monitoring the freshness of information. One application of the Pull model is in the real-time stock quote service, where a customer (i.e., user) submits a query to multiple stock quote providers (i.e., servers) sharing common information sources and each provider responds with its most up-to-date information. Other applications include flight aggregators and real estate listings apps.

To the best of our knowledge, however, none of the existing work on the timeliness optimization has considered such a Pull model. In stark contrast, we focus on the Pull model and propose to employ request replication to minimize the AoI or to maximize the AoI-based utility at the user's side. Although a similar Pull model is considered for data synchronization in [11], [12], the problems are quite different and request replication is not exploited. Note that the concept of replication is not new and has been extensively studied for various applications (e.g., cloud computing and datacenters [13], [14], storage clouds [15], parallel computing [16], [17], and databases [18], [19]). *However, for the AoI minimization problem under the*

Pull model, replication schemes exhibit a unique property and capture a novel tradeoff between different levels of information freshness and different response times across the servers. This tradeoff reveals the power of waiting for more than one response and can be exploited to optimize information freshness at the user's side.

Next, we explain the above key tradeoff through a comparison with cloud computing systems. It has been observed that in a cloud or a datacenter, the processing time of a same job can be highly variable on different servers [14]. Due to this important fact, replicating a job on multiple servers and waiting for the first finished copy can help reduce the latency [13], [14]. Apparently, in such a system it is *not* beneficial to wait for more copies of the job to finish, as all the copies would give the same outcome. By contrast, in the information-update system we consider, although the servers may possess the same type of information (weather forecast, stock prices, etc.), they could have different versions of the information with different levels of freshness due to the random updating processes. In fact, the first response may come from a server with stale information; waiting for more than one response has the potential of receiving fresher information and thus helps reduce the AoI. *Hence, it is no longer the best to stop waiting after receiving the first response (as in the other aforementioned applications).* On the other hand, waiting for too many responses will lead to a longer total waiting time, and thus, it also incurs a larger AoI at the user's side. *Therefore, it is challenging to determine the optimal number of responses to wait for in order to minimize the expected AoI (or to maximize the AoI-based utility) at the user's side.* The problem is further exacerbated by the fact that the updating rate and the mean response time, which are important to making such decisions, are typically unknown to the user a priori.

We summarize our key contributions as follows.

- To the best of our knowledge, this work, for the first time, introduces the Pull model for studying the timeliness optimization problem and proposes to employ request replication to reduce the AoI.
- Assuming Poisson updating process at the servers and exponentially distributed response time, we derive a closed-form formula for computing the expected AoI and obtain the optimal number of responses to wait for to minimize the expected AoI. We also discuss some extensions to account for more general replication schemes and different types of response time distributions.
- We further consider scenarios where the user aims to maximize the utility, which is an exponential function of the negative AoI. The utility represents the user's satisfaction level with respect to freshness of the received information. We derive a set of similar theoretical results for the utility maximization problem.
- Moreover, we consider a more realistic scenario where the user has no prior knowledge of the system parameters such as the updating rate the and the mean response time. In this case, we formulate the utility maximization problem as a stochastic Multi-Armed Bandit (MAB) problem with side observations. The side observations lead to the feedback graph with a special linear structure,

which can be leveraged to design learning algorithms with improved regret upper bounds.
- Finally, we conduct extensive simulations to elucidate our theoretical results. We also investigate the impact of the system parameters on the achieved gain. Our findings reveal that under the Pull model, waiting does not necessarily lead to aging; waiting for more than one response can often significantly reduce the AoI and improve the AoI-based utility in most scenarios. In the case of unknown system parameters, our simulation results show that algorithms exploiting the special linear feedback graph outperform the classic learning algorithms.

The remainder of this paper is organized as follows. We first discuss related work in Section II and then describe our new Pull model in Section III. In Section IV, we analyze the expected AoI under replication schemes and obtain the optimal number of responses for minimizing the expected AoI. In Section V, we consider the utility maximization problem in the settings where the updating rate and the mean response time are known and unknown, respectively. Section VI presents the simulation results, and we conclude the paper in Section VII.

II. RELATED WORK

Since the seminal work on AoI [3], there has been a large body of work focusing on AoI analysis and optimization in a wide variety of settings and applications (see [9], [10] for surveys). However, almost all prior work considers the Push model, in contrast to the Pull model we consider in this paper.

A series of work (e.g., [8], [20]–[27]) has been focused on analyzing the AoI performance of various queueing models. In [8], the authors analyze the expected AoI in M/M/1, M/D/1, and D/M/1 systems under the First-Come-First-Served (FCFS) policy. A follow-up work in [22] extends the analysis to M/M/2 and M/M/∞ models. The expected AoI is also characterized for M/M/1 Last-Come-First-Served (LCFS) model, with and without preemption, for single-source and multi-source systems [20], [21]. Furthermore, controlling the AoI through packet deadlines is studied in [23], [24]; the effect of the packet management (e.g., prioritizing new arrivals and discarding old packets) on the AoI is considered in [25], [26]. In [27], the authors show that the preemptive Last-Generated-First-Served (LGFS) policy achieves the optimal (or near-optimal) AoI performance in a multi-server queueing system.

There have also been lots of recent efforts denoted to the design and analysis of AoI-oriented scheduling algorithms in various network settings (e.g., [28]–[36]). In [28], the authors aim to minimize the weighted sum AoI of the clients in a broadcast wireless network with unreliable channels. A similar problem with throughput constraints is considered in a follow-up study [29]. In [30], [31], [33], the authors consider AoI-optimal scheduling problems in ad hoc wireless networks under interference constraints. Considering a similar network setting, the authors of [32] aim to design AoI-aware algorithms for scheduling real-time traffic with hard deadlines. Recently, the study on AoI has also been pushed towards more challenging settings with multi-hop flows [34]–[36].

We want to point out that the preliminary version of our paper [1] is the first work that employs the Pull model and

replication schemes to study the AoI at the user's side. Since then, the idea of replication has also been adopted for studying the AoI under different models (see, e.g., [27], [37], [38]). Recently, the authors of [39] also aim to minimize the AoI from the users' perspective by considering multiple users. Note that outside the AoI area, similar Pull models have been investigated (e.g., for data synchronization [11], [12]) since decades ago. However, the problems they study are very different, and request replication is not exploited.

Besides the linear AoI considered in the above work, there are several studies that investigate more general functions of the AoI (e.g., [40]–[42]). Such functions are often used to model utility/penalty, which represents the user's satisfaction/dissatisfaction level with respect to freshness of the received information. In this extended journal version (see Section V), we also consider the AoI-based utility, which is an exponential function of the negative AoI. However, the model and the problem we consider are quite different from those in the existing work. Furthermore, we study the scenario where the system parameters are unknown and cast the AoI-based utility maximization problem as an online learning problem based on the stochastic MAB formulation.

Although variants of MAB formulations have recently been considered for AoI minimization problems (see, e.g., [43]–[45]), they consider Markovian MAB, where the state of each arm evolves in a Markovian fashion and the reward drawn at each time is a function of the current state of the selected arm. This is very different from the stochastic MAB we consider, in terms of model, algorithm design, and regret analysis. Moreover, we consider a new Pull model and exploit a special linear structure of the feedback graph to design learning algorithms with improved regret upper bounds.

III. SYSTEM MODEL

We consider an information-update system where a user pulls time-sensitive information from n servers. These n servers are connected to a common information source and update their data *asynchronously*. We call such a model the *Pull* model (see Fig. 1). Let $\mathcal{N} \triangleq \{1, 2, \ldots, n\}$ be the set of indices of the servers, and let $i \in \mathcal{N}$ be the server index. We assume that the information updates at the source for each server follow a Poisson process with rate $\lambda > 0$ (where the update rate can model system limitations or resource budgets). The updating processes are asynchronous and are assumed to be independent and identically distributed (*i.i.d.*) across the servers. We also assume that there is no transmission delay from the source to the servers. That is, servers instantaneously receive their updates once the updates are generated at the source. This implies that the inter-update time (i.e., the time duration between two successive updates) at each server follows an exponential distribution with mean $1/\lambda$. Let $u_i(t)$ denote the time when the most recent update at server i occurs, and let $\Delta_i(t)$ denote the AoI at server i, which is defined as the time elapsed since the most recent update at this server:

$$\Delta_i(t) \triangleq t - u_i(t). \tag{1}$$

Therefore, the AoI at a server drops to zero if an update occurs at this server; otherwise, the AoI increases linearly as
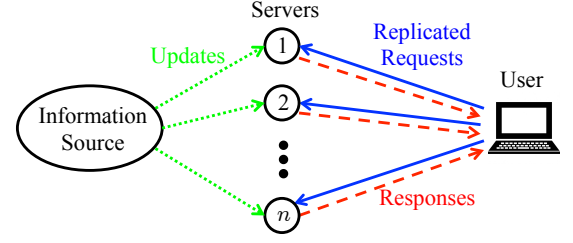

Fig. 1: The Pull model of information-update systems. Note that the arrows in the figure denote logical links rather than physical connections. The updates, requests, and responses are all transmitted through (wired or wireless) networks.
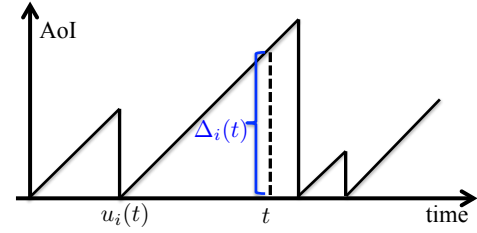

Fig. 2: An illustration of the AoI evolution at server i

time goes by until the next update occurs. Fig. 2 provides an illustration of the AoI evolution at server i.

In this work, we consider the (n, k) replication scheme, under which the user sends the replicated copies of the request to all n servers and waits for the first k responses. Let R_i denote the response time for server i. Note that each server may have a different response time, which is the time elapsed since the request is sent out by the user until the user receives the response from this server. We assume that the time for the requests to reach the servers is negligible compared to the time for the user to download the data from the servers. Hence, the response time can be interpreted as the downloading time. Let s denote the downloading start time, which is the same for all the servers, and let f_i denote the downloading finish time for server i. Then, the response time for server i is $R_i = f_i - s$. We assume that the response time is exponentially distributed with mean $1/\nu$ and is *i.i.d.* across the servers. Note that the model we consider above is simple, but it suffices to capture the key aspects and novelty of the problem we study.

Under the (n, k) replication scheme, when the user receives the first k responses, it uses the freshest information among these k responses to make certain decisions (e.g., stock trading decisions based on the received stock price information). Let (j) denote the index of the server corresponding to the j-th response received by the user. Then, set $\mathcal{K} \triangleq \{(1), (2), \ldots, (k)\}$ contains the indices of the servers that return the first k responses, and the following is satisfied: $f_{(1)} \leq f_{(2)} \leq \cdots \leq f_{(k)}$ and $R_{(1)} \leq R_{(2)} \leq \cdots \leq R_{(k)}$. Let server i^* be the index of the server that provides the freshest information (i.e., that has the smallest AoI) among these k responses when downloading starts at time s, i.e.,

$$\Delta_{i^*}(s) = \min_{i \in \mathcal{K}} \Delta_i(s). \tag{2}$$

Here, we are interested in the AoI at the user's side when it receives the k-th response, denoted by $\Delta(k)$, which is the time difference between when the k-th response is received and when the information at server i^* is updated, i.e.,

$$\Delta(k) \triangleq f_{(k)} - u_{i^*}(s). \tag{3}$$

Then, there are two natural questions of interest:

(Q1): For a given k, can one obtain a closed-form formula for computing the expected AoI at the user's side, $\mathbb{E}[\Delta(k)]$?
(Q2): How to determine the optimal number of responses to wait for, such that $\mathbb{E}[\Delta(k)]$ is minimized?

The second question can be formulated as the following optimization problem:

$$\min_{k \in \mathcal{N}} \mathbb{E}\left[\Delta(k)\right]. \tag{4}$$

We will answer these two questions in Section IV.

Furthermore, we will generalize the proposed framework and consider maximizing an AoI-based utility function at the user's side. The utility maximization problem will be studied in Section V, where we consider both cases of known and unknown system parameters (i.e., the updating rate and the mean response time).

IV. AoI Minimization

In this section, we focus on the AoI minimization problem under the Pull model. We first derive a closed-form formula for computing the expected AoI at the user's side under the (n, k) replication scheme (Section IV-A). Then, we find the optimal number of responses to wait for in order to minimize the expected AoI (Section IV-B). Finally, we discuss some immediate extensions (Section IV-C).

A. Expected AoI

In this subsection, we focus on answering Question (Q1) and derive a closed-form formula for computing the expected AoI under the (n, k) replication scheme.

To begin with, we provide a useful expression of the AoI at the user's side under the (n, k) replication scheme (i.e., $\Delta(k)$, as defined in Eq. (3)) as follows:

$$
\begin{aligned}
\Delta(k) &= f_{(k)} - u_{i^*}(s) \\
&= f_{(k)} - s + s - u_{i^*}(s) \\
&= R_{(k)} + \Delta_{i^*}(s) \\
&= R_{(k)} + \min_{i \in \mathcal{K}} \Delta_i(s),
\end{aligned} \tag{5}
$$

where the second last equality is from the definition of R_i and $\Delta_i(t)$ (i.e., Eq. (1)), and the last equality is from Eq. (2). As can be seen from the above expression, under the (n, k) replication scheme the AoI at the user's side consists of two terms: (i) $R_{(k)}$, the total waiting time for receiving the first k responses, and (ii) $\min_{i \in \mathcal{K}} \Delta_i(s)$ (also denoted by $\Delta_{i^*}(s)$), the AoI of the freshest information among these k responses when downloading starts at time s. An illustration of these two terms and $\Delta(k)$ is shown in Fig. 3.

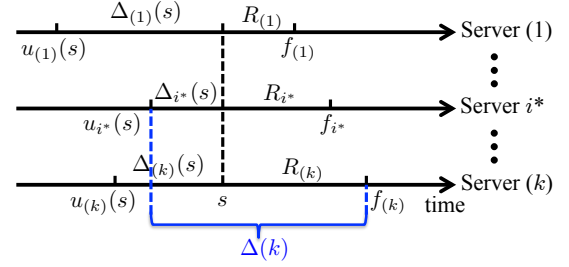Taking the expectation of both sides of Eq. (5), we have

Fig. 3: An illustration of the AoI at the user's side and its two terms under the (n, k) replication scheme

$$\mathbb{E}[\Delta(k)] = \mathbb{E}\left[R_{(k)}\right] + \mathbb{E}\left[\min_{i \in \mathcal{K}} \Delta_i(s)\right]. \tag{6}$$

In the above equation, the first term (i.e., the expected total waiting time) can be viewed as the cost of waiting, while the second term (i.e., the expected AoI of the freshest information among these k responses) can be viewed as the benefit of waiting. Intuitively, as k increases (i.e., waiting for more responses), the expected total waiting time (i.e., the first term) increases. On the other hand, upon receiving more responses, the expected AoI of the freshest information among these k responses (i.e., the second term) decreases. Hence, there is a natural tradeoff between these two terms, which is a unique property of our newly introduced Pull model.

Next, we formalize this tradeoff by deriving the closed-form expressions of the above two terms as well as the expected AoI. We state the main result of this subsection in Theorem 1.

Theorem 1. Under the (n, k) replication scheme, the expected AoI at the user's side can be expressed as

$$\mathbb{E}[\Delta(k)] = \frac{1}{\nu}(\mathbf{H}(n) - \mathbf{H}(n - k)) + \frac{1}{k\lambda}, \tag{7}$$

where $\mathbf{H}(n) = \sum_{l=1}^{n} \frac{1}{l}$ is the n-th partial sum of the diverging harmonic series.

Proof. We first analyze the first term of the right-hand side of Eq. (6) and want to show $\mathbb{E}[R_{(k)}] = \frac{1}{\nu}(\mathbf{H}(n) - \mathbf{H}(n - k))$. Note that the response time is exponentially distributed with mean $1/\nu$ and is *i.i.d.* across the servers. Hence, random variable $R_{(k)}$ is the k-th smallest value of n *i.i.d.* exponential random variables with mean $1/\nu$. The order statistics results of exponential random variables give that $R_{(j)} - R_{(j-1)}$ is an exponential random variable with mean $\frac{1}{(n+1-j)\nu}$ for any $j \in \mathcal{N}$, where we set $R_{(0)} = 0$ for ease of notation [46]. Hence, we have the following:

$$
\begin{aligned}
\mathbb{E}\left[R_{(k)}\right] &= \mathbb{E}\left[\sum_{j=1}^{k}(R_{(j)} - R_{(j-1)})\right] \\
&= \sum_{j=1}^{k} \mathbb{E}\left[R_{(j)} - R_{(j-1)}\right] \\
&= \sum_{j=1}^{k} \frac{1}{(n+1-j)\nu} \\
&= \frac{1}{\nu}(\mathbf{H}(n) - \mathbf{H}(n - k)).
\end{aligned} \tag{8}
$$

Next, we analyze the second term of the right-hand side of Eq. (6) and want to show the following:

$$\mathbb{E}\left[\min_{i\in\mathcal{K}}\Delta_i(s)\right] = \frac{1}{k\lambda}. \tag{9}$$

Note that the updating process at the source for each server is a Poisson process with rate λ and is *i.i.d.* across the servers. We also assume that there is no transmission delay from the source to the servers. Hence, the inter-update time for each server is exponentially distributed with mean $1/\lambda$. Due to the memoryless property of the exponential distribution, at any given request time s, the AoI at each server has the same distribution as the inter-update time, i.e., random variable $\Delta_i(s)$ is also exponentially distributed with mean $1/\lambda$ and is *i.i.d.* across the servers [47]. Therefore, random variable $\min_{i\in\mathcal{K}}\Delta_i(s)$ is the minimum of k *i.i.d.* exponential random variables with mean $1/\lambda$, which is also exponentially distributed with mean $\frac{1}{k\lambda}$. This implies Eq. (9).

Combining Eqs. (8) and (9), we complete the proof. □

Remark. The above analysis indeed agrees with our intuition: while the expected total waiting time for receiving the first k responses (i.e., Eq. (8)) is a monotonically increasing function of k, the expected AoI of the freshest information among these k responses (i.e., Eq. (9)) is a monotonically decreasing function of k.

B. Optimal Replication Scheme

In this subsection, we will exploit the aforementioned trade-off and focus on answering Question (Q2) that we discussed at the end of Section III. Specifically, we aim to find the optimal number of responses to wait for in order to minimize the expected AoI at the user's side.

First, due to Eq. (7), we can rewrite the optimization problem in Eq. (4) as

$$\min_{k\in\mathcal{N}} \frac{1}{\nu}(\mathbf{H}(n) - \mathbf{H}(n-k)) + \frac{1}{k\lambda}. \tag{10}$$

Let k^* be an optimal solution to Eq. (10). We state the main result of this subsection in Theorem 2.

Theorem 2. An optimal solution k^* to Problem (10) can be computed as

$$k^* = \min\left\{ \left\lceil \frac{2\nu n}{\sqrt{(\lambda+\nu)^2 + 4\lambda\nu n} + \lambda + \nu} \right\rceil, n \right\}. \tag{11}$$

Proof. We first define $D(k)$ as the difference of the expected AoI between the $(n, k+1)$ and (n, k) replication schemes, i.e., $D(k) \triangleq \mathbb{E}[\Delta(k+1)] - \mathbb{E}[\Delta(k)]$ for any $k \in \{1, 2, \ldots, n-1\}$. From Eq. (7), we have the following:

$$D(k) = \frac{1}{(n-k)\nu} - \frac{1}{k(k+1)\lambda}, \tag{12}$$

for any $k \in \{1, 2, \ldots, n-1\}$. It is easy to see that $D(k)$ is a monotonically increasing function of k.

We now extend the domain of $D(k)$ to the set of positive real numbers and want to find k' such that $D(k') = 0$. With

some standard calculations and dropping the negative solution, we derive the following:

$$k' = \frac{2\nu n}{\sqrt{(\lambda+\nu)^2 + 4\lambda\nu n} + \lambda + \nu}. \tag{13}$$

Next, we discuss two cases: (i) $k' > n - 1$ and (ii) $0 < k' \leq n - 1$.

In Case (i), we have $k' > n - 1$. This implies that $D(k) = \mathbb{E}[\Delta_i(k+1)] - \mathbb{E}[\Delta_i(k)] < 0$ for all $k \in \{1, 2, \ldots, n-1\}$, as $D(k)$ is monotonically increasing. Hence, the expected AoI, $\mathbb{E}[\Delta(k)]$, is a monotonically decreasing function for $k \in \mathcal{N}$. Therefore, $k^* = n$ must be the optimal solution.

In Case (ii), we have $0 < k' \leq n - 1$. We consider two subcases: k' is an integer in $\{1, 2, \ldots, n-1\}$ and k' is not an integer.

If k' is an integer in $\{1, 2, \ldots, n-1\}$, we have $D(k) = \mathbb{E}[\Delta(k+1)] - \mathbb{E}[\Delta(k)] \leq 0$ for $k \in \{1, 2, \ldots, k'\}$ and $D(k) = \mathbb{E}[\Delta(k+1)] - \mathbb{E}[\Delta(k)] > 0$ for $k \in \{k'+1, \ldots, n-1\}$, as $D(k)$ is monotonically increasing. Hence, the expected AoI, $\mathbb{E}[\Delta(k)]$, is first decreasing (for $k \in \{1, 2, \ldots, k'\}$) and then increasing (for $k \in \{k'+1, \ldots, n\}$). Therefore, there are two optimal solutions: $k^* = k'$ and $k^* = k' + 1$ since $\mathbb{E}[\Delta(k'+1)] = \mathbb{E}[\Delta(k')]$ (due to $D(k') = 0$).

If k' is not an integer, we have $D(k) = \mathbb{E}[\Delta(k+1)] - \mathbb{E}[\Delta(k)] < 0$ for $k \in \{1, 2, \ldots, \lfloor k' \rfloor\}$ and $D(k) = \mathbb{E}[\Delta(k+1)] - \mathbb{E}[\Delta(k)] > 0$ for $k \in \{\lceil k' \rceil, \ldots, n-1\}$, as $D(k)$ is monotonically increasing. Hence, the expected AoI, $\mathbb{E}[\Delta(k)]$, is first decreasing (for $k \in \{1, 2, \ldots, \lfloor k' \rfloor, \lceil k' \rceil\}$) and then increasing (for $k \in \{\lceil k' \rceil, \ldots, n-1\}$). Therefore, $k^* = \lceil k' \rceil$ must be the optimal solution.

Combining two subcases, we have $k^* = \lceil k' \rceil$ in Case (ii). Then, combining Cases (i) and (ii), we have $k^* = \min\{\lceil k' \rceil, n\} = \min\{\lceil \frac{2\nu n}{\sqrt{(\lambda+\nu)^2 + 4\lambda\nu n} + \lambda + \nu} \rceil, n\}$. □

Remark. There are two special cases that are of particular interest: (i) waiting for the first response only (i.e., $k^* = 1$) and (ii) waiting for all the responses (i.e., $k^* = n$). In Corollary 1, we provide a sufficient and necessary condition for each of these two special cases.

Corollary 1. (i) $k^* = 1$ is an optimal solution to Problem (10) if and only if $\lambda \geq \frac{\nu(n-1)}{2}$; (ii) $k^* = n$ is an optimal solution to Problem (10) if and only if $\lambda \leq \frac{\nu}{n(n-1)}$.

Proof. The proof follows straightforwardly from Theorem 2. A little thought gives the following: $k^* = 1$ is an optimal solution if and only if $D(1) \geq 0$. Solving $D(1) = \frac{1}{(n-1)\nu} - \frac{1}{2\lambda} \geq 0$ gives $\lambda \geq \frac{\nu(n-1)}{2}$. Similarly, $k^* = n$ is an optimal solution if and only if $D(n-1) \leq 0$. Solving $D(n-1) = \frac{1}{\nu} - \frac{1}{n(n-1)\lambda} \leq 0$ gives $\lambda \leq \frac{\nu}{n(n-1)}$. □

Remark. The above results agree well with the intuition. For a given number of servers, if the inter-update time is much smaller than the response time (i.e., $1/\lambda \ll 1/\nu$), then the difference of the freshness levels among the servers is relatively small. In this case, it is not beneficial to wait for more responses. On the other hand, if the inter-update time is much larger than the response time (i.e., $1/\lambda \gg 1/\nu$), then one server may possess much fresher information than another

server. In this case, it is worth waiting for more responses, which leads to a significant gain in the AoI reduction.

Note that Theorem 2 also implies how the optimal solution (i.e., k^*) scales as the number of servers (i.e., n) increases: when n becomes large, we have $k^* = \lceil \frac{2\nu n}{\sqrt{(\lambda+\nu)^2+4\lambda\nu n}+\lambda+\nu} \rceil = O(\sqrt{n})$.

C. Extensions

In this subsection, we discuss some immediate extensions of the considered model, including more general replication schemes and different types of response time distributions.

1) Replication schemes: So far, we have only considered the (n,k) replication scheme. One limitation of this scheme is that it requires the user to send a replicated request to every server, which may incur a large overhead when there are a large number of servers (i.e., when n is large). Instead, a more practical scheme would be to send the replicated requests to a subset of servers. Hence, we consider the (n,m,k) replication schemes, under which the user sends a replicated request to each of the m servers that are chosen from the n servers uniformly at random and waits for the first k responses, where $m \in \mathcal{N}$ and $k \in \{1, 2, \ldots, m\}$. Making the same assumptions as in Section III, we can derive the expected AoI at the user's side in a similar manner. Specifically, reusing the proof of Theorem 1 and replacing n with m in the proof, we can show the following:

$$\mathbb{E}[\Delta(k)] = \frac{1}{\nu}(\mathbf{H}(m) - \mathbf{H}(m - k)) + \frac{1}{k\lambda}. \quad (14)$$

2) Uniformly distributed response time: Note that our current analysis requires the memoryless property of the Poisson updating process. However, the analysis can be extended to the uniformly distributed response time. We make the same assumptions as in Section III, except that the response time is now uniformly distributed on interval $[b, b+h]$ with $b \geq 0$ and $h \geq 0$. In this case, it is easy to derive $\mathbb{E}[R_{(k)}] = \frac{kh}{n+1} + b$ (see, e.g., [46]). Since Eq. (9) still holds, from Eq. (6) we have

$$\mathbb{E}[\Delta(k)] = \frac{kh}{n + 1} + b + \frac{1}{k\lambda}. \quad (15)$$

Following a similar line of analysis to that in the proof of Theorem 2, we can show that an optimal solution k^* can be computed as

$$k^* = \min\left\{ \left\lceil \frac{2(n+1)}{\sqrt{h^2\lambda^2 + 4h\lambda(n+1)} + h\lambda} \right\rceil, n \right\}. \quad (16)$$

3) Heterogeneous servers: In order to obtain the theoretical results and corresponding insights, we have assumed homogeneous servers in our model. However, it is important to consider realistic settings with heterogeneous servers. That is, the servers have different mean inter-update times and different mean response times. In the following, we share our thoughts about the extension of our analysis to the settings with heterogeneous servers and discuss the challenges. Recall from Eq. (6) that the expected AoI at the user's side consists of two terms: (i) $\mathbb{E}[R_{(k)}]$, the expected total waiting time for receiving

the first k responses, and (ii) $\mathbb{E}[\min_{i \in \mathcal{K}} \Delta_i(s)]$, the expected AoI of the freshest information among these k responses at request time s. Following a similar line of analysis to that in the proof of Theorem 1 and applying the order statistics results for independent and non-identically distributed exponential random variables, it is not difficult to derive the expression for $\mathbb{E}[R_{(k)}]$, which is more involved though. On the other hand, it becomes much harder to derive the closed-form expression for $\mathbb{E}[\min_{i \in \mathcal{K}} \Delta_i(s)]$ as the analysis involves $\binom{n}{k}$ possible combinations for the realization of the first k responses and the probability of each realization depends on the mean response times. Therefore, it becomes more challenging to derive the closed-form expression for the expected AoI and thus the optimal solution k^* in such heterogeneous settings.

V. AoI-based Utility Maximization

In Section IV, our study has been focused on minimizing the expected AoI at the user's side. For certain practical applications, however, the user might be more interested in maximizing the utility that is dependent on the AoI than minimizing the AoI itself. Such an AoI-based utility function can serve as a *Quality of Experience (QoE)* metric, which measures the user's satisfaction level with respect to freshness of the received information. To that end, in this section we will investigate the problem of AoI-based utility maximization. Specifically, we will consider both cases of known and unknown system parameters (i.e., the updating rate and the mean response time) in Sections V-B and V-C, respectively.

A. AoI-based Utility Function

Consider a function $U : [0, \infty) \to [0, \infty)$, which maps the AoI at the user's side under the (n, k) replication scheme (i.e., $\Delta(k)$) to a utility obtained by the user. Such a function $U(\cdot)$ is called a utility function. Similar to [48], we assume that the utility function $U(\cdot)$ is measurable, non-negative, and non-increasing. The specific choice of the utility function depends on applications under consideration in practice.

We consider the same model as that in Section III. From the analysis in the proof of Theorem 1, it is easy to see that the AoI, $\Delta(k)$, is the sum of $k+1$ independent exponential random variables, which are $R_{(j)} - R_{(j-1)}$ for $j \in \{1, 2, \ldots, k\}$, and $\min_{i \in \mathcal{K}} \Delta_i(s)$. Therefore, the AoI, $\Delta(k)$, is a hyperexponential random variable (or a generalized Erlang random variable). The probability density function of a hyperexponential random variable with rate parameters $\alpha_1, \alpha_2, \ldots, \alpha_r$ can be expressed as $f(x) = \sum_{i=1}^{r} w_i \alpha_i e^{-\alpha_i x}$, where α_i is the rate of the i-th exponential distribution and $w_i = \prod_{j=1, j \neq i}^{r} \frac{\alpha_j}{\alpha_j - \alpha_i}$. For the AoI, $\Delta(k)$, we have $r = k + 1$, and the rate parameters α_i's are: $\alpha_i = (n + 1 - i)\nu$ for $i = 1, 2, \ldots, k$ and $\alpha_{k+1} = k\lambda$. Then, the expected utility can be calculated as

$$\mathbb{E}[U(\Delta(k))] = \int_0^\infty U(x) \sum_{i=1}^{k+1} w_i \alpha_i e^{-\alpha_i x} dx. \quad (17)$$

Now, the problem is to find the optimal value k^* that achieves the maximum expected utility:

$$k^* \in \underset{k \in \mathcal{N}}{\operatorname{argmax}} \, \mathbb{E}[U(\Delta(k))]. \quad (18)$$

In the following subsections, we will consider a specific AoI-based utility function, which is an exponential function of the negative AoI, and aim to maximize the expected utility. We will consider both cases of known and unknown system parameters (i.e., the updating rate and the mean response time).

B. Case with Known System Parameters

In this subsection, we consider a specific AoI-based utility function in the following exponential form:

$$U(\Delta(k)) = e^{-a\Delta(k)}, \tag{19}$$

where a is a positive constant. The above exponential utility function implies that the user receives the full utility when the AoI is zero (which is an ideal case) and the utility decreases exponentially as the AoI increases. Such a utility function decreases very quickly with respect to the AoI and is desirable for real-time applications that require extremely fresh information to provide satisfactory service to the users (e.g., stock quote service).

Assuming that the updating rate and the mean response time are known, we first derive a closed-form formula for computing the expected utility $\mathbb{E}[U(\Delta(k))]$. Then, we find an optimal k^* that yields the maximum expected utility. The main results of this subsection are stated in Theorems 3 and 4. The proofs of Theorems 3 and 4 follow a similar line of analysis to that for Theorems 1 and 2, respectively. The detailed proofs are provided in our online technical report [49].

Theorem 3. *Under the (n, k) replication scheme, the expected utility can be expressed as*

$$\mathbb{E}[U(\Delta(k))] = \frac{k\lambda}{k\lambda + a} \prod_{j=1}^{k} \frac{(n+1-j)\nu}{(n+1-j)\nu + a}. \tag{20}$$

Theorem 4. *An optimal solution k^* to Problem* (18) *(i.e., achieving the maximum expected utility) can be computed as*

$$k^* = \min\left\{ \left\lceil \frac{2\nu n}{\sqrt{(\lambda + \nu + a)^2 + 4\lambda\nu n} + (\lambda + \nu + a)} \right\rceil, n \right\}. \tag{21}$$

Remark. Similar to the AoI minimization problem studied in Section IV-B, there are also two interesting special cases: (i) waiting for the first response only (i.e., $k^* = 1$) and (ii) waiting for all the responses (i.e., $k^* = n$). In Corollary 2, we provide a sufficient and necessary condition for each case. The proof of is provided in our online technical report [49].

Corollary 2. *(i) $k^* = 1$ is an optimal solution to Problem* (18) *if and only if $\lambda \geq \frac{\nu(n-1)}{2} - \frac{a}{2}$; (ii) $k^* = n$ is an optimal solution to Problem* (18) *if and only if $\lambda \leq \frac{\nu}{n(n-1)} - \frac{a}{n}$.*

C. Case with Unknown System Parameters

In Section V-B, we have addressed the utility maximization problem in Eq. (18), assuming the knowledge of the updating rate (i.e., λ) and the mean response time (i.e., $1/\nu$). Similar assumptions are also made for obtaining a good understanding of the studied theoretical problems (see, e.g., [9], [48] and references therein). However, such information is typically unavailable to the user in practice. For example, the user generally has no prior knowledge of the updating processes between the information source and the servers. Moreover, it is difficult, if not impossible, for the user to estimate the updating rate as the user has no direct observation about the updating processes. Therefore, an interesting and important question naturally arises: *How to maximize the expected utility in the presence of unknown system parameters?*

To that end, in this subsection we aim to address the above question through the design of learning-based algorithms. Specifically, in the presence of unknown system parameters we will reformulate the utility maximization problem as a stochastic *Multi-Armed Bandit (MAB)* problem. *To the best of our knowledge, this is one of the first studies that leverage an MAB formulation to study the AoI problem.*

In the following, we will first briefly introduce the basic setup of the stochastic MAB model (Section V-C1). Then, we formulate the utility maximization problem with unknown system parameters as an MAB problem and explain the special linear feedback graph of our problem, which can be exploited to achieve improved performance guarantees (Section V-C2). Finally, we introduce several MAB algorithms that can be applied to address our problem (Section V-C3).

1) The MAB Model: The MAB model has been widely employed for studying many sequential decision-making problems of practical importance (clinical trials, network resource allocation, online ad placement, crowdsourcing, etc.) with unknown parameters (see, e.g., [50]–[53]).

In the classic MAB model, a player (i.e., a decision maker) is faced with n options, which are often called arms in the MAB literature. In each round, the player can choose to play one arm and receives the reward generated by the played arm. The reward of playing arm k in round t, denoted by $X_{k,t}$, is a random variable distributed on interval $[0, 1]$, i.e., $X_{k,t} \in [0, 1]$. The reward $X_{k,t}$ of each arm k is assumed to be *i.i.d.* over time. Let μ_k be the mean reward of arm k; let μ^* be the highest mean reward among all the arms, i.e., $\mu^* \triangleq \max_k \mu_k$. The specific distributions of $X_{k,t}$'s and the values of μ_k's are unknown to the player.

An algorithm π chooses an arm I_t to play in each round $t \in \{1, 2, \ldots, T\}$, where T is the length of the time horizon. The objective here is to design an algorithm that maximizes the expected cumulative reward during this time horizon, i.e., $\sum_{t=1}^{T} \mu_{I_t}$. This is equivalent to minimizing the *regret*, which is the difference between the expected cumulative reward obtained by an optimal algorithm that always plays the best arm and that of the considered algorithm. We use $R(T)$ to denote the regret, which is formally defined as follows:

$$R(T) \triangleq \mu^* T - \sum_{t=1}^{T} \mu_{I_t}. \tag{22}$$

In order to maximize the reward or minimize the regret, the player is faced with a key tradeoff: how to balance *exploitation* (i.e., playing the arm with the highest empirical mean reward) and *exploration* (i.e., trying other arms, which could potentially be better)? There exist several well-known algorithms that can address this challenge. We will discuss them in Section V-C3.

2) The MAB Formulation of the Utility Maximization Problem: We now want to formulate the utility maximization problem with unknown system parameters as an MAB problem. Note that when the updating rate and the mean response time are unknown, one cannot easily derive a closed-form formula for the expected utility and find the optimal solution as in Section V-B. Therefore, for each sent request the user needs to decide how many responses to wait for in a dynamic manner. In this case, one can naturally reformulate the utility maximization problem using the MAB model: making a decision for each sent request corresponds to a round; waiting for k responses corresponds to playing arm k. Let $\Delta(k,t)$ be the AoI at the user's side when the user sends the t-th request and waits for the first k responses. Then, the utility $U(\Delta(k,t))$, normalized to interval $[0,1]$, corresponds to the obtained reward $X_{k,t}$ of playing arm k in round t. The mean reward of arm k is $\mu_k = \mathbb{E}[X_{k,t}] = \mathbb{E}[U(\Delta(k,t)]$. In this MAB formulation, the utility function $U(\cdot)$ is not limited to the exponential function in the form of Eq. (19); instead, $U(\cdot)$ can be very general, as long as it is a measurable, non-negative, and non-increasing of the AoI. Note that for each arm k, the reward $X_{k,t} = U(\Delta(k,t))$ is *i.i.d.* over rounds since the AoI $\Delta(k,t)$ is *i.i.d.* over rounds due to the memoryless property of the exponential distribution. We provide a detailed explanation for this in our online technical report [49].

Recently, MAB models with side observations have been studied (see, e.g., [54]–[56]). In these models, playing an arm not only reveals the reward of the played arm but also that of some other arm(s). Such side observations are typically encoded in a feedback graph, where each node corresponds to an arm and each directed edge (k, k') means that playing arm k also reveals the reward of arm k'.

We would like to point out that the utility maximization problem with unknown system parameters can be formulated as an MAB problem with side observations. Note that although the rewards of different arms are dependent in our problem, as pointed out in [57], the MAB formulation and various learning algorithms are still applicable in such settings. Moreover, as illustrated in Fig. 4, such dependence leads to a special linear structure in the associated feedback graph of our problem. Specifically, note that upon receiving the k-th response, the user has the information about the first $k-1$ responses. Thus, the user can know the utility she would have obtained if she had waited for only k' responses for all $k' < k$. Mapping this property to the MAB model, it means that playing arm k reveals not only the reward of arm k but also that of arm k' for all $k' < k$. Such special properties can be leveraged to design learning algorithms that perform exploration more efficiently and thus lead to an improved regret performance.

3) Learning Algorithms: There exist several well-known learning algorithms that can address the classic MAB problem, including ϵ_t-*Greedy* [57] and *Upper Confidence Bound (UCB)* [50], [57], [58]. In the sequel, we will introduce these algorithms and explain how to leverage the side observations and the special linear structure of the graphical feedback to design algorithms with improved regret upper bounds.

We begin with ϵ_t-Greedy, a very simple algorithm that performs exploration explicitly. Specifically, it plays the arm
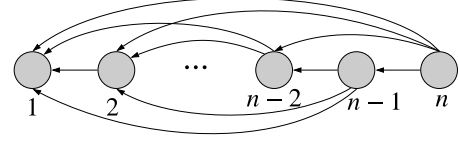

Fig. 4: Linear feedback graph where each node $k \in \{2, 3, \ldots, n\}$ has a directed edge to every node in $\{1, 2, \ldots, k-1\}$ and node 1 does not have any outgoing edge

---

Algorithm 1 ϵ_t-Greedy [57]

1: Input: $c > 0$ and $0 < d < 1$.
2: Let $\bar{x}_k(t)$ be the empirical mean reward of arm k at the beginning of round t.
3: **for** $t = 1, 2, \ldots, T$ **do**
4:     Let $j^* \in \arg\max_k \bar{x}_k(t)$ and $\epsilon_t \triangleq \min\{1, \frac{cn}{d^2 t}\}$.
5:     Play arm j^* with probability $1 - \epsilon_t$; play a random arm (uniformly) with probability ϵ_t.
6:     Update $\bar{x}_{I_t}(t+1)$ for the played arm I_t.
7: **end for**

Algorithm 2 ϵ_t-Greedy-LP [56]

1: Input: $c > 0$ and $0 < d < 1$.
2: Let $\bar{x}_k(t)$ be the empirical mean reward of arm k at the beginning of round t.
3: **for** $t = 1, 2, \ldots, T$ **do**
4:     Let $j^* \in \arg\max_k \bar{x}_k(t)$ and $\epsilon_t \triangleq \min\{1, \frac{cn}{d^2 t}\}$.
5:     Play arm j^* with probability $1 - \epsilon_t$; play arm n with probability ϵ_t.
6:     Update $\bar{x}_k(t+1)$ for all $k \in \{1, \cdots, n\}$, accounting for all the observations, including side observations.
7: **end for**

---

with the highest empirical mean reward with probability $1 - \epsilon_t$ (i.e., exploitation) and plays a random arm (uniformly) with probability ϵ_t (i.e., exploration), where ϵ_t decreases as $O(1/t)$. We summarize the operations of ϵ_t-Greedy in Algorithm 1. When side observations are available, one can incorporate additional samples from side observations into the update of the empirical mean reward of non-played arms. We call ϵ_t-Greedy that exploits the side observations as ϵ_t-*Greedy-N*. Note that ϵ_t-Greedy-N is almost the same as ϵ_t-Greedy except that in Line 6 of Algorithm 1, one needs to update the empirical mean reward $\bar{x}_k(t+1)$ for all arms $k \in \{1, \cdots, n\}$, including the non-played arms, accounting for side observations. Apparently, ϵ_t-Greedy-N accelerates the exploration process by taking advantage of additional samples from side observations and is expected to outperform ϵ_t-Greedy.

Although ϵ_t-Greedy-N leverages side observations and can speed up the exploration process compared to ϵ_t-Greedy, it still randomly chooses an arm during the exploration process, being agnostic about the structure of the feedback graph. Therefore, all the arms have to be played in the exploration phase (i.e., with probability ϵ_t). The analysis in [57] suggests that $O(\log T)$ samples are sufficient for accurately estimating the mean reward of an arm. This implies that both ϵ_t-Greedy

Algorithm 3 UCB1 [57]

1: Let $\bar{x}_k(t)$ and $T_k(t)$ be the empirical mean reward and the total number of samples of arm k at the beginning of round t, respectively.
2: **for** $t = 1, 2, \ldots, T$ **do**
3:    Play arm I_t such that

$$I_t \in \underset{k}{\operatorname{argmax}} \left\{ \bar{x}_k(t) + \sqrt{\frac{2 \log t}{T_k(t)}} \right\}. \quad (23)$$

4:    Update $\bar{x}_{I_t}(t+1)$ and $T_{I_t}(t+1)$.
5: **end for**

---

and ϵ_t-Greedy-N have a regret upper bounded by $O(n \log T)$. However, many of such explorations appear unnecessary in our studied problem. This is because in our utility maximization problem, playing arm n reveals a sample for every arm, due to the special linear structure of the feedback graph in Fig. 4. This suggests that one should always choose arm n for exploration, which leads to a graph-aware algorithm summarized in Algorithm 2. We call this algorithm ϵ_t-*Greedy-LP* as it turns out to be a special case of the ϵ_t-Greedy-LP algorithm proposed in [56].

One can show that the regret of ϵ_t-Greedy-LP is upper bounded by $O(\log T)$, which improves upon $O(n \log T)$ of ϵ_t-Greedy and ϵ_t-Greedy-N. This result follows immediately from Corollary 8 of [56] as the linear feedback graph in Fig. 4 is a special case of the graphs considered in [56]. Note that the improved regret upper bound relies on the assumption that ϵ_t-Greedy-LP has the knowledge of the difference between the reward of the optimal arm and that of the best suboptimal arm for choosing parameters d and c (see Corollary 8 of [56] for the specific form).

Next, we consider another simple algorithm called Upper Confidence Bound (UCB). As the name suggests, UCB considers the upper bound of a suitable confidence interval for the mean reward of each arm and chooses the arm with the highest such upper confidence bound (see, e.g., Eq. (23)). There are several variants of the UCB algorithm [50], [57], [58]. We present a popular variant, called *UCB1*, in Algorithm 3. When side observations are available, similar to ϵ_t-Greedy-N, there is a slightly modified UCB algorithm, called *UCB-N* [55], which incorporates additional samples from side observations into the update of the empirical mean reward and the total number of samples of non-played arms (i.e., Line 4 in Algorithm 3). Like ϵ_t-Greedy-N, UCB-N is also agnostic about the structure of the feedback graph. In order to take the graph structure into consideration, we introduce *UCB-LP*, which is based on another UCB variant, called UCB-Improved [58]. UCB-LP is a special case of the one proposed in [56]. We summarize the operations of UCB-LP in Algorithm 4.

The key idea of UCB-LP is the following: we divide T into multiple stages. For each stage m, we use B_m to denote the set of arms not eliminated yet and use $\tilde{\delta}_m$ to estimate δ_k. At the beginning, set B_0 is initialized to the set of all arms; the value of $\tilde{\delta}_0$ is initialized to 1. We ensure that by the end of stage m, there are at least t_m samples available for each arm

Algorithm 4 UCB-LP [56]

1: **Initialization:** Set $B_0 = \mathcal{N}$ and $\tilde{\delta}_0 = 1$.
2: Let $\bar{x}_k(m)$ and $T_k(m)$ be the empirical mean reward and the total number of samples of arm k up to and including stage m.
3: **for** $m = 0, 1, 2, \ldots, \left\lfloor \frac{1}{2} \log_2 \frac{T}{e} \right\rfloor$ **do**
4:    **Arm selection:**
5:    Let $t_m \triangleq \left\lceil \frac{2 \log(T \tilde{\delta}_m^2)}{\tilde{\delta}_m^2} \right\rceil$.
6:    **if** $|B_m| = 1$ **then**
7:      Play the single arm in B_m until time T.
8:    **else if** $2|B_m|\tilde{\delta}_m \geq 1$ **then**
9:      Play arm n for $(t_m - t_{m-1})$ times.
10:    **else**
11:      Play each arm $k \in B_m$ for $(t_m - t_{m-1})$ times.
12:    **end if**
13:    Update $\bar{x}_k(m)$ and $T_k(m)$ for all $k \in \{1, \cdots, n\}$.
14:    **Arm elimination:**
15:    Let D_m be the set of all arms j in B_m for which

$$\bar{x}_j(m) + \sqrt{\frac{\log(T \tilde{\delta}_m^2)}{2 T_j(m)}} < \max_{k \in B_m} \left\{ \bar{x}_k(m) - \sqrt{\frac{\log(T \tilde{\delta}_m^2)}{2 T_k(m)}} \right\}. \quad (24)$$

16:    Set $B_{m+1} = B_m \setminus D_m$ and $\tilde{\delta}_{m+1} = \tilde{\delta}_m / 2$.
17: **end for**

Algorithm 5 UCB-LFG

1: **Initialization:** Set $B_0 = \mathcal{N}$ and $\tilde{\delta}_0 = 1$.
2: Let $\bar{x}_k(m)$ and $T_k(m)$ be the empirical mean reward and the total number of samples of arm k up to and including stage m.
3: **for** $m = 0, 1, 2, \ldots, \left\lfloor \frac{1}{2} \log_2 \frac{T}{e} \right\rfloor$ **do**
4:    **Arm selection:**
5:    Let $t_m \triangleq \left\lceil \frac{2 \log(T \tilde{\delta}_m^2)}{\tilde{\delta}_m^2} \right\rceil$.
6:    **if** $|B_m| = 1$ **then**
7:      Play the single arm in B_m until time T.
8:    **else**
9:      Play arm $j^*(B_m)$ for $(t_m - t_{m-1})$ times, where $j^*(B_m)$ is the largest index of arms in set B_m.
10:    **end if**
11:    Update $\bar{x}_k(m)$ and $T_k(m)$ for all $k \in \{1, \cdots, n\}$.
12:    **Arm elimination:**
13:    Let D_m be the set of all arms j in B_m for which

$$\bar{x}_j(m) + \sqrt{\frac{\log(T \tilde{\delta}_m^2)}{2 T_j(m)}} < \max_{k \in B_m} \left\{ \bar{x}_k(m) - \sqrt{\frac{\log(T \tilde{\delta}_m^2)}{2 T_k(m)}} \right\}. \quad (25)$$

14:    Set $B_{m+1} = B_m \setminus D_m$ and $\tilde{\delta}_{m+1} = \tilde{\delta}_m / 2$.
15: **end for**

---

in set B_m, from playing either arm n or the arm itself, where t_m is determined by $\tilde{\delta}_m$ (Lines 8-12). Then, at the end of stage m we obtain set B_{m+1} by eliminating those arms estimated to be suboptimal according to Eq. (24) and obtain $\tilde{\delta}_{m+1}$ by halving the value of $\tilde{\delta}_m$.

Under some mild assumptions, one can show that for

our studied problem with a linear feedback graph, UCB-LP achieves an improved regret upper bounded of $O(\log T)$ compared to $O(n \log T)$ of UCB1 and UCB-N. This result follows immediately from Proposition 10 of [56]. Note that without the knowledge of δ_k, UCB-LP achieves an improved regret upper bound that is similar to that of ϵ_t-Greedy-LP. Although UCB-LP presented in Algorithm 4 requires the information about the time horizon T, this requirement can be relaxed using the techniques suggested in [56], [58].

Furthermore, leveraging the linear feedback graph in Fig. 4, we propose a further enhanced UCB algorithm by slightly modifying UCB-LP. We call this new algorithm UCB-LFG (UCB-Linear Feedback Graph) and present it in Algorithm 5. The key difference is in Line 9 (vs. Lines 8-12 in Algorithm 4). Recall that in Algorithm 4, the purpose of Lines 8-12 is to explore arms in set B_m. Specifically, this is to ensure that by the end of stage m, each arm in set B_m has at least t_m samples. Because of the linear feedback graph, this exploration step can be achieved in a smaller number of rounds. Specifically, during stage m we simply play arm $j^*(B_m)$ for $(t_m - t_{m-1})$ times, where $j^*(B_m)$ is the largest index of arms in set B_m. This is because each time when arm $j^*(B_m)$ is played, there will be a sample generated for every arm in set B_m, thanks to the special structure of the linear feedback graph. Following the regret analysis for UCB-LP in [56], we can show that UCB-LFG achieves an improved regret upper bounded of $O(\log T)$. Although UCB-LFG and UCB-LP have the same regret upper bound, UCB-LFG typically achieves a better empirical performance than UCB-LP. This can be observed from the simulation results in Section VI-B.

VI. NUMERICAL RESULTS

In this section, we perform extensive simulations to elucidate our theoretical results. We present the simulation results for AoI minimization and AoI-based utility maximization in Sections VI-A and VI-B, respectively.

A. Simulation Results for AoI Minimization

We first describe our simulation settings. We consider an information-update system with $n = 20$ servers. Throughout the simulations, the updating process at each server is assumed to be Poisson with rate λ and is *i.i.d.* across the servers. The user's request for the information is generated at time s, which is selected uniformly at random on interval $[0, 10^6/\lambda]$. This means that each server has a total of 10^6 updates on average.

Next, we evaluate the AoI performance through simulations for three types of response time distribution: *exponential*, *uniform*, and *Gamma*. First, we assume that the response time is exponentially distributed with mean $1/\nu$. We consider three representative setups: (i) $\lambda = 1, \nu = 200$; (ii) $\lambda = 1, \nu = 5$; (iii) $\lambda = 100, \nu = 2$. Fig. 5a shows how the average AoI changes as the number of responses k varies, where each point represents an average of 10^3 simulation runs. We also include plots of our theoretical results (i.e., Eq. (7)) for comparison. A crucial observation from Fig. 5a is that the simulation results match our theoretical results perfectly. In addition, we observe three different behaviors of the average AoI performance: a)

If the inter-update time is much larger than the response time (e.g., $\lambda = 1$, $\nu = 200$), then the average AoI decreases as k increases, and thus, it is worth waiting for all the responses so as to achieve a smaller average AoI. b) In contrast, if the inter-update time is much smaller than the response time (e.g., $\lambda = 100$, $\nu = 2$), then the average AoI increases as k increases, and thus, it is not beneficial to wait for more than one response. c) When the inter-update time is comparable to the response time (e.g., $\lambda = 1$, $\nu = 5$), then as k increases, the AoI would first decrease and then increase. In this setup, when k is small, the freshness of the data at the servers dominates, and thus, waiting for more responses helps reduce the average AoI. On the other hand, when k becomes large, the total waiting time becomes dominant, and thus, the average AoI increases as k further increases.

In Section IV-C, we discussed the extension of our theoretical results to the case of uniformly distributed response time. Hence, we also perform simulations for the response time uniformly distributed on $[\frac{1}{2\nu}, \frac{3}{2\nu}]$ with mean $1/\nu$. Fig. 5b presents the average AoI as the number of responses k changes. In this scenario, the simulation results also match our theoretical results (i.e., Eq. (15)). Also, we observe a very similar phenomenon to that in Fig. 5a on how the average AoI varies as k increases in three different simulation setups.

In addition, Fig. 5c presents the simulation results for the response time with Gamma distribution, which can be used to model the response time in relay networks [59]. Specifically, we consider a special class of the Gamma(r, θ) distribution that is the sum of r *i.i.d.* exponential random variables with mean θ (which is also called the Erlang distribution). Then, the mean response time $1/\nu$ is equal to $r\theta$. We fix $r = 5$ in the simulations. In this case, although we do not have any analytical results, the observations are similar to that under the exponential and uniform distributions.

Finally, we investigate the impact of the system parameters (the updating rate, the mean response time, and the total number of servers) on the optimal number of responses k^* and the *AoI improvement ratio*, defined as $\rho_\Delta \triangleq \mathbb{E}[\Delta(1)]/\mathbb{E}[\Delta(k^*)]$. The AoI improvement ratio captures the gain in the AoI reduction under the optimal scheme compared to a naive scheme of waiting for the first response only.

Fig. 6a shows the impact of the updating rate λ. We observe that the optimal number of responses k^* decreases as λ increases. This is because when the updating rate is large, the AoI diversity at the servers is small. In this case, waiting for more responses is unlikely to receive a response with much fresher information. Therefore, the optimal scheme will simply be a naive scheme that waits for the first response only, when the updating rate is relatively large (e.g., $\lambda = 2$). Fig. 6b shows the impact of the mean response time $1/\nu$. We observe that the optimal number of responses k^* increases as ν increases. This is because when ν is large (i.e., when the mean response time is small), the cost of waiting for additional responses becomes marginal, and thus, waiting for more responses is likely to lead to the reception of a response with fresher information. Fig. 6c shows the impact of the total number of servers n. We observe that both the optimal number of responses k^* and the improvement ratio increase with n. This is because
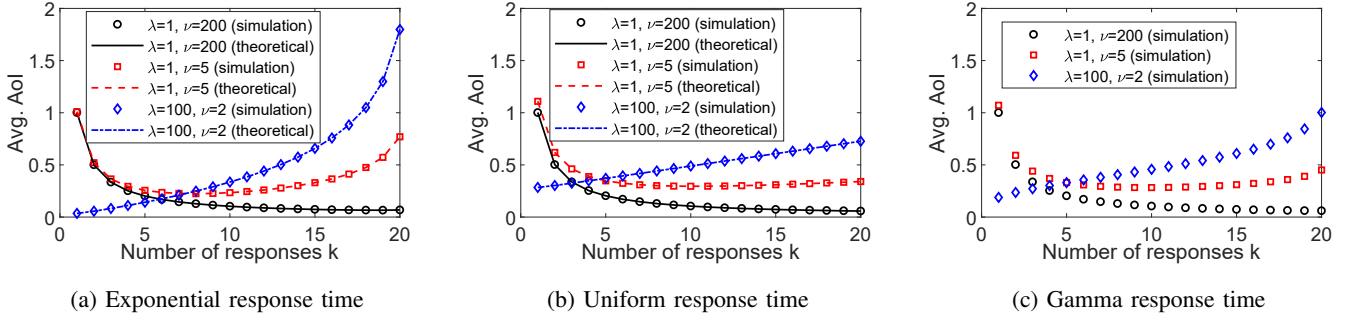
(a) Exponential response time   (b) Uniform response time   (c) Gamma response time

Fig. 5: Simulation results of average AoI vs. the number of responses k for three different types of response time distributions

(a) Impact of updating rate λ (b) Impact of mean response time $1/\nu$ (c) Impact of total number of servers n
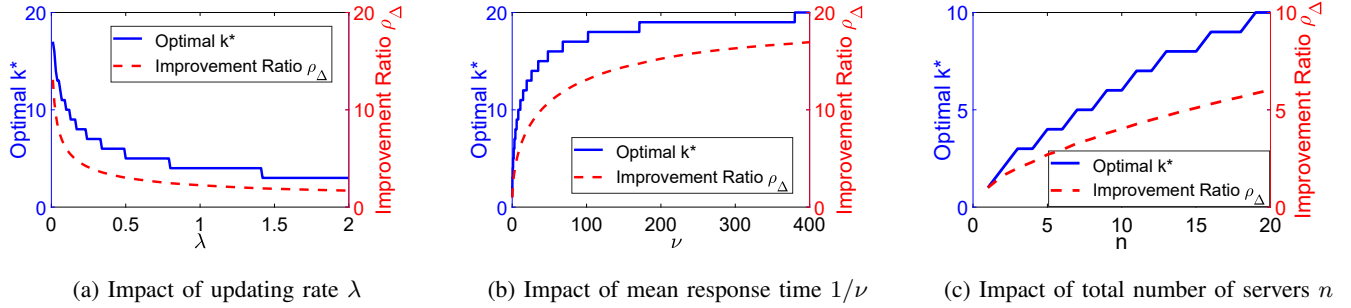
Fig. 6: Impact of the system parameters on the optimal k^* and the corresponding improvement ratio. We consider the exponential distribution for the response time. In (a), we fix $\nu = 1, n = 20$; in (b), we fix $\lambda = 1, n = 20$; in (c), we fix $\lambda = 1, \nu = 10$.

an increased number of servers leads to more diversity gains both in the AoI at the servers and in the response time. As we discussed at the end of Section IV-B, the optimal solution k^* scales with $O(\sqrt{n})$ as the number of servers n becomes large.

B. Simulation Results for AoI-based Utility Maximization

In this subsection, we consider the maximization of an AoI-based exponential utility function (Eq. (19) with $a = 1$) and present simulation results for the utility performance under the same settings as in Section VI-A. We first evaluate the utility performance in the setting with known system parameters. Then, we evaluate various learning algorithms described in Section V-C3, when the system parameters are unknown.

1) Utility Maximization with Known Parameters: The setups we consider are exactly the same as those in Section VI-A, except that we now focus on the utility performance instead of the AoI performance. In Fig. 7, we present the simulation results for the average utility performance with a varying number of responses k in three representative setups. The observations are also similar, except that the utility has an opposite trend compared to the AoI. This is because the utility is a non-increasing function of the AoI.

Similarly, we also investigate the impact of the system parameters on the optimal number of responses k^* (with respect to utility maximization) and the *utility improvement ratio*, defined as $\rho_U \triangleq \mathbb{E}[U(\Delta(k^*))]/\mathbb{E}[U(\Delta(1))]$. The utility improvement ratio captures the gain in the utility improvement under the optimal scheme compared to a naive scheme of waiting for the first response only. We present the results in Fig. 8, from which we can make similar observations to those from Fig. 6 for AoI minimization.

2) Utility Maximization with Unknown Parameters: Next, we consider a more realistic scenario where the system parameters (i.e., the updating rate and the mean response time) are unknown to the user. Given that the overall behaviors are similar for different types of response time distributions (see Fig. 7), in the following evaluations we will focus on the case where the update process is Poisson with rate λ and the response time is exponentially distributed with mean $1/\nu$.

As in Section VI-A, we assume $n = 20$ and consider three representative setups: (i) $\lambda = 1, \nu = 200$; (ii) $\lambda = 1, \nu = 5$; (iii) $\lambda = 100, \nu = 2$. We evaluate the regret performance of two classes of learning algorithms we introduced in Section V-C3: the Greedy algorithms (i.e., ϵ_t-Greedy, ϵ_t-Greedy-N, and ϵ_t-Greedy-LP) and the UCB algorithms[1] (i.e., UCB1, UCB-N, UCB-Improved, UCB-LP, and UCB-LFG). For Greedy algorithms, we use $d = 0.05$ and $c = 1$ in all the three setups. In Fig. 9, we plot the evolution of cumulative regret of the considered algorithms over 10^6 rounds for each of the above three setups. The results represent an average of 10 simulation runs. From the simulation results in Fig. 9, we can observe the following.

First, algorithms that take advantage of side observations generally outperform their counterparts that do not use side observations. That is, ϵ_t-Greedy-N and UCB-N outperform ϵ_t-Greedy and UCB1, respectively. This is because additional samples from side observations can help accelerate the learning process.

Second, although graph-aware algorithms can achieve im-

---

[1]We do not include the results for UCB-Improved as it performs much worse than the other algorithms in the setups we consider.

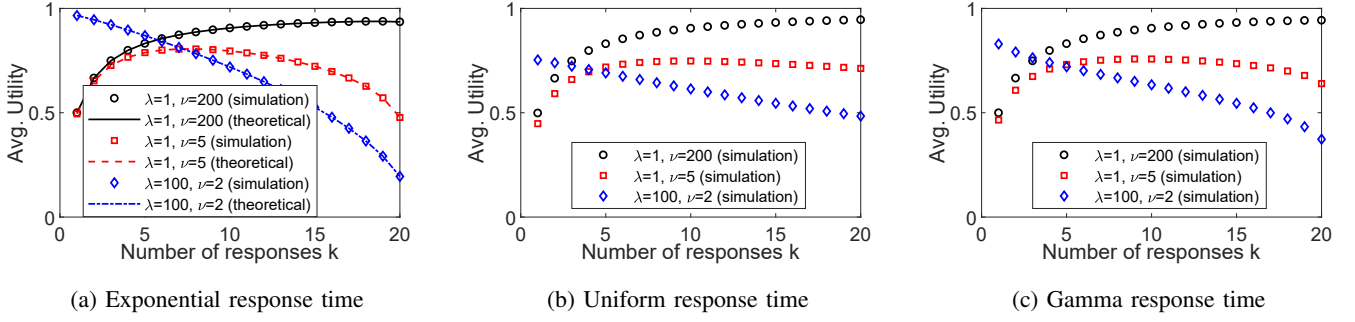(a) Exponential response time   (b) Uniform response time   (c) Gamma response time

Fig. 7: Simulation results of average utility vs. the number of responses k for three different types of response time distributions

(a) Impact of updating rate λ (b) Impact of mean response time $1/\nu$ (c) Impact of total number of servers n
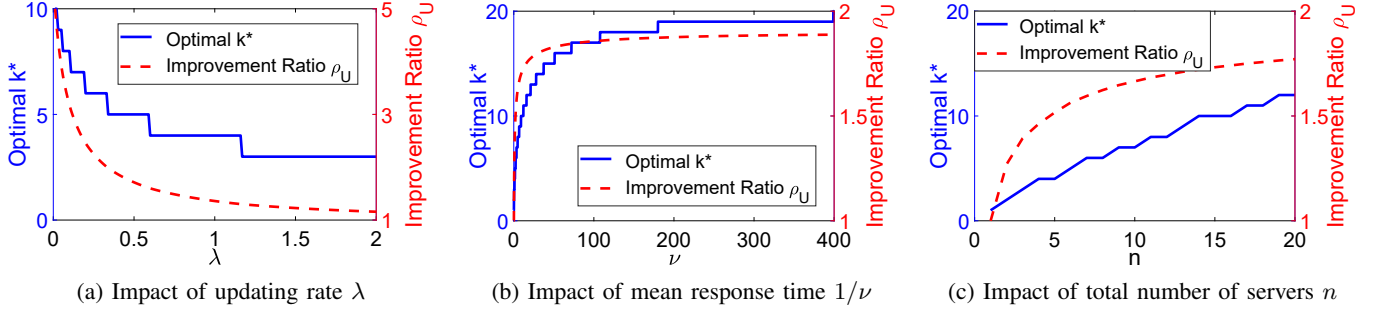
Fig. 8: Impact of the system parameters on the optimal k^* for utility maximization and the corresponding improvement ratio. We consider the exponential distribution for the response time. In (a), we fix $\nu = 1, n = 20$; in (b), we fix $\lambda = 1, n = 20$; in (c), we fix $\lambda = 1, \nu = 10$.

(a) Setup (i): $\lambda = 1, \nu = 200$ (b) Setup (ii): $\lambda = 1, \nu = 5$ (c) Setup (iii): $\lambda = 100, \nu = 2$
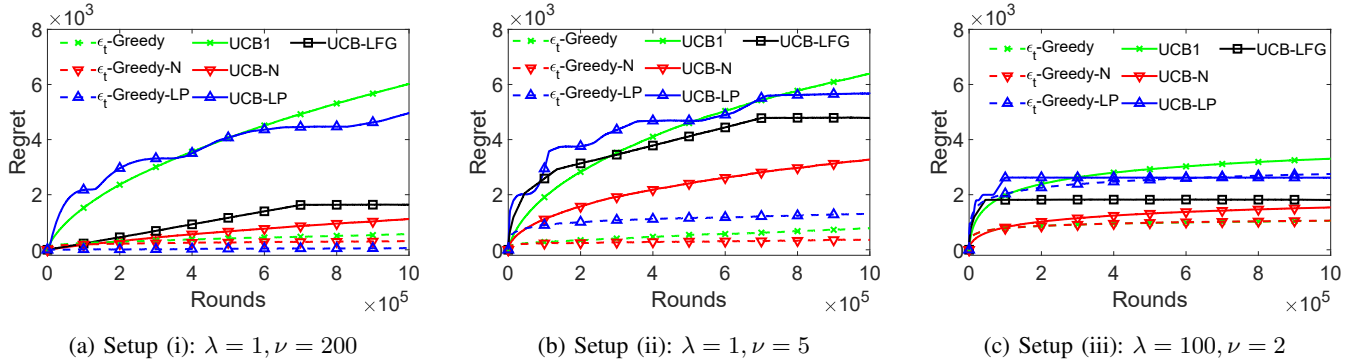
Fig. 9: Comparison of regret performance of various learning algorithms in three representative setups

proved regret upper bounds, their empirical performances may or may not be better than that of their graph-agnostic counterparts. That is, ϵ_t-Greedy-LP and UCB-LP may or may not be better than ϵ_t-Greedy-N and UCB-N, respectively. Consider the Greedy algorithms for example. In Setup (i), ϵ_t-Greedy-LP slightly outperforms ϵ_t-Greedy-N. This is because in the phase of exploration, ϵ_t-Greedy-LP always chooses arm n, which happens to be the best arm. However, in Setup (ii), ϵ_t-Greedy-LP performs worse than ϵ_t-Greedy-N. This is because arm n is no longer the best arm, and in fact, it can be much worse than the optimal arm. This phenomenon is exacerbated in Setup (iii), where arm n is the worst arm. Among all the considered UCB algorithms, UCB-N has the best empirical performance. This is because UCB-LP and UCB-LFG are modified from UCB-Improved, which is an "arm-elimination" algorithm and is very different from UCB1, from which UCB-

N is modified. Although UCB-Improved has a better regret upper bound with a smaller constant factor, it has a much worse empirical performance than UCB1 in the setups we consider. Therefore, it is not surprising that UCB-N has a better empirical performance than UCB-LP and UCB-LFG.

Third, UCB-LFG typically outperforms UCB-LP. This is expected because UCB-LFG is a further enhanced version of UCB-LP. Specifically, UCB-LFG explicitly exploits the linear structure of the feedback graph and can accelerate the learning process by reducing the number of rounds for exploration.

Finally, ϵ_t-Greedy-N seems to be quite robust and has a very good empirical performance in all the setups we consider.

VII. CONCLUSION

In this paper, we introduced a new Pull model for studying the problems of AoI minimization and AoI-based utility max-

imization under the replication schemes. Assuming Poisson updating process and exponentially distributed response time, we derived the closed-form expression of the expected AoI at the user's side and provided a formula for computing the optimal solution. We also derived a set of similar theoretical results for the utility maximization problem. Furthermore, we considered a more realistic scenario where the user has no prior knowledge of the system parameters. In this setting, we reformulated the utility maximization problem as a stochastic MAB problem with side observations. Leveraging the special linear structure of the feedback graph associated with side observations, we introduced several learning algorithms, which outperform those basic algorithms that are agnostic about such properties. Not only did our work reveal a novel tradeoff between different levels of information freshness and different response times across the servers, but we also demonstrated the power of waiting for more than one response in minimizing the AoI as well as in maximizing the utility at the user's side.
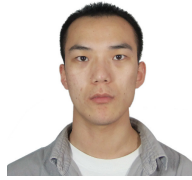
REFERENCES

[1] Y. Sang, B. Li, and B. Ji, "The power of waiting for more than one response in minimizing the age-of-information," in *Proceedings of IEEE GLOBECOM*, 2017, pp. 1–6.

[2] "Cisco visual networking index: Forecast and trends, 20172022 white paper," February 2019, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[3] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *Proceedings of IEEE SECON*, 2011, pp. 350–358.

[4] S. Kaul, R. Yates, and M. Gruteser, "On piggybacking in vehicular networks," in *Proceedings of IEEE GLOBECOM*, 2011, pp. 1–5.

[5] J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh, "Wireless sensor networks for healthcare," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1947–1960, 2010.

[6] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore, "Environmental wireless sensor networks," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1903–1917, 2010.

[7] M. Costa, S. Valentin, and A. Ephremides, "On the age of channel state information for non-reciprocal wireless links," in *Proceedings of IEEE ISIT*, 2015, pp. 2356–2360.

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

[9] A. Kosta, N. Pappas, V. Angelakis *et al.*, "Age of information: A new concept, metric, and tool," *Foundations and Trends® in Networking*, vol. 12, no. 3, pp. 162–259, 2017.

[10] Y. Sun, I. Kadota, R. Talak, and E. Modiano, "Age of information: A new metric for information freshness," *Synthesis Lectures on Communication Networks*, vol. 12, no. 2, pp. 1–224, 2019.

[11] L. Bright, A. Gal, and L. Raschid, "Adaptive pull-based data freshness policies for diverse update patterns," University of Maryland, Tech. Rep., 2004. [Online]. Available: http://drum.lib.umd.edu/handle/1903/1334

[12] ——, "Adaptive pull-based policies for wide area data delivery," *ACM Transactions on Database Systems*, vol. 31, no. 2, pp. 631–671, 2006.

[13] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, and E. Hyytia, "Reducing latency via redundant requests: Exact analysis," *ACM SIGMETRICS Performance Evaluation Review*, 2015.

[14] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Why let resources idle? Aggressive cloning of jobs with dolly," in *The 4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2012.

[15] B. Li, A. Ramamoorthy, and R. Srikant, "Mean-field-analysis of coding versus replication in cloud storage systems," in *Proceedings of IEEE INFOCOM*, 2016, pp. 1–9.

[16] D. Wang, G. Joshi, and G. Wornell, "Efficient task replication for fast response times in parallel computation," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, 2014, pp. 599–600.

[17] ——, "Using straggler replication to reduce latency in large-scale parallel computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 3, pp. 7–11, 2015.

[18] E. Pacitti, "Improving data freshness in replicated databases," Ph.D. dissertation, INRIA, 1999.

[19] J. Pereira and M. Araújo, "Evaluating data freshness in large scale replicated databases," *INForum 2010-II Simpósio de Informática*, pp. 231–242, 2010.

[20] S. K. Kaul, R. D. Yates, and M. Gruteser, "Status updates through queues," in *2012 46th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2012, pp. 1–6.

[21] R. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Transactions on Information Theory*, vol. 65, no. 3, pp. 1807–1827, 2019.

[22] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides, "Effect of message transmission path diversity on status age," *IEEE Transactions on Information Theory*, vol. 62, no. 3, pp. 1360–1374, March 2016.

[23] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Controlling the age of information: Buffer size, deadline, and packet replacement," in *MILCOM 2016-2016 IEEE Military Communications Conference*. IEEE, 2016, pp. 301–306.

[24] ——, "On the age of information with packet deadlines," *IEEE Transactions on Information Theory*, vol. 64, no. 9, pp. 6419–6428, 2018.

[25] N. Pappas, J. Gunnarsson, L. Kratz, M. Kountouris, and V. Angelakis, "Age of information of multiple sources with queue management," in *IEEE International Conference on Communications (ICC)*, June 2015, pp. 5935–5940.

[26] M. Costa, M. Codreanu, and A. Ephremides, "On the age of information in status update systems with packet management," *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1897–1910, April 2016.

[27] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Minimizing the age of information through queues," *IEEE Transactions on Information Theory*, vol. 65, no. 8, pp. 5215–5232, Aug 2019.

[28] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Minimizing the age of information in broadcast wireless networks," in *Proceedings of the Annual Conference on Communication, Control and Computing (Allerton)*, 2016.

[29] I. Kadota, A. Sinha, and E. Modiano, "Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints," *IEEE/ACM Transactions on Networking (TON)*, vol. 27, no. 4, pp. 1359–1372, 2019.

[30] R. Talak, S. Karaman, and E. Modiano, "Optimizing information freshness in wireless networks under general interference constraints," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2018, pp. 61–70.

[31] ——, "Optimizing age of information in wireless networks with perfect channel state information," in *2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. IEEE, 2018, pp. 1–8.

[32] N. Lu, B. Ji, and B. Li, "Age-based scheduling: Improving data freshness for wireless real-time traffic," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2018, pp. 191–200.

[33] C. Joo and A. Eryilmaz, "Wireless scheduling for information freshness and synchrony: Drift-based design and heavy-traffic analysis," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2556–2568, Dec 2018.

[34] R. Talak, S. Karaman, and E. Modiano, "Minimizing age-of-information in multi-hop wireless networks," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2017, pp. 486–493.

[35] A. M. Bedewy, Y. Sun, and N. B. Shroff, "The age of information in multihop networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1248–1257, 2019.

[36] R. D. Yates, "The age of information in networks: Moments, distributions, and sampling," *arXiv preprint arXiv:1806.03487*, 2018.

[37] J. Zhong, E. Soljanin, and R. D. Yates, "Status updates through multicast networks," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2017, pp. 463–469.

[38] J. Zhong, R. D. Yates, and E. Soljanin, "Minimizing content staleness in dynamo-style replicated storage systems," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 361–366.

[39] B. Yin, S. Zhang, Y. Cheng, L. X. Cai, Z. Jiang, S. Zhou, and Z. Niu, "Only those requested count: Proactive scheduling policies for minimizing effective age-of-information," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 109–117.

[40] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Transactions on Information Theory*, vol. 63, no. 11, pp. 7492–7508, Nov 2017.

[41] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Age and value of information: Non-linear age case," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 326–330.

[42] M. Klügel, M. H. Mamduhi, S. Hirche, and W. Kellerer, "AoI-penalty minimization for networked control systems with packet loss," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 189–196.

[43] J. Sun, Z. Jiang, B. Krishnamachari, S. Zhou, and Z. Niu, "Closed-form whittles index-enabled random access for timely status update," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1538–1551, 2019.

[44] Y.-P. Hsu, "Age of information: Whittle index for scheduling stochastic arrivals," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 2634–2638.

[45] Z. Jiang, B. Krishnamachari, X. Zheng, S. Zhou, and Z. Niu, "Timely status update in wireless uplinks: Analytical solutions with asymptotic optimality," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3885–3898, 2019.

[46] B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja, *A first course in order statistics*. SIAM, 2008.

[47] R. Nelson, *Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling*. Springer Science & Business Media, 2013.

[48] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Transactions on Information Theory*, in press, 2017.

[49] F. Li, Y. Sang, Z. Liu, B. Li, H. Wu, and B. Ji, "Waiting but not aging: Minimizing age-of-information and maximizing age-based utility under the pull model," *arXiv preprint arXiv:1912.08722*, 2019.

[50] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[51] J. Gittins, K. Glazebrook, and R. Weber, *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[52] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[53] F. Li, J. Liu, and B. Ji, "Combinatorial sleeping bandits with fairness constraints," in *Proceedings of IEEE INFOCOM*, 2019, pp. 1–9.

[54] S. Mannor and O. Shamir, "From bandits to experts: On the value of side-observations," in *Advances in Neural Information Processing Systems*, 2011, pp. 684–692.

[55] S. Caron, B. Kveton, M. Lelarge, and S. Bhagat, "Leveraging side observations in stochastic bandits," in *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, 2012, pp. 142–151.

[56] S. Buccapatnam, F. Liu, A. Eryilmaz, and N. B. Shroff, "Reward maximization under uncertainty: Leveraging side-observations on networks," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 7947–7980, 2017.

[57] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[58] P. Auer and R. Ortner, "UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem," *Periodica Mathematica Hungarica*, vol. 61, no. 1-2, pp. 55–65, 2010.

[59] E. Najm and R. Nasser, "Age of information: The gamma awakening," in *Proceedings of IEEE ISIT*, 2016.

Yu Sang received his B.E. degree in Electronic Information Engineering from University of Science and Technology of China in 2014 and his Ph.D. degree in Computer and Information Sciences from Temple University. His research interests are in wireless networks and cloud computing systems.

Zhongdong Liu is a PhD student in the Department of Computer Science at Virginia Tech. He received his B.S. degree in Mathematics and Applied Mathematics with honor from Northeast Forestry University in 2016. His research interests are in the modeling, analysis, control, and optimization of complex network systems.

Bin Li (S'12–M'16–SM'20) received his B.S. degree in Electronic and Information Engineering in 2005, M.S. degree in Communication and Information Engineering in 2008, both from Xiamen University, China, and Ph.D. degree in Electrical and Computer Engineering from The Ohio State University in 2014. Between 2014 and 2016, he worked as a Postdoctoral Researcher in the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign. He is currently an Assistant Professor in the Department of Electrical, Computer, and Biomedical Engineering at the University of Rhode Island. His research focuses on the intersection of networking, machine learning, and system developments, and their applications in networking for virtual/augmented reality, mobile edge computing, mobile crowd-learning, and Internet-of-Things. He is a senior member of the IEEE and a member of the ACM. He received both the National Science Foundation (NSF) CAREER Award and Google Faculty Research Award in 2020, and ACM MobiHoc 2018 Best Poster Award.

Huasen Wu (S'12-M'14) received his B.S. and Ph.D. degrees from Beihang University, Beijing, China, in 2007 and 2014, respectively. From October 2012 to January 2014, he worked as a research intern in the Wireless and Networking Group, Microsoft Research Asia, and from May 2014 to February 2017, he worked as a postdoctoral researcher in the Department of Computer Science, University of California, Davis. His research interests are in stochastic learning and optimization for wireless networks and recommendation systems.

Bo Ji (S'11-M'12-SM'18) received his B.E. and M.E. degrees in Information Science and Electronic Engineering from Zhejiang University, Hangzhou, China, in 2004 and 2006, respectively, and his Ph.D. degree in Electrical and Computer Engineering from The Ohio State University, Columbus, OH, USA, in 2012. Dr. Ji is an Associate Professor in the Department of Computer Science at Virginia Tech, Blacksburg, VA, USA. Prior to joining Virginia Tech, he was an Associate/Assistant Professor in the Department of Computer and Information Sciences at Temple University from July 2014 to July 2020. He was also a Senior Member of the Technical Staff with AT&T Labs, San Ramon, CA, from January 2013 to June 2014. His research interests are in the modeling, analysis, control, and optimization of computer and network systems, such as wired and wireless networks, large-scale IoT systems, high performance computing systems and data centers, and cyber-physical systems. He currently serves on the editorial boards of the IEEE/ACM Transactions on Networking, IEEE Internet of Things Journal, and IEEE Open Journal of the Communications Society. Dr. Ji is a senior member of the IEEE and a member of the ACM. He is a National Science Foundation (NSF) CAREER awardee (2017) and an NSF CISE Research Initiation Initiative (CRII) awardee (2017). He is also a recipient of the IEEE INFOCOM 2019 Best Paper Award.

Fengjiao Li is a PhD student at the CS department of Virginia Tech. She received her M.S. degrees in Communications and Information System at Beijing University of Posts and Telecommunications in 2015. Prior to that, she received her B.S. degree in Electronical and Information Engineering with honors from Taiyuan University of Technology in 2012. Her current research interests include modeling, analysis, networking and integrating queueing and learning. Her work receives the IEEE INFOCOM 2019 Best Paper Award and also the IEEE INFOCOM 2019 Best In-Session Presentation Award.