Eywa: A General Framework for Scheduler Design and Its Applications to A Family of AoI-Related Problems

Chengzhang Li[†] Shaoran Li[†] Qingyu Liu[†] Y. Thomas Hou[†] Wenjing Lou[†] Sastry Kompella[‡]
[†]Virginia Tech, Blacksburg, VA
[‡]Nexcepta Inc, Gaithersburg, MD

***Abstract*—Age of Information (AoI) is a metric that can be used to measure the freshness of information. Since its inception, there have been active research efforts on designing scheduling algorithms to AoI-related problems. These problems vary in specific AoI-based objectives and network settings. For each problem, typically a custom-designed scheduler was developed. Instead of following the (custom-design) path, we envision and pursue a general framework that can be applied to design a wide range of schedulers to solve AoI-related problems. As a first step toward this vision, we present a general framework—Eywa, that can be applied to construct high-performance schedulers for a family of AoI-related optimization and decision problems, all sharing a common setting of an IoT data collection network. We show how to apply Eywa to solve three important problems: to minimize weighted sum of AoIs, to minimize bandwidth requirement under AoI constraints, and to determine the existence of feasible schedulers to satisfy AoI constraints. We show that for each problem, Eywa can either offer a stronger performance guarantee than the state-of-the-art algorithms or provide new (or general) results that are not available in the literature.**

I. Introduction

Age of Information (AoI) is a metric that can be used to measure the freshness of information. It is defined as the elapsed time period between the present and the time when the information (a sample) was generated. Since the AoI concept was introduced by Kaul *et al.* [1], [2], there has been a growing body of research on designing scheduling algorithms to optimize AoI-related objectives (see, e.g., [3]–[29]).

The optimization objectives in these works include: minimize average/peak AoI (e.g., [4]–[12], [15], [19], [27], [28]), to minimize transmission bandwidth under AoI constraints (e.g., [26]), to determine the existence of feasible schedulers to satisfy AoI constraints (e.g, [24], [25]), and to optimize variants of the AoI metric (e.g., [20]–[22]), among others. Many different network settings (scenarios) have been considered, including IoT data collection (e.g., [10]–[14], [22]–[28]), queuing systems (e.g., [4], [8]), cache systems (e.g., [3], [20]), and so on.

For each of these AoI problems, typically a *custom-designed* scheduler was proposed. With so many custom-designed AoI schedulers around, it is important to explore the intrinsic connections among these problems and proposed solutions and extract some important properties for fundamental understanding. More important, it is desirable to develop a *general framework* that can be applied to design a wide range of schedulers.[1]

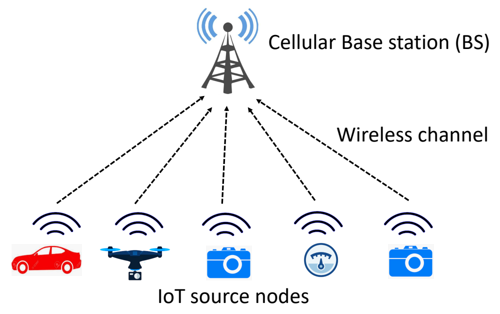


Figure 1: System model: N source nodes collect information and send it to a BS.

In this paper, we make a first attempt to develop such as a general framework—code-named *Eywa*[2]. Eywa considers a data collection network setting (Fig. 1) and can be applied to construct high-performance schedulers for a family of AoI-related optimization and decision problems in this network setting. The problems that we have identified so far include: to minimize sum of AoIs, to minimize bandwidth under AoI constraints, and to determine the existence of feasible schedulers to satisfy AoI constraints. Other problems to which Eywa may be applied will be addressed in a future paper.

The core of Eywa hinges upon the notion of *Almost Uniform Scheduler* (AUS)—a unique type of *cyclic* schedulers where the transmission pattern of each source node is at least *nearly uniform.* Another core idea in Eywa is a clever use of the so-called *step-down* rate vectors, which can be used to construct AUS-based schedulers efficiently. We will elaborate the details of these two ideas in Section II.

The general framework of Eywa consists of three steps. The first step is to transform the original (AoI-based) objective function and/or the constraints to a *rate-based* objective function and *rate-based* constraints. The second step is to find the optimal step-down rate vector by solving an optimization problem with the rate-based objective function and constraints. The third step is to construct an AUS-based scheduler using the optimal step-down rate vector.

[1]The vision of this general framework for AoI research came from co-author Y.T. Hou's early work on Internet QoS scheduling [30] and asymptotic capacity analysis in wireless networks [31], [32].

[2]Eywa is the guiding force of life and deity of Pandora and the Na'vi (from movie Avatar).

We put the Eywa framework in action by solving a family of AoI-related problems in the literature: to minimize sum of AoIs [27]–[29], to minimize bandwidth under AoI constraints [26], and to determine the existence of feasible schedulers to satisfy AoI constraints [24], [25]. We show that for each problem, Eywa can either offer a stronger performance guarantee than state-of-the-art algorithms, or provide new/general results that are not available in the literature.

The remainder of this paper is organized as follows. In Section II, we present the general framework of Eywa and elaborate its three steps in details. In Sections III to V, we apply the Eywa framework to solve several problems. Specifically, in Section III, we apply Eywa to find a scheduler that minimizes the weighed sum of AoIs. In Section IV, we apply Eywa to find schedulers that minimize bandwidth requirement under peak and average AoI constraints, respectively. In Section V, we apply Eywa to solve a decision problem, i.e., to determine the existence of feasible schedulers to satisfy hard/soft AoI deadlines. Section VI summarizes this paper.

II. Eywa: A General Approach

In this section, we present a general approach—code-named Eywa—to construct *high-performance* cyclic schedulers for many AoI optimization problems. By "high-performance" we mean some strong theoretical guarantees, either w.r.t the optimal objective value or some schedulability conditions. The types of AoI optimization problems that Eywa may be applied, include (but not necessarily limited to):

1) **Min-Sum:** Minimize (weighted) sum of AoI;
2) **Min-BW:** Minimize bandwidth requirement under AoI constraints;
3) **Decision Problems:** Determine the existence of feasible schedulers to satisfy AoI constraints.

A **side benefit** of Eywa is that its schedulers follow a "fixed" cyclic transmission pattern and do not require the BS to convey its scheduling decisions in every time slot. This will *lower communication overhead* in the control channel as well as *reduce delay* (the time to send the scheduling decision from the BS to each source).

A. System Model

Consider a data collection network where N source nodes collect information from the environment and send it to a base station (BS) through a shared wireless channel, as shown in Fig. 1. Tables I lists key notations used in this paper.

We assume time is slotted, and each source node takes a sample at the beginning of each time slot. If scheduled for transmission, a source will send its freshest sample (collected at the beginning of the time slot) to the BS.

Denote W as the transmission bandwidth of the wireless network. The wireless channel allows at most W samples to be transmitted in each time slot, and the BS scheduler needs to decide which samples will be chosen for transmission.

Table I: Notations

Symbol	Definition
	General Notations
$A_i(t)$	AoI for sample from source node i at the BS at time t
W	Transmission bandwidth.
c	Cycle length for a cyclic scheduler
d_i	AoI deadline for source node i
N	Number of source nodes in the network
$\pi_i(t)$	Scheduling decision for source i at time t
p_i	Packet loss rate for source i
$p_{\max}$	The largest one among all p_i's
$q_i(t)$	Channel indicator for source i at time t
r_i	Average transmission rate rate w.r.t. source node i
$U_i(t)$	Generation time of the most recent sample at the BS from source node i at time t
	Notations for Min-Sum
$\bar{A}_i$	Long-term average AoI for source i
$\bar{A}$	Weighted sum of all $\bar{A}_i$'s
$\bar{A}_{\text{LB}}$	A lower bound for $\bar{A}$
$\bar{A}^*$	Minimum $\bar{A}$ for all schedulers
$\boldsymbol{r}'$	Optimal solution to OPT-LB
$\hat{\boldsymbol{r}}^*$	Optimal solution to OPT-Hat
w_i	Weight for source i
	Notations for Min-BW
α_i	Threshold for average AoI for source i
$\boldsymbol{\alpha}$	A vector denoting $[\alpha_1, \alpha_2, \cdots, \alpha_N]$
W^*	Minimum W for all feasible schedulers
d_i	Threshold for peak AoI for source i
$\boldsymbol{d}$	A vector denoting $[d_1, d_2, \cdots, d_N]$
	Notations for Decision Problems
d_i	(Hard or soft) AoI deadline for source i
$\boldsymbol{d}$	A vector denoting $[d_1, d_2, \cdots, d_N]$
ϵ_i	Violation tolerance rate for source node i's AoI deadline
$\boldsymbol{\epsilon}$	A vector denoting $[\epsilon_1, \epsilon_2, \cdots, \epsilon_N]$
$l(\boldsymbol{d})$	System load for $\boldsymbol{d}$ under hard AoI deadlines
$l(\boldsymbol{d}, \boldsymbol{\epsilon}, \boldsymbol{p})$	System load for $(\boldsymbol{d}, \boldsymbol{\epsilon}, \boldsymbol{p})$ under soft AoI deadlines
p_i	Packet loss rate for source node i
$\boldsymbol{p}$	A vector denoting $[p_1, p_2, \cdots, p_N]$

For scheduler π, we denote $\pi_i(t) \in \{0, 1\}$ as the scheduling decision at time t w.r.t. each source i $(i = 1, 2, \cdots, N)$, i.e.,

$$\pi_i(t) = \begin{cases} 1, & \text{if source } i \text{ is scheduled to transmit at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

For any scheduler π, at any $t = 0, 1, 2, \cdots$ we have

$$\sum_{i=1}^{N} \pi_i(t) \le W. \tag{2}$$

The long-term average *transmission rate*, r_i for source i, can be defined as:

$$r_i = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \pi_i(t). \tag{3}$$

Then we have

$$r_i \le 1, \quad \text{for } i = 1, 2, \cdots, N, \tag{4a}$$

$$\sum_{i=1}^{N} r_i \leq W. \tag{4b}$$

We consider unreliable wireless channel in our model. We assume there is a (fixed) packet loss rate (due to transmission failure) for each source node i, which we denote as p_i. With the presence of packet loss rate, the transmission from a source may not always be successful. Denote $q_i(t)$ as a binary indicator for whether or not the transmission from source i at time t (if scheduled) is successful, i.e.,

$$q_i(t) = \begin{cases} 1, & \text{if transmission from } i \text{ is successful at } t \\ & \text{(should } i \text{ be scheduled for transmission),} \\ 0, & \text{otherwise.} \end{cases}$$

Then we have

$$\mathbb{P}\{q_i(t)=0\} = p_i,$$
$$\mathbb{P}\{q_i(t)=1\} = 1-p_i.$$

The AoI of source i (at the BS) at time t, denoted by $A_i(t)$, is defined as the elapsed time between the current time t and $U_i(t)$— the sample's generation time at its source, i.e.,

$$A_i(t) = t - U_i(t). \tag{5}$$

If the BS does not receive a sample from source i in time t (i.e., $\pi_i(t) = 0$ or $q_i(t) = 0$), then by definition of AoI, at time $(t+1)$ we have $A_i(t+1) = A_i(t) + 1$. Otherwise, the AoI will drop to 1. We have,

$$A_i(t+1) = \begin{cases} 1, & \text{if } \pi_i(t)\cdot q_i(t) = 1, \\ A_i(t)+1, & \text{otherwise.} \end{cases} \tag{6}$$

B. Main Idea: Almost Uniform Scheduler

Eywa focuses on a special class of cyclic schedulers,[3] which we call *Almost Uniform Schedulers (AUSs)*. The concept of AUS first appeared in our previous work [25] for the special case of unit channel bandwidth (i.e., $W = 1$). The AUS that we present here is general and can be applied to arbitrary channel bandwidth ($W \geq 1$).

Denote τ_i^k as the time slot when the k-th sample ($k = 1, 2, \cdots$) from source i is scheduled for transmission, i.e., $\pi_i(\tau_i^k) = 1$. Denote T_i^k as the time interval between the k-th and the $(k+1)$-th transmission for source i, i.e.,

$$T_i^k = \tau_i^{k+1} - \tau_i^k. \tag{7}$$

Then, AUS is defined as the following.

Definition 1 *A cyclic scheduler π is an AUS if for each source i there exists an integer b_i such that we have either $T_i^k = b_i$ or $T_i^k = b_i + 1$ for any $k \geq 1$.*

[3]A cyclic scheduler repeats the same scheduling decisions every c time slots (where c is the cycle length). More formally, a scheduler π is cyclic if there exists an integer c such that $\pi_i(t) = \pi_i(t+c)$ for all i's and $t \geq 0$. Cyclic schedulers do not require the BS to convey its scheduling decisions in every time slot, which will i) lower communication overhead in the control channel, and ii) reduce the delay for sending the scheduling decision to the sources.

For example, consider four sources A, W, C, and D. Denote "$(\cdots)$" as scheduling decisions for one cycle. Then when $W = 1$, the scheduler

$$(ABABCABDABC)$$

is an AUS with $b_A = 2$, $b_W = 2$, $b_C = 5$, and $b_D = 11$. Note that for source C, $b_C = 5$ because the interval length between the last transmission in this cycle and the first transmission in the next cycle is 5.

As the second example, when $W = 2$ [where for each time slot, 2 samples from 2 source nodes (a column in the scheduling bracket) are transmitted to the BS], the scheduler

$$\begin{pmatrix} ACBACBBADBA \\ BADBAACBACB \end{pmatrix}$$

is an AUS with $b_A = 1$, $b_W = 1$, $b_C = 2$, and $b_D = 5$.

We also identify a special case of AUS, which we call *Exact Uniform Scheduler* (EUS). We say an AUS is an EUS if for each source i, we have $T_i^k = b_i$ for all $k \geq 1$. That is, each source i is periodically scheduled for transmission with an *exact* period of b_i.

C. Eywa: Complete Procedure

Eywa is designed to solve a family of AoI optimization problems and decision problems, including Min-Sum, Min-BW, and the decision problems that we listed in the beginning of Section II. Although these problems have different objectives and constraints, they all conform to the system model as described in Section II-A. As such, we have discovered the following framework (Eywa) that can be used to custom design a high-performance AUS-based scheduler for each of these problems. Our proposed Eywa framework is given in Algorithm 1.

Algorithm 1 A General Framework of Eywa

```
1: Transform objective function and constraints: If the original objective function and/or the constraints are not rate-based (e.g., AoI-based), then transform them to a rate-based objective function and rate-based constraints.
2: Find optimal transmission rates: Find the optimal transmission rates by solving an optimization problem (OPT-SD) with the rate-based objective function.
3: Construct an AUS-based scheduler: Construct an AUS-based scheduler using the optimal transmission rates.
```

Eywa (Algorithm 1) consists of three steps. In the rest of this section, we elaborate the details in each step.

Step 1: Transform objective function and constraints The essence of Eywa is to design an AUS-based scheduler based on each source i's transmission rate r_i. We start with the objective function. Since the original objective function may not be rate-based (e.g., AoI-based), the first step of Eywa is to transform the original objective function to a *rate-based* objective function.

We denote $J_o(\cdot)$ as the original objective function that we want to minimize.[4] For example, $J_o(\cdot)$ could be a function of AoIs, i.e., $J_o(\cdot) = J_o\big(A_1(t),\, A_2(t), \cdots, A_N(t)\big)$ as in the Min-Sum problem. We will transform $J_o(\cdot)$ to a rate-based objective function, which we denote as $J_R(r_1, r_2, \cdots, r_N)$. The goal of this transformation is to minimize the distance (gap) between the two functions such that

$$J_o(\cdot) \approx J_R(r_1, r_2, \cdots, r_N). \tag{8}$$

Similarly, if any of the original constraints are not rate-based, then transform them into rate-based constraints.

Step 2: Find optimal transmission rates The goal of Step 2 is to find the transmission rates r_i's that optimize $J_R(r_1, r_2, \cdots, r_N)$. In particular, we solve a special optimization problem (OPT-SD) that will ensure the optimal rates (r_i's) can be readily used for the design of AUS-based schedulers in Step 3.

Denote the transmission rate vector $\boldsymbol{r} = [r_1\ r_2\ \cdots\ r_N]$. We define a sorted rate vector $\boldsymbol{\gamma} = [\gamma_1\ \gamma_2\ \cdots\ \gamma_N]$ as:

$$\boldsymbol{\gamma} = \text{sort}(\boldsymbol{r}), \tag{9}$$

where sort$(\cdot)$ is a function that sorts the elements of the input vector in a *non-increasing* order. We are interested in $\boldsymbol{\gamma}$'s with a special *"step-down"* property, which we define as follows:

Definition 2 *A sorted vector $\boldsymbol{\gamma}$ is step-down if $\gamma_i/\gamma_{i+1} \in \mathbb{N}^*$ for all $\gamma_i < 1$ and $i < N$.*

In a step-down vector, the leading elements in $\boldsymbol{\gamma}$ can be 1's, while the remaining elements (with $\gamma_i < 1$) must satisfy $\gamma_i/\gamma_{i+1} \in \mathbb{N}^*$. For example, $\boldsymbol{\gamma} = [1\ 1\ \frac{4}{5}\ \frac{4}{5}\ \frac{2}{5}\ \frac{1}{5}]$ is step-down.

We now solve the following optimization problem:

$$\begin{aligned}
\text{OPT-SD:}\ \min_{\boldsymbol{r}}\quad & J_R(r_1, r_2, \cdots, r_N)\\
\text{s.t.}\quad & \boldsymbol{\gamma} = \text{sort}(\boldsymbol{r}),\\
& \boldsymbol{\gamma} \text{ is step-down},\\
& \sum_{i=1}^{N} \gamma_i = W,\\
& \text{Additional problem-specific rate-based}\\
& \text{constraints},\\
& 0 < r_i \le 1, \quad i = 1, 2, \cdots, N.
\end{aligned}$$

Note that we purposely make $\sum_{i=1}^{N} \gamma_i = W$ (instead of $\le W$) so that the rates will fit perfectly into a discrete slot-based scheduler.

Denote the optimal solution to OPT-SD as $\boldsymbol{r}^* = [r_1^*\ r_2^*\ \cdots\ r_N^*]$. After solving OPT-SD and obtaining $\boldsymbol{r}^*$, we will use $\boldsymbol{r}^*$ to construct an AUS in Step 3.

Step 3: Construct an AUS-based Scheduler The goal of Step 3 is to construct an AUS-based scheduler using the optimal rate vector $\boldsymbol{r}^*$ that we find in Step 2.

We first consider the special case where $W = 1$ and present an algorithm on how to construct an AUS-based scheduler with $\boldsymbol{r}^*$ (with $\sum_{i=1}^{N} r_i^* = 1$) and $\boldsymbol{\gamma}^* = \text{sort}(\boldsymbol{r}^*)$ is step-down. We use an example to show how our algorithm works. The complete pseudocode is given in Algorithm 2.[5]

Example 1. Consider six sources A, B, C, D, E, F and $\boldsymbol{r}^* = [\frac{1}{10}\ \frac{3}{10}\ \frac{1}{10}\ \frac{3}{10}\ \frac{1}{10}\ \frac{1}{10}]$. We have $\boldsymbol{\gamma}^* = [\frac{3}{10}\ \frac{3}{10}\ \frac{1}{10}\ \frac{1}{10}\ \frac{1}{10}\ \frac{1}{10}]$, which is step-down, and we have $\sum_{i=1}^{N} r_i^* = 1$. We will show how to construct an AUS for this $\boldsymbol{r}^*$.

Denote c^{AUS} as the cycle length of the AUS we are going to construct, and N_i as the number of slots allocated to the i-th source (indexed w.r.t. $\boldsymbol{\gamma}^*$) in c^{AUS}. Clearly, we have $N_i = \gamma_i^* \cdot c^{\text{AUS}}$.

In this example, the sequence of the sources in $\boldsymbol{\gamma}^*$ is B-D-A-C-E-F. Since the smallest rate among the γ_i^*'s is $1/10$ (for source F), we set the cycle length of the AUS to $c^{\text{AUS}} = 10$. Then we have $N_B = 3$, $N_D = 3$, $N_A = 1$, $N_C = 1$, $N_E = 1$, and $N_F = 1$.

Let's start with the first source B. Within a cycle with $c^{\text{AUS}} = 10$ slots, there are $N_B = 3$ slots allocated to source B. Ideally, we want the 3 slots to be evenly spaced in 10 slots. But this is not possible under 10 slots. So we will *add* the minimum number of slots (which is 2) to the cycle to make this happen. This corresponds to adding $(\lceil \frac{c^{\text{AUS}}}{N_B} \rceil \cdot N_B - c^{\text{AUS}})$ extra slots. These extra slots will be removed in the end when we change the EUS scheduler to an AUS scheduler. With a cycle of 12 slots ($c = 12$), we can place source B evenly as follows:

π	$(B$	□	□	□	B	□	□	□	B	□	□	□$)$
t	1	2	3	4	5	6	7	8	9	10	11	12

Now we we consider the second source node—D. With $N_D = 3$, we can place D evenly among the empty slots left by B. Among the empty slots ($t = 2, 3, 4, 6, 7, 8, 10, 11, 12$), we select the time slots with the smallest number of elapsed time slots following its predecessor node B, i.e., $t = 2, 6, 10$. We allocate all of them ($t = 2, 6, 10$) to source D, and we have:

π	$(B$	D	□	□	B	D	□	□	B	D	□	□$)$
t	1	2	3	4	5	6	7	8	9	10	11	12

The next source node to consider is A. Since $N_A = 1$, we only need to allocate 1 empty slot to source A. Among all the remaining empty slots, we want to find one with the shortest distance (in number of time slots) to A's predecessor nodes, starting from the oldest one (node B). The time slots with the shortest distance to B are: $t = 3, 7, 11$. Among them ($t = 3, 7, 11$), we select the time slots with the shortest distance to D and we still have: $t = 3, 7, 11$. Since we only need one slot, we choose $t = 3$ to A.

π	$(B$	D	A	□	B	D	□	□	B	D	□	□$)$
t	1	2	3	4	5	6	7	8	9	10	11	12

[4]We focus on a minimization problem here to concretize our discussions. Note that a maximization problem can be easily reformulated as a minimization problem, i.e., minimize the negative of the objective function.

[5]This algorithm is based on our previous work in [25] which checks the schedulability of a given set of AoI deadlines and violation tolerance rates. That problem is referred as the "Max-Region under soft AoI deadlines" problem in this paper.

Algorithm 2 Construct an AUS-based Scheduler with $W = 1$

```
Input: A step-down vector $\gamma^{(1)}$ with $\sum_{i=1}^{N} \gamma_i^{(1)} = 1$.
Output: An AUS-based scheduler $\pi^{(1)}$.
 1: Set $c^{\text{AUS}} = 1/\gamma_N^{(1)}$ and $N_i = \gamma_i^{(1)} \cdot c^{\text{AUS}}$ for each $1 \le i \le N$.
 2: Initialize an EUS cycle with length $c = \lceil \frac{c^{\text{AUS}}}{N_1} \rceil \cdot N_1$ slots.
 3: Set $b_1 = c/N_1$. Allocate time slots $t = 1, b_1 + 1, 2b_1 + 1, \cdots, c - b_1 + 1$ in the EUS cycle to the first source in $\gamma$.
 4: for source $i = 2, 3, \cdots, N$ in $\gamma$ do
 5:     Let $\mathcal{S}_0$ be the set of empty time slots in the EUS cycle.
 6:     for $j = 1, 2, \cdots, i - 1$ do
 7:         Let $\mathcal{S}_j$ be the subset of $\mathcal{S}_{j-1}$ containing time slots with the shortest distance after a source $j$.
 8:     end for
 9:     Let $\tau$ be the first slot in $\mathcal{S}_{i-1}$. Set $b_i = c/N_i$. Allocate slots $t = \tau, b_i + \tau, 2b_i + \tau, \cdots, c - b_i + \tau$ to source $i$.
10: end for
11: Remove the $(c - c^{\text{AUS}})$ unassigned time slots in the EUS cycle and output the resulting scheduler $\pi^{(1)}$.
```

The next source node to consider is C. Since $N_C = 1$, we only need to allocate 1 empty slot to source C. Again, among all the remaining empty slots, our goal is to find one with the shortest distance to C's predecessor nodes (B, D, A), starting from the oldest one (node B). With respect to each B in the cycle, the time slots with the shortest distance (on the right side) are: $t = 7, 11$. Between the two ($t = 7, 11$), we want to select the time slots with the shortest distance to a D and we still have: $t = 7, 11$. Now between the two ($t = 7, 11$), we want to select one time slot with the shortest distance to A and we only have one choice, which is $t = 7$. We allocate $t = 7$ to C. Note that the above process can be put into an iteratively procedure (as shown in Steps 4-8 in Algorithm 2).

π	$(B$	D	A	$\square$	B	D	C	$\square$	B	D	$\square$	$\square)$
t	1	2	3	4	5	6	7	8	9	10	11	12

Following the same token, we allocate $t = 11$ to source E and $t = 4$ to source F and we have:

π	$(B$	D	A	F	B	D	C	$\square$	B	D	E	$\square)$
t	1	2	3	4	5	6	7	8	9	10	11	12

Clearly, outcome of our recursion will ensure the scheduler is an EUS throughout the process. In the last step, we remove the empty slots (meticulously placed in the cycle) in the EUS scheduler and obtain an AUS scheduler $\pi^{(1)}$. We have:

$\pi^{(1)}$	$(B$	D	A	F	B	D	C	B	D	$E\ )$
t	1	2	3	4	5	6	7	8	9	10

■

The ideas of Example 1 are stated as pseudocode in Algorithm 2. The time complexity of Algorithm 2 is $O(N^2/\gamma_N)$.

Now we extend Algorithm 2 to the general case when $W \geq 1$. The problem is to construct an AUS scheduler $\pi^{(M)}$ for a step-down vector γ with $\sum_{i=1}^{N} \gamma_i = W$.

Denote M as the number of elements in γ such that $\gamma_i = 1$. For each source node $1, 2, \cdots, M$, we allocate every slot to

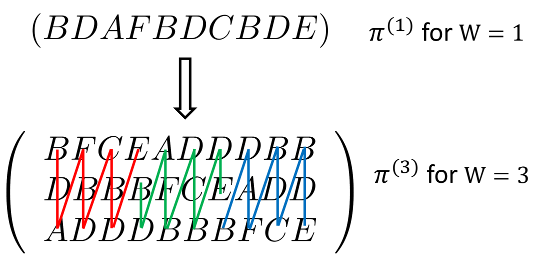


Figure 2: An example for Zigzag packing

them and get an EUS $\pi^{(M)}$ with bandwidth M.

For other source nodes, denote $\gamma' = [\gamma_{M+1}\ \gamma_{M+2}\ \cdots\ \gamma_N]$, and the problem is to construct an AUS $\pi^{(W-M)}$ for γ' with bandwidth $(W - M)$. To solve this problem, the idea is to first construct an AUS-based scheduler $\pi^{(1)}$ using Algorithm 2 for the special case $W = 1$. Since Algorithm 2 requires $\sum_{i=1}^{N} \gamma_i^{(1)} = 1$, we have to scale down γ by a factor $(W-M)$, i.e., $\gamma^{(1)} = \gamma'/(W - M)$. Clearly, we have $\sum_{i=1}^{N} \gamma_i^{(1)} = 1$, and we can use Algorithm 2 to construct $\pi^{(1)}$.

With $\pi^{(1)}$ as the basic building block, we use a procedure called *Zigzag packing* to construct the final AUS scheduler $\pi^{(W-M)}$ by packing scheduler $\pi^{(1)}$ column-by-column into a frequency-time grid exactly W times. We use an example to show how it works.

Example 2. Consider six sources A, B, C, D, E, F, $W = 3$, and $\boldsymbol{r} = [\frac{3}{10}\ \frac{9}{10}\ \frac{3}{10}\ \frac{9}{10}\ \frac{3}{10}\ \frac{3}{10}]$. We have $\gamma = [\frac{9}{10}\ \frac{9}{10}\ \frac{3}{10}\ \frac{3}{10}\ \frac{3}{10}\ \frac{3}{10}]$, which is step-down, $\sum_{i=1}^{N} \gamma_i = 3$, and $M = 0$. We now show how to construct an AUS scheduler $\pi(3)$ for this γ.

As the first step, we scale down γ by a factor 3, and get $\gamma^{(1)} = [\frac{3}{10}\ \frac{3}{10}\ \frac{1}{10}\ \frac{1}{10}\ \frac{1}{10}\ \frac{1}{10}]$, with $\sum_{i=1}^{N} \gamma_i^{(1)} = 1$. Then we use Algorithm 2 to find AUS $\pi^{(1)}$ with $\gamma^{(1)}$ as input, which is identical to Example 1. To construct $\pi^{(3)}$, we roll $\pi^{(1)}$ into $\pi^{(3)}$'s frequency-time structure column-by-column 3 times, as shown in Fig. 2, and obtain the AUS scheduler $\pi^{(3)}$. ■

After obtaining $\pi^{(M)}$ and $\pi^{(W-M)}$, we can combine them and get our target AUS $\pi^{(W)}$ with bandwidth W. A pseudocode on how to construct an AUS-based scheduler is given in Algorithm 3. This completes Step 3 of Eywa.

Now we have presented all details for the three steps of our proposed Eywa framework. In the rest of this paper, we put this framework in action by solving a family of AoI optimization and decision problems.

III. Application to the Min-Sum Problem

In this section, we show how to use Eywa to solve Min-Sum problem that we mentioned in the beginning of Section II. Then we will compare Eywa's scheduler with the state-of-the-art.

Algorithm 3 Construct an AUS-based Scheduler with $W \geq 1$

```
Input: A step-down vector $\boldsymbol{\gamma}$ with $\sum_{i=1}^{N} \gamma_i = W$.
Output: An AUS-based scheduler $\pi^{(W)}$.
1: For each source node $1, 2, \cdots, M$, allocate every slot to them and get an EUS $\pi^{(M)}$ with bandwidth $M$.
2: For other source nodes, set $\boldsymbol{\gamma}' = [\gamma_{M+1}\ \gamma_{M+2}\ \cdots\ \gamma_N]$. Set $\boldsymbol{\gamma}^{(1)} = \boldsymbol{\gamma}'/(W - M)$. Construct an AUS scheduler $\pi^{(1)}$ by Algorithm 2 with $\boldsymbol{\gamma}^{(1)}$ as input.
3: If $W - M > 1$, use Zigzag packing to construct an AUS scheduler $\pi^{(W-M)}$ with bandwidth $(W - M)$.
4: Combine $\pi^{(M)}$ and $\pi^{(W-M)}$, and get AUS $\pi^{(W)}$.
```

A. Problem Statement

The network setting for Min-Sum is the same as that in Section II-A. We consider a general $W \geq 1$. The long-term average AoI for source node i is defined as:

$$\bar{A}_i = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} A_i(t). \tag{10}$$

Denote w_i as the weight of source node i. The weighted sum of long-term average AoI over all source nodes, denoted as $\bar{A}$, is defined as:

$$\bar{A} = \sum_{i=1}^{N} w_i \bar{A}_i. \tag{11}$$

The objective for the Min-Sum problem is to design a scheduler π that can minimize $\bar{A}$.

B. A Lower Bound

To date, there is no known polynomial time optimal scheduler for the Min-Sum problem in the literature. This is partially due to p_i (packet loss probability) associated for each source, which make it hard to prove optimality. As such, a lower bound for the objective $\bar{A}$ is important because it can serve as a performance benchmark (in place of the optimal objective).

In [3], the authors showed that for each source i, we have

$$w_i \bar{A}_i \geq \frac{w_i}{2(1-p_i)r_i} + \frac{w_i}{2}. \tag{12}$$

Therefore, $\min \sum_{i=1}^{N} w_i \bar{A}_i \geq \min \sum_{i=1}^{N} (\frac{w_i}{2(1-p_i)r_i} + \frac{w_i}{2})$. So we solve the following optimization problem:

$$\begin{aligned} \text{OPT-LB:}\ \min_{\boldsymbol{r}} \quad & \sum_{i=1}^{N} \frac{w_i}{2(1-p_i)r_i} \\ \text{s.t.} \quad & \sum_{i=1}^{N} r_i \leq W, \\ & 0 < r_i \leq 1, \quad i = 1, 2, \cdots, N. \end{aligned}$$

Denote $\boldsymbol{r}^{\text{LB}} = [r_1^{\text{LB}}, r_2^{\text{LB}}, \cdots, r_N^{\text{LB}}]$ as the optimal solution to OPT-LB. Since OPT-LB is a convex optimization problem, we can get $\boldsymbol{r}'$ easily. Then, by (12), a lower bound of $\bar{A}$ is given by

$$\bar{A}_{\text{LB}} = \sum_{i=1}^{N} \frac{w_i}{2(1-p_i)r_i^{\text{LB}}} + \sum_{i=1}^{N} \frac{w_i}{2}. \tag{13}$$

C. Eywa: Step 1—Transform Objective Function and Constraints

In this and the following two sections, we show how to apply Eywa framework to find a high-performance scheduling solution to the Min-Sum problem. As presented in Section II-C, the first step of Eywa is to transform the AoI-based objective function $J_o\big(A_1(t), A_2(t), \cdots, A_N(t)\big) = \sum_{i=1}^{N} w_i \bar{A}_i$ to a rate-based objective function $J_R(r_1, r_2, \cdots, r_N)$. For the Min-Sum problem, we will approximate $\bar{A}_i$ as a rate-based function $q_i(r_i)$, i.e., $\bar{A}_i \approx q_i(r_i)$ for each i. Then we will have $J_R(r_1, r_2, \cdots, r_N) = \sum_{i=1}^{N} w_i \cdot q_i(r_i)$.

Consider an AUS π. To make a connection between $\bar{A}_i$ and $q_i(r_i)$, we propose to *decompose* the AUS scheduler into *two mini EUS* schedulers, for which we can calculate the AoI easily and express it as a rate-based function. We will use an example to show how this work.

Example 3. Consider an AUS π with $W = 1$:

$$\pi = (BDAFBDCBDE).$$

We use source B as an example. To find $\bar{A}_B$, we decompose π into two mini EUSs. We consider the simple case where packet loss rate $p_B = 0$ first.

Since we are only interested in calculating source B's average age, $\bar{A}_B$, we focus on the time slots that source B are scheduled for transmission, i.e.,

$$\pi^0 = (B\square\square\square B\square\square B\square\square).$$

In scheduler π^0, we notice that there are three elapsed intervals w.r.t. source B in the cycle: 4, 3, and 3. So we can decompose π^0 into two EUSs π^1 and π^2 as following:

$$\pi^1 = (B\square\square\square), \quad \pi^2 = (B\square\square B\square\square).$$

Denote $\bar{A}_B^1$ and $\bar{A}_B^2$ as the average AoIs under EUS π^1 and π^2, respectively. Clearly, we have $\bar{A}_B^1 = \frac{5}{2}$ (average of 1, 2, 3, and 4) and $\bar{A}_B^2 = 2$ (average of 1, 2, 3). Denote $q_B(r_B)$ as the weighted average (proportional to mini-cycle length) AoI of two mini EUS π^1 and π^2. Since there are 4 slots in π^1 and 6 slots in π_2, $q_B(r_B)$ is given by

$$q_B(r_B) = \frac{4 \cdot \bar{A}_B^1 + 6 \cdot \bar{A}_B^2}{4+6} = \frac{4 \cdot \frac{5}{2} + 6 \cdot 2}{10} = \frac{11}{5}. \quad \blacksquare$$

The general case where $p_B \geq 0$ can be easily generalized by taking expectations.

For a general π with rate r_i and $p_i = 0$ for each i, we now show how to find $q_i(r_i)$ based on the idea presented in Example 3. We first focus on the time slots that source i are scheduled for transmission and get π^0. Under π^0, the interval length for source i can only be either $\lfloor \frac{1}{r_i} \rfloor$ or $\lfloor \frac{1}{r_i} \rfloor + 1$, where $\lfloor \cdot \rfloor$ is the floor function. Then we decompose π^0 into two mini EUSs π^1 and π^2. Denote $q_i(r_i)$ as the weighted average (proportional to mini-cycle length) AoI of π^1 and π^2. It is not hard to derive the following expression:

$$q_i(r_i) = \lfloor \frac{1}{r_i} \rfloor + 1 - \frac{r_i}{2}(\lfloor \frac{1}{r_i} \rfloor^2 + \lfloor \frac{1}{r_i} \rfloor), \quad (\text{for } p_i = 0). \tag{14}$$

For the general case where $p_i \geq 0$, it can be shown (by taking expectations) that

$$q_i(r_i) = \frac{1+p_i}{1-p_i} \cdot \big(\lfloor \frac{1}{r_i} \rfloor + \frac{1}{2} - \frac{r_i}{2}(\lfloor \frac{1}{r_i} \rfloor^2 + \lfloor \frac{1}{r_i} \rfloor)\big) + \frac{1}{2}. \quad (15)$$

Note that when packet loss rate $p_i = 0$, $q_i(r_i)$ is exactly equal to $\bar{A}_i$. But when $p_i > 0$, $q_i(r_i)$ will be slightly greater than $\bar{A}_i$. This is because there is a slight relaxation for AoI when we decompose AUS π^0 into two mini EUSs π^1 and π^2.

D. Eywa: Step 2—Find Optimal Transmission Rates

The second step of Eywa is to find the optimal rate vector $\boldsymbol{r}^*$. That is, we need to solve OPT-SD, with objective $J_R(r_1, r_2, \cdots, r_N) = \sum_{i=1}^N w_i \cdot q_i(r_i)$. Note that there are no additional constraints in OPT-SD for the Min-Sum problem.

Due to the complexity of the rate function $q_i(r_i)$ in (15), it is too difficult to solve OPT-SD directly. So we present an algorithm that can find a (provably) near-optimal solution to OPT-SD.

Our proposed near-optimal solution is based on $\boldsymbol{r}^{\text{LB}}$, the optimal solution to OPT-LB. Since $\boldsymbol{r}^{\text{LB}}$ may not be a feasible solution to OPT-SD, we propose to find a β $(0 < \beta \leq 1)$ and $\boldsymbol{r}$, such that $\boldsymbol{r} \geq \beta \cdot \boldsymbol{r}^{\text{LB}}$ (i.e., $r_i \geq \beta \cdot r_i^{\text{LB}}$ for each $i = 1, 2, \cdots, N$) and $\boldsymbol{r}$ is step-down and feasible to OPT-SD. Clearly, the greater the β is, the closer the $\boldsymbol{r}$ and $\boldsymbol{r}^{\text{LB}}$ will be. So we want to solve the following optimization problem:

$$\begin{aligned} \text{OPT-}\beta\text{:} \max_{\boldsymbol{r},\beta} \quad & \beta \\ \text{s.t.} \quad & \boldsymbol{\gamma} = \text{sort}(\boldsymbol{r}), \\ & \boldsymbol{\gamma} \text{ is step-down}, \\ & \beta \cdot r_i^{\text{LB}} \leq r_i \leq 1, \ i = 1, 2, \cdots, N, \\ & \sum_{i=1}^N \gamma_i = W. \end{aligned}$$

Denote $\boldsymbol{r}^*$ and β^* as the optimal solution to OPT-β. Clearly, $\boldsymbol{r}^*$ is feasible to OPT-SD, which we will use to construct an AUS-based scheduler in Step 3. An algorithm (based on bisection and dynamic programming) to solve OPT-β is given in Appendix A.

E. Eywa: Step 3—Construct AUS

The third step of Eywa is to construct AUS based on $\boldsymbol{r}^*$, which is exactly the same as what we have done in Section II-C.

F. Performance and Comparison with State-of-the-Art

In this section, we show that Eywa offers a strong theoretical performance guarantee to the Min-Sum problem. Denote $p_{\max}$ as the maximum packet loss rate among all source nodes, i.e.,

$$p_{\max} = \max_{i=1,2,\cdots,N} p_i.$$

Denote $\bar{A}^*$ as the optimal (unknown) objective value for the Min-Sum problem. We have the following theorem that guarantees the performance of $\bar{A}$—the objective value obtained by Eywa.

Theorem 1 *When $p_{\max} \leq 0.807$, the objective value obtained by Eywa satisfies:*

$$\bar{A} \leq (1 + p_{\max}) \log_2 e \cdot \bar{A}^*. \quad (16)$$

A proof of Theorem 1 is given in Appendix B.

In practice, 4G LTE and 5G NR (eMBB) use link adaption algorithms to make sure the BLER (packet loss rate) is about 10% for each user [33]. For $p_{\max} = 0.12$, we have $\bar{A} \leq 1.62 \cdot \bar{A}^*$ under Eywa.

Table II compares the performance guarantees offered by Eywa and the state-of-the-art algorithms for the Min-Sum problem. When $W = 1$, three solutions were proposed ([27], [29](Ch. 3)): the stationary randomized scheduler, the max-weight scheduler, and the Whittle's index scheduler. We can see that in practice (when $p_{\max} = 0.12$), Eywa offers a tighter bound than the other three schedulers. When $W \geq 2$, the max-weight scheduler and the Whittle's index scheduler are not available in the literature, while the stationary randomized scheduler was studied in [28] and [29] (Ch. 4). Again, we can see that in practice (when $p_{\max} = 0.12$), Eywa offers a tighter bound than the stationary randomized scheduler.

In addition to offering a stronger performance guarantee, Eywa also has a lower overhead and latency than the state-of-the-art schedulers, as we discussed in the beginning of Section II.

IV. Application to the Min-BW Problem

In this section, we show how to apply Eywa to solve the Min-BW problem. Recall that the Min-BW problem (see Section II) is to minimize the uplink bandwidth requirement under some AoI constraints. Specifically, we study the problems under two AoI constraints: i) Min-BW under *peak* AoI constraints, and ii) Min-BW under *average* AoI constraints. The first problem was studied in [26]. We show that Eywa can offer a stronger performance guarantee (a tighter bound) than that in [26]. The second problem has not yet been studied in the literature and thus our scheduler (by applying Eywa) is the first known solution to this problem.

A. Min-BW Under Peak AoI Constraints

1) Problem Statement: The network setting is the same as that in Section II-A. For this problem, we assume there is a deadline (denoted as $d_i \in \mathbb{N}^*$) constraint for the peak AoI of each source $i = 1, 2, \cdots, N$. That is,

$$A_i(t) \leq d_i, \ \text{for all } t \geq 0. \quad (17)$$

Since the above constraint is a deterministic one and applies to all $t \geq 0$ and all $i = 1, 2, \cdots, N$, it can hold only when $p_i = 0$ for $i = 1, 2, \cdots, N$. Therefore, we assume $p_i = 0$ for all i's in this problem. The objective is to find a scheduler π that minimizes bandwidth W, such that (17) is satisfied.

Table II: Comparison between Eywa and state-of-the-art for the Min-Sum problem.

Setting	Algorithm	Performance Guarantee	When $p_{\max} = 0.12$
$W = 1$	Eywa	$\bar{A} \leq (1 + p_{\max}) \log_2 e \cdot \bar{A}^*$	$\bar{A} \leq 1.62 \cdot \bar{A}^*$
	Stat. Rand. [27], [29] (Ch. 3)	$\bar{A} \leq 2 \cdot \bar{A}^*$	$\bar{A} \leq 2 \cdot \bar{A}^*$
	Max-Weight [27], [29] (Ch. 3)	$\bar{A} \leq 2 \cdot \bar{A}^*$	$\bar{A} \leq 2 \cdot \bar{A}^*$
	Whittle's Index [27], [29] (Ch. 3)	$\bar{A} \leq 4(\frac{\sqrt{2}}{1-p_{\max}} + \frac{1}{\sqrt{2}})^2 \cdot \bar{A}^*$	$\bar{A} \leq 21.42 \cdot \bar{A}^*$
$W \geq 2$	Eywa	$\bar{A} \leq (1 + p_{\max}) \log_2 e \cdot \bar{A}^*$	$\bar{A} \leq 1.62 \cdot \bar{A}^*$
	Stat. Rand. [28], [29] (Ch. 4)	$\bar{A} \leq 2 \cdot \bar{A}^*$	$\bar{A} \leq 2 \cdot \bar{A}^*$

2) Eywa: Step 1—Transform Objective Function and Constraints: Since the original objective function is $J_o(\cdot) = W$, and that $W = \sum_{i=1}^{N} r_i$, we can just replace it by $J_R(r_1, r_2, \cdots, r_N) = \sum_{i=1}^{N} r_i$.

In addition, for constraint (17), which is AoI-based and specific for the Min-BW problem, we need to transform it to a rate-based constraint. For an AUS with rate r_i's, the transmission intervals T_i^k's for source i are upper bounded by $\lceil \frac{1}{r_i} \rceil$, where $\lceil \cdot \rceil$ is the ceiling function. Therefore, if $r_i \geq \frac{1}{d_i}$, we will have $T_i^k \leq d_i$ for all $k \geq 1$. That is, $r_i \geq \frac{1}{d_i}$ is a sufficient condition for constraint (17). As such, we can replace constraint (17) (AoI-based) by the following rate-based constraint:

$$r_i \geq \frac{1}{d_i}. \tag{18}$$

3) Eywa: Step 2—Find Optimal Transmission Rates: The second step of Eywa is to find the optimal rate vector $\boldsymbol{r}^*$, i.e., to solve the following optimization problem OPT-SD (Min-BW).

$$\begin{aligned}
&\text{OPT-SD (Min-BW)} \\
&\min_{\boldsymbol{r}, W} \quad \sum_{i=1}^{N} r_i \\
&\text{s.t.} \quad \boldsymbol{\gamma} = \text{sort}(\boldsymbol{r}), \\
&\qquad \boldsymbol{\gamma} \text{ is step-down}, \\
&\qquad \sum_{i=1}^{N} r_i = W, \\
&\qquad r_i \geq \frac{1}{d_i}, \quad i = 1, 2, \cdots, N, \\
&\qquad 0 < r_i \leq 1, \quad i = 1, 2, \cdots, N. \\
&\qquad W \in \mathbb{N}^*.
\end{aligned}$$

Note that in OPT-SD (Min-BW), $r_i \geq \frac{1}{d_i}$ is the additional problem-specific rate-based constraint. Denote $\boldsymbol{r}^*$ as the optimal solution to OPT-SD (Min-BW), which we will use to construct an AUS-based scheduler in Step 3. The complete algorithm to solve OPT-SD (Min-BW) is given in Appendix C.

4) Eywa: Step 3—Construct an AUS-Based Scheduler: The third step of Eywa is to construct an AUS-based scheduler based on $\boldsymbol{r}^*$, which is exactly the same as what we have done in Section II-C.

5) Performance and Comparison with State-of-the-Art: In this section, we show that Eywa offers a strong theoretical performance guarantee to the Min-BW problem under peak AoI constraints. Denote W^* as the optimal (unknown) objective value for the Min-BW problem under peak AoI constraints. We have the following theorem that guarantees the performance of W—-the objective value obtained by Eywa.

Theorem 2 *For the Min-BW problem under peak AoI constraints, the objective value obtained by Eywa satisfies:*

$$W \leq \lceil \log_2 e \cdot W^* \rceil. \tag{19}$$

A proof of Theorem 2 is given in Appendix D.

For the Min-BW problem under peak AoI constraints, a state-of-the-art solution (called Aion) was proposed in [26]. Aion can offer a performance guarantee of $W \leq 2 \cdot W^*$ (a proof is given in Appendix E). Since $\log_2 e \approx 1.44$, we can see that Eywa offers a tighter performance bound than Aion.

B. Min-BW under Average AoI Constraints

1) Problem Statement: For this problem, we assume there is an average AoI constraint (denoted as $\alpha_i \in \mathbb{R}^*$) for the average AoI $\bar{A}_i$. That is, for $i = 1, 2, \cdots, N$,

$$\bar{A}_i \leq \alpha_i. \tag{20}$$

For average AoI constraints, the packet loss rate p_i for each source i can be greater than 0, i.e., $p_i \geq 0$. Recall in our system model, we assume $r_i \leq 1$, i.e., there is *at most* one packet transmitted from source i in every time slot, regardless of how large W is. So to satisfy constraint (20), there must be a limit on the value p_i (channel condition). It can be shown that $\bar{A}_i \geq \frac{1}{1-p_i}$ when $r_i \leq 1$. So for constraint (20) to hold, we must have $\alpha_i \geq \frac{1}{1-p_i}$, i.e.,

$$p_i \leq \frac{1}{\alpha_i} - 1, \tag{21}$$

for each $i = 1, 2, \cdots, N$.

The objective of the Min-BW problem under average AoI constraints is to find a scheduler π that minimizes bandwidth W such that constraint (20) is satisfied.

2) Eywa: Step 1—Transform Objective Function and Constraints: Again, in the first step of Eywa, we transform the original objective function $J_o(\cdot) = W$ to $J_R(r_1, r_2, \cdots, r_N) = \sum_{i=1}^{N} r_i$.

For the AoI-based constraint (20), we need to transform it to a rate-based constraint. Recall in Section III-C we have $\bar{A}_i \leq q_i(r_i)$. Then, to satisfy (20), its sufficient to have $q_i(r_i) \leq \alpha_i$ It can be shown that $q_i(r_i) \leq \alpha_i$ will hold if

$$r_i \geq \frac{2\lfloor x \rfloor + 1 - x}{\lfloor x \rfloor^2 + \lfloor x \rfloor}, \tag{22}$$

where $x = \frac{(1-p_i)(2\alpha_i - 1)}{1+p_i}$. So we can replace the original AoI-based constraint (20) by the new rate-based constraint (22).

3) Eywa: Step 2—Find Optimal Transmission Rates: The second step of Eywa is to find the optimal rate vector $\boldsymbol{r}^*$, i.e., to solve OPT-SD (Min-BW) by replacing problem-specific constraint "$r_i \geq \frac{1}{d_i}$" with constraint (22). The solution to this problem is identical to that shown in Appendix C (except the minor change in the RHS value of the problem-specific constraint).

4) Eywa: Step 3—Construct an AUS-based scheduler: The third step of Eywa is to construct an AUS-based scheduler based on $\boldsymbol{r}^*$, which is exactly the same as what we have done in Section II-C.

5) Performance Guarantee: In this section, we show that Eywa offers a strong theoretical performance guarantee to the Min-BW problem under average AoI constraints. Denote W^* as the optimal (unknown) objective value for the Min-BW problem under average AoI constraints. We have the following theorem that guarantees the performance of W—-the objective value obtained by Eywa.

Theorem 3 *For the Min-BW problem under average AoI constraints, the objective value obtained by Eywa satisfies:*

$$W \leq \lceil \frac{9 \log_2 e \cdot (1 + p_{\max})}{8} \cdot W^* \rceil. \tag{23}$$

A proof of Theorem 3 is given in Appendix F.

To date, a solution to the Min-BW problem under average AoI constraints is not available in the literature. So the solution that we presented here is all new.

V. Application to Decision Problems

In the previous two sections, we applied Eywa to solve two optimization problems: Min-Sum and Min-BW. In addition to solving optimization problems, the Eywa framework can also be applied to solve decision problems related to AoI, which we will show in this section.

The decision problem we study is to determine the existence of feasible schedulers to satisfy AoI constraints. Specifically, we study the problems under two AoI constraints: i) *hard* AoI deadlines, and ii) *soft* AoI deadlines. The first problem was studied in [24], and the second problem was studied in [25], both for the special case of $W = 1$. We show that Eywa can be applied to solve both problems for the general case of $W \geq 1$.

A. A Decision Problem with Hard AoI Deadlines

1) Problem Statement: We consider the system model in Section II-A with fixed bandwidth $W \geq 1$. Denote $\boldsymbol{d} = [d_1, d_2, \cdots, d_N]$ as the deadline vector. Similar to Section IV-A, we assume a hard constraint (17) for each $i = 1, 2, \cdots, N$. Since constraint (17) is a deterministic one, it can hold only when $p_i = 0$ for $i = 1, 2, \cdots, N$.

Our goal is to solve the following decision problem: For a given $\boldsymbol{d}$, determine whether or not there exists a feasible scheduler that satisfies (hard) deadline constraint (17). If there exists a feasible scheduler, then we want to find one.

2) Eywa: Step 1—Transform Objective Function and Constraints: For the decision problem, there is no objective function and we don't need to make any transform.

For the constraints, just like in Section IV-A, we replace the AoI-based constraint (17) by the rate-based constraint $r_i \geq \frac{1}{d_i}$.

3) Eywa: Step 2—Find Optimal Transmission Rates: The second step of Eywa is to find the optimal rate vector $\boldsymbol{r}$. For a decision problem (no objective function), we need to solve the following problem (DEC-SD), along with the problem-specific constraint $r_i \geq \frac{1}{d_i}$.

$$\begin{aligned}
\text{DEC-SD:}\quad & \text{Determine whether or not } \boldsymbol{r} \text{ exists.} \\
\text{s.t.}\quad & \boldsymbol{\gamma} = \text{sort}(\boldsymbol{r}), \\
& \boldsymbol{\gamma} \text{ is step-down}, \\
& \sum_{i=1}^{N} r_i = W, \\
& r_i \geq \frac{1}{d_i}, \quad i = 1, 2, \cdots, N, \\
& 0 < r_i \leq 1, \quad i = 1, 2, \cdots, N.
\end{aligned}$$

The algorithm to solve DEC-SD is given in Appendix G. There are two possible outcomes. If we can find a feasible $\boldsymbol{r}$ to DEC-SD, then we will use it to construct an AUS-based scheduler in Step 3. Otherwise, we conclude we cannot find a feasible scheduler for $\boldsymbol{d}$.

4) Eywa: Step 3—Construct an AUS-based scheduler: The third step of Eywa is to construct an AUS-based scheduler based on $\boldsymbol{r}$, which is exactly the same as what we have done in Section II-C.

5) Eywa: Performance and Comparison with State-of-the-Art: The following theorem gives a performance guarantee of Eywa:

Theorem 4 *For any* $\boldsymbol{d}$ *with* $\sum_{i=1}^{N} \frac{1}{d_i} \leq \frac{W}{\log_2 e}$, *Eywa can always find a feasible scheduler.*

Theorem 4 can be easily proved based on Lemma 1 in Appendix B and the design of Algorithm 6 in Appendix G.

For the decision problem with hard AoI deadlines, a state-of-the-art solution (called FPM) was proposed in [24]. But FPM was only designed for the special case of $W = 1$ and it is not clear how to extend it to the general case of $w \geq 1$. The main result of FPM is that for any $\boldsymbol{d}$ with $\sum_{i=1}^{N} \frac{1}{d_i} \leq \frac{1}{\log_2 e}$, FPM can always find a feasible scheduler [24], which is consistent to what Eywa can also deliver (when $W = 1$).

B. A Decision Problem with Soft AoI Deadlines

1) Problem Statement: Now we assume the AoI deadline d_i's are soft, i.e., occasional violations can be tolerated. Denote $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \cdots, \epsilon_N]$ as the vector for violation rates, where $\epsilon_i \in [0, 1)$ for each $i = 1, 2, \cdots, N$. The soft AoI deadline constraint for each source node $i = 1, 2, \cdots, N$ is given by

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} [A_i(t) > d_i] \le \epsilon_i, \tag{24}$$

where "$[\cdot]$" is Iverson bracket, returning 1 if the inside statement is true and 0 otherwise. Denote $\boldsymbol{p} = [p_1, p_2, \cdots, p_N]$ as the vector for packet loss rates. For soft AoI deadlines, p_i for each source i can be greater than 0, i.e., $p_i \ge 0$.

Similar to that in Section V-A, our goal is to solve the following decision problem: For a given $(\boldsymbol{d}, \boldsymbol{\epsilon}, \boldsymbol{p})$, determine whether or not there exists a feasible scheduler that satisfies (soft) deadline constraint (24). If there exists a feasible scheduler, then we want to find one.

2) Eywa: Step 1—Transform Objective Function and Constraints: Again, for the decision problem, there is no objective function and we don't need to make any transform.

For the AoI-based constraint (24), we need to transform it to a rate-based constraint. In our previous work [25], we showed that (24) is satisfied if

$$r_i \ge r_i^{\text{tar}}, \tag{25}$$

where r_i^{tar} is given by

$$r_i^{\text{tar}} = \begin{cases} \dfrac{1}{\left\lfloor \frac{(1-p_i)d_i}{(1-p_i)\lfloor \log_{p_i} \epsilon_i \rfloor + 1 - \epsilon_i p_i^{-\lfloor \log_{p_i} \epsilon_i \rfloor}} \right\rfloor}, & \text{if } \epsilon_i < p_i, \\ \frac{1-\epsilon_i}{(1-p_i)d_i}, & \text{if } \epsilon_i \ge p_i. \end{cases}$$

So we can replace the original AoI-based constraint (24) by the new rate-based constraint (25).

3) Eywa: Step 2—Find Optimal Transmission Rates: The second step of Eywa is to find the optimal rate vector $\boldsymbol{r}$. Similar to that in Section V-A, we need to solve DEC-SD by replacing problem-specific constraint "$r_i \ge \frac{1}{d_i}$" with constraint (25). The solution to this problem is identical to that shown in Appendix G. Again, if we can find a feasible $\boldsymbol{r}$ to DEC-SD, then we will use it to construct an AUS-based scheduler in Step 3. Otherwise, we conclude we cannot find a feasible scheduler for $\boldsymbol{d}$.

4) Eywa: Step 3—Construct an AUS-based scheduler: The third step of Eywa is to construct an AUS-based scheduler based on $\boldsymbol{r}$, which is exactly the same as what we have done in Section II-C.

5) Eywa: Performance and Comparison with State-of-the-Art: The following theorem gives a performance guarantee of Eywa:

Theorem 5 *For any $(\boldsymbol{d}, \boldsymbol{\epsilon}, \boldsymbol{p})$ with $\sum_{i=1}^{N} r_i^{tar} \le \frac{W}{\log_2 e}$, Eywa can always find a feasible scheduler.*

Theorem 5 can be easily proved based on Lemma 1 in Appendix B.

For the decision problem with soft AoI deadlines, a state-of-the-art solution (called UTS) was proposed in [25]. Again, UTS was only designed for the special case of $W = 1$. The main result of UTS is that for any $(\boldsymbol{d}, \boldsymbol{\epsilon}, \boldsymbol{p})$ with $\sum_{i=1}^{N} r_i^{\text{tar}} \le \frac{1}{\log_2 e}$, UTS can always find a feasible scheduler [25], which is consistent to what Eywa can also deliver (when $W = 1$).

VI. Summary

This paper presented Eywa—a general design framework that can be applied to construct high-performance schedulers for AoI-related optimization and decision problems. The core of Eywa hinges upon the notions of AUS schedulers and step-down rate vectors. The framework of Eywa consists of three steps: (i) transform the (AoI-based) objective function and constraints to rate-based ones, (ii) find the optimal step-down rate vector by solving an optimization problem (OPT-SD), and (iii) construct an AUS-based scheduler using the optimal step-down rate vector. To validate the efficacy of the proposed Eywa framework, we applied it to solve a number of AoI optimization and decision problems, such as Min-Sum, Min-BW, and the decision problems with AoI constraints. The results are summarized in Fig. 3. We found that for each problem, Eywa can either offer a stronger performance guarantee than the state-of-the-art algorithms, or provide new/general results that are not available in the literature.

References

[1] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing Age of Information in Vehicular Networks," in *Proc. IEEE SECON,* pp. 350–358, Salt Lake City, UT, USA, June 27–30, 2011.

[2] S. Kaul, R. Yates, and M. Gruteser, "Real-Time Status: How Often Should One Update?" in *Proc. IEEE INFOCOM,* pp. 2731–2735, Orlando, FL, USA, Mar. 25–30, 2012.

[3] R.D. Yates, P. Ciblat, A. Yener, and M. Wigger, "Age-Optimal Constrained Cache Updating," in *Proc. IEEE ISIT,* pp. 141–145, Aachen, Germany, June 25–30, 2017.

[4] A.M. Bedewy, Y. Sun, and N.B. Shroff, "Optimizing Data Freshness, Throughput, and Delay in Multi-Server Information-Update Systems," in *Proc. IEEE ISIT,* pp. 2569–2573, Barcelona, Spain, July 10–15, 2016.

[5] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Minimizing the Age of Information in Broadcast Wireless Networks," in *Proc. Allerton Conference,* pp. 844–851, Monticello, IL, USA, Sept. 27–30, 2016.

[6] I. Kadota, A. Sinha, and E. Modiano, "Optimizing Age of Information in Wireless Networks with Throughput Constraints," in *Proc. of IEEE INFOCOM,* pp. 1844–1852, Honolulu, HI, USA, Apr. 16–18, 2018

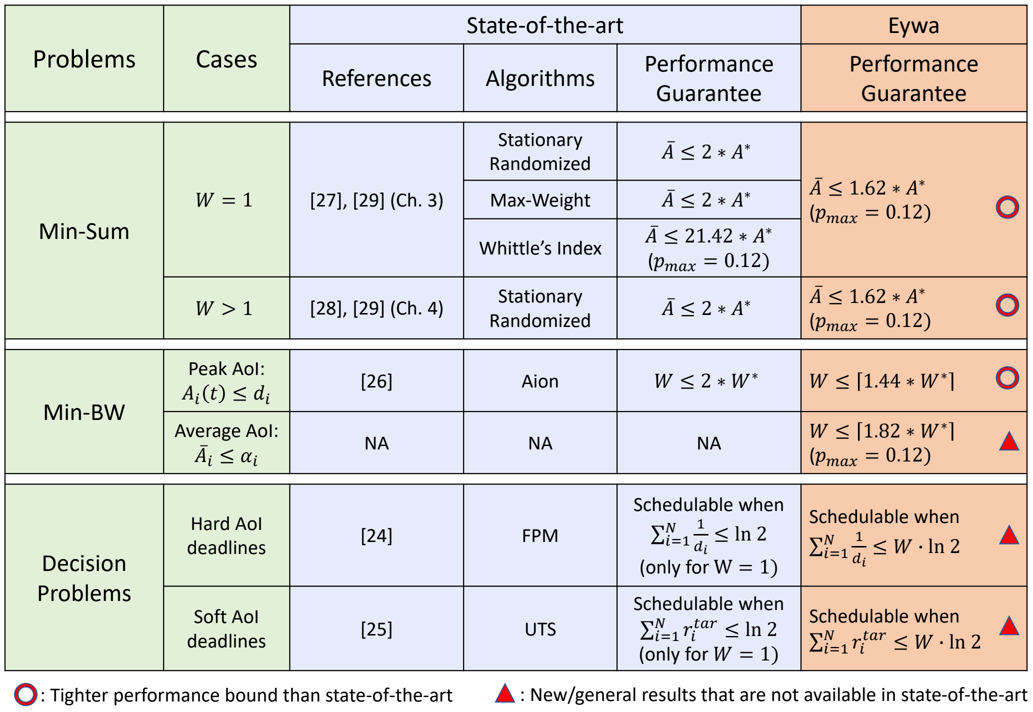
[7] C. Li, S. Li, Y. Chen, Y.T. Hou, and W. Lou, "Minimizing Age of Information under General Models for IoT Data Collection," *IEEE Trans. on Network Science and Engineering*, vol. 7, no. 4, pp. 2256–2270, Oct. 2020.

[8] Y. Sun, E. Uysal-Biyikoglu, and S. Kompella, "Age-Optimal Updates of Multiple Information Flows," in *Proc. IEEE INFOCOM Workshops – Age of Information Workshop,* pp. 136–141, Honolulu, HI, USA, April 15–19, 2018.

[9] Z. Qian, F. Wu, J. Pan, K. Srinivasan, and N.B. Shroff, "Minimizing Age of Information in Multi-channel Time-sensitive Information Update Systems," in *Proc. IEEE INFOCOM,* pp. 446–455, online conference, July 6–9, 2020.

[10] B. Sombabu and S. Moharir, "Age-of-Information Based Scheduling for Multi-Channel Systems," *IEEE Trans. on Wireless Communication,* vol. 19, no. 7, pp. 4439–4448, July 2020.

[11] C. Li, Y. Huang, Y. Chen, B. Jalaian, Y.T. Hou, and W. Lou, "Kronos: A 5G Scheduler for AoI Minimization under Dynamic Channel Conditions," in *Proc. IEEE ICDCS,* pp. 1466–1472, Dallas, TX, USA, July 7–9, 2019.



Problems	Cases	State-of-the-art			Eywa
		References	Algorithms	Performance Guarantee	Performance Guarantee
Min-Sum	$W = 1$	[27], [29] (Ch. 3)	Stationary Randomized	$\bar{A} \leq 2 * A^*$	$\bar{A} \leq 1.62 * A^*$ ($p_{max} = 0.12$) ○
			Max-Weight	$\bar{A} \leq 2 * A^*$	
			Whittle's Index	$\bar{A} \leq 21.42 * A^*$ ($p_{max} = 0.12$)	
	$W > 1$	[28], [29] (Ch. 4)	Stationary Randomized	$\bar{A} \leq 2 * A^*$	$\bar{A} \leq 1.62 * A^*$ ($p_{max} = 0.12$) ○
Min-BW	Peak AoI: $A_i(t) \leq d_i$	[26]	Aion	$W \leq 2 * W^*$	$W \leq \lceil 1.44 * W^* \rceil$ ○
	Average AoI: $\bar{A}_i \leq \alpha_i$	NA	NA	NA	$W \leq \lceil 1.82 * W^* \rceil$ ($p_{max} = 0.12$) ▲
Decision Problems	Hard AoI deadlines	[24]	FPM	Schedulable when $\sum_{i=1}^{N} \frac{1}{d_i} \leq \ln 2$ (only for W = 1)	Schedulable when $\sum_{i=1}^{N} \frac{1}{d_i} \leq W \cdot \ln 2$ ▲
	Soft AoI deadlines	[25]	UTS	Schedulable when $\sum_{i=1}^{N} r_i^{tar} \leq \ln 2$ (only for $W = 1$)	Schedulable when $\sum_{i=1}^{N} r_i^{tar} \leq W \cdot \ln 2$ ▲

○: Tighter performance bound than state-of-the-art ▲: New/general results that are not available in state-of-the-art

Figure 3: Applications of Eywa to a family of AoI problems and a comparison of performance guarantees. NA indicates results are not available in the literature.

[12] H. Tang, J. Wang, L. Song, and J. Song, "Minimizing Age of Information With Power Constraints: Multi-User Opportunistic Scheduling in Multi-State Time-Varying Channels," *IEEE Journal on Selected Areas in Communications,* vol. 38, no. 5, pp. 854-–868, May 2020.

[13] T. Park, W. Saad, and B. Zhou, "Centralized and Distributed Age of Information Minimization with Non-linear Aging Functions in the Internet of Things," *IEEE Internet of Things Journal,* vol. 8, no. 10, pp. 8437-–8455, May 2021.

[14] Y. Hsu, E. Modiano, and L. Duan, "Age of Information: Design and Analysis of Optimal Scheduling Algorithms," in *Proc. IEEE ISIT,* pp. 561–565, Archen, Germany, June 25–30, 2017.

[15] A.M. Bedewy, Y. Sun, and N.B. Shroff, "Age-Optimal Information Updates in Multihop Networks," in *Proc. IEEE ISIT,* pp. 576–580, Archen, Germany, June 25–30, 2017.

[16] J. Lou, X. Yuan, S. Kompella, and N. Tzeng, "AoI and Throughput Tradeoffs in Routing-aware Multi-hop Wireless Networks," in *Proc. IEEE INFOCOM,* pp. 476–485, online conference, July 6–9, 2020.

[17] C. Xu, Q. Xu, J. Wang, K. Wu, K. Lu, and C. Qiao, "AoI-centric Task Scheduling for Autonomous Driving Systems," in *Proc. IEEE INFOCOM,* pp. 1019–1028, online conference, May 2–5, 2022.

[18] N. Lu, B. Ji, and B. Li, "Age-based Scheduling: Improving Data Freshness for Wireless Real-Time Traffic," in *Proc. ACM MobiHoc,* pp. 191–200, Los Angeles, CA, USA, June 26–29, 2018.

[19] J. Pan, A.M. Bedewy, Y. Sun, and N.B. Shroff, "Minimizing Age of Information via Scheduling over Heterogeneous Channels," in *Proc. ACM MobiHoc,* pp. 111–120, Shanghai, China, July 26—29, 2021.

[20] J. Zhong, R.D. Yates, and E. Soljanin, "Two Freshness Metrics for Local Cache Refresh," in *Proc. IEEE ISIT,* pp. 1924–1928, Vail, CO, USA, June 17–22, 2018.

[21] A. Maatouk, S. Kriouile, M. Assaad, and A. Ephremides, "The Age of Incorrect Information: A New Performance Metric for Status Updates," *IEEE/ACM Trans. on Networking,* vol. 28, no. 5, pp. 2215-–2228, Oct. 2020.

[22] Q. Liu, C. Li, Y.T. Hou, W. Lou, J.H. Reed, and S. Kompella, "Ao^{2}I: Minimizing Age of Outdated Information to Improve Freshness in Data Collection," in *Proc. IEEE INFOCOM,* pp. 1359–1368, online conference, May 2–5, 2022.

[23] D. Guo, K. Nakhleh, I. Hou, S. Kompella, and C. Kam, "A Theory of Second-Order Wireless Network Optimization and Its Application on AoI," in *Proc. IEEE INFOCOM,* pp. 999–1008, online conference, May 2–5, 2022.

[24] C. Li, S. Li, Y. Chen, Y.T. Hou, and W. Lou, "AoI Scheduling with Maximum Thresholds," in *Proc. IEEE INFOCOM,* pp. 436–445, online conference, July 6–9, 2020.

[25] C. Li, Q. Liu, S. Li, Y. Chen, Y.T. Hou, and W. Lou, "On Scheduling with AoI Violation Tolerance," in *Proc. IEEE INFOCOM,* online conference, May 10–13, 2021.

[26] Q. Liu, C. Li, Y.T. Hou, W. Lou, and Sastry Kompella, "Aion: A Bandwidth Optimized Scheduler with AoI Guarantee," in *Proc. IEEE INFOCOM,* online conference, May 10–13, 2021.

[27] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Scheduling Policies for Minimizing Age of Information in Broadcast Wireless Networks," *IEEE/ACM Trans. on Networking,* vol. 26, no. 6, pp. 2637-–26250, Dec. 2018.

[28] R. Talak, S. Karaman, E. Modiano, "Optimizing Information Freshness in Wireless Networks under General Interference Constraints," in *Proc. ACM MobiHoc,* pp. 61–70, Los Angeles, USA, June 25—28, 2018.

[29] Y. Sun, I. Kadota, R. Talak, and E. Modiano, "Age of information: A new metric for information freshness," *Synthesis Lectures on Communication Networks, Morgan & Claypool Publishers,* 2019.

[30] Z. Zhang, Z. Duan, and Y.T. Hou, "Virtual Time Reference System: A Unifying Scheduling Framework for Scalable Support of Guaranteed Services," *IEEE Journal on Selected Areas in Communications,* vol. 18, no. 12, pp. 2684-–2695, Dec. 2000.

[31] C. Jiang, Y. Shi, Y.T. Hou, W. Lou, S. Kompella, and S.F. Midkiff, "Toward Simple Criteria to Establish Capacity Scaling Laws for Wireless Networks," in *Proc. IEEE INFOCOM,* pp. 774–782, Orlando, FL, USA, Mar. 25–30, 2012.

[32] C. Jiang, Y. Shi, Y.T. Hou, W. Lou, S. Kompella, and S.F. Midkiff, "A General Method to Determine Asymptotic Capacity Upper Bounds for Wireless Networks," *IEEE Trans. on Network Science and Engineering*, vol. 6, no. 1, pp. 2–15, Nov. 2017.

[33] A. Durán, M. Toril, F. Ruiz, and A. Mendo, "Self-Optimization Algorithm for Outer Loop Link Adaptation in LTE," *IEEE Communications*

Letters, vol. 9, issue 11, Nov. 2015.

Appendix

A. An Algorithm to Solve OPT-β

We first transform the decision variable $\boldsymbol{r}$ to its sorted version γ, to make the problem easier. We define $\boldsymbol{\gamma}^{\text{LB}} = \text{sort}(\boldsymbol{r}^{\text{LB}})$. It can be shown that in the optimal solution $\boldsymbol{r}^*$ to OPT-β, the order of elements should be the same as the order of elements. That is, for any $i \neq j$, if $r_i^{\text{LB}} > r_j^{\text{LB}}$, then we have $r_i^* > r_j^*$. Therefore, it's sufficient to solve the following OPT-β ($\boldsymbol{\gamma}$) with variable $\boldsymbol{\gamma}$ (rather than $\boldsymbol{r}$):

$$\begin{aligned}\text{OPT-}\beta\ (\boldsymbol{\gamma}):\ \max_{\boldsymbol{\gamma},\beta}\quad & \beta \\ \text{s.t.}\quad & \boldsymbol{\gamma} \text{ is step-down}, \\ & \beta\cdot\gamma_i^{\text{LB}} \le \gamma_i \le 1,\ i=1,2,\cdots,N, \\ & \sum_{i=1}^N \gamma_i = W.\end{aligned}$$

After obtaining $\boldsymbol{\gamma}^*$ to OPT-β ($\boldsymbol{\gamma}$), we can re-sort it and obtain $\boldsymbol{r}^*$, the optimal solution to OPT-β.

We notice that there are two variables $\boldsymbol{\gamma}$ and β in OPT-β ($\boldsymbol{\gamma}$), which are hard for us to handle at the same time. So we propose to fix β first, and transform OPT-β ($\boldsymbol{\gamma}$) to a decision problem DEC-$\boldsymbol{\gamma}$, for whether or not there exists a feasible $\boldsymbol{\gamma}$ to OPT-β ($\boldsymbol{\gamma}$).

$$\begin{aligned}\text{DEC-}\boldsymbol{\gamma}:\ & \text{Determine whether or not } \boldsymbol{\gamma} \text{ exists.} \\ \text{s.t.}\quad & \boldsymbol{\gamma} \text{ is step-down}, \\ & \beta\cdot\gamma_i^{\text{LB}} \le \gamma_i \le 1,\ i=1,2,\cdots,N, \\ & \sum_{i=1}^N \gamma_i = W.\end{aligned}$$

If we have an algorithm to solve DEC-$\boldsymbol{\gamma}$ under fixed β, we could use bisection to find the optimal β^* to OPT-β ($\boldsymbol{\gamma}$).

To solve DEC-$\boldsymbol{\gamma}$, we further transform it to an easier optimization problem OPT-$\hat{\boldsymbol{\gamma}}$. we define $l_i = \beta\cdot\gamma_i^{\text{LB}}$ for each $i = 1,2,\cdots,N$. Clearly, we have $l_1 \ge l_2 \ge \cdots \ge l_N$. Then for fixed β, we consider the following optimization problem with objective $\hat{\boldsymbol{\gamma}}$.

$$\begin{aligned}\text{OPT-}\hat{\boldsymbol{\gamma}}:\ \min_{\hat{\boldsymbol{\gamma}}}\quad & \sum_{i=1}^N \hat{\gamma}_i \\ \text{s.t.}\quad & \hat{\boldsymbol{\gamma}} \text{ is step-down}, \\ & l_i \le \hat{\gamma}_i \le 1,\ i=1,2,\cdots,N.\end{aligned}$$

Denote $\hat{\boldsymbol{\gamma}}^*$ as the optimal objective to OPT-$\hat{\boldsymbol{\gamma}}$. If $\sum_{i=1}^N \hat{\gamma}_i^* \le W$, then $\boldsymbol{\gamma} = \hat{\boldsymbol{\gamma}}^*\cdot W/(\sum_{i=1}^N \hat{\gamma}_i^*)$ is a feasible solution to DEC-$\boldsymbol{\gamma}$. On the other hand, if $\sum_{i=1}^N \hat{\gamma}_i^* \le W$, then there does not exist any feasible solution to DEC-$\boldsymbol{\gamma}$ for the fixed β.

OPT-$\hat{\boldsymbol{\gamma}}$ can be solved by a DP-based solution (see Algorithm 2 in our previous work [25]). We only need the following minor change. In the step-down definition (Definition 2) in this paper, we allow the leading elements in γ to be 1. As a result, we need to set $\mathcal{R}_i = \{1\} \cup \{n\hat{r}_k \mid r_i^{\text{lb}} \le n\hat{r}_k \le 1, n \in \mathbb{N}^*\}$ in Step 5 of Algorithm 2 in [25].

Algorithm 4 A Bisection-Based Solution to OPT-β

```
Input: $\boldsymbol{r}^{\text{LB}}$, $W$, $\epsilon$.
Output: $\beta^*$ and $\boldsymbol{r}^*$.
1: Sort $\boldsymbol{r}^{\text{LB}}$ and get $\boldsymbol{\gamma}^{\text{LB}} = \text{sort}(\boldsymbol{r}^{\text{LB}})$. Set $\beta_{\text{LB}} = 0$, $\beta_{\text{UB}} = 1$.
2: while $\beta_{\text{UB}} - \beta_{\text{LB}} > \epsilon$ do
3:     Set $\beta = (\beta_{\text{UB}} + \beta_{\text{LB}})/2$.
4:     Set $l_i = \beta\cdot\gamma_i^{\text{LB}}$ for each $i = 1,2,\cdots,N$. Solve OPT-$\hat{\boldsymbol{\gamma}}$
       (by Algorithm 2 in [25]) and obtain an optimal solution $\hat{\boldsymbol{\gamma}}^*$.
5:     if $\sum_{i=1}^N \hat{\gamma}_i^* \le W$ then
6:         Set $\boldsymbol{\gamma}^* = \hat{\boldsymbol{\gamma}}^*\cdot W/(\sum_{i=1}^N \hat{\gamma}_i^*)$. Set $\beta^* = \beta$. Set $\beta_{\text{LB}} = \beta$.
7:     else
8:         Set $\beta_{\text{UB}} = \beta$.
9:     end if
10: end while
11: Re-sort $\boldsymbol{\gamma}^*$ and get $\boldsymbol{r}^*$.
```

A pseudocode for a bisection-based solution to solve OPT-β is given in Algorithm 4. Note that ϵ is an input parameter which stands for error tolerance of the objective β, with $\epsilon << 1$.

B. A Proof of Theorem 1

Before proving Theorem 1, we first introduce the following lemma w.r.t. $\hat{\boldsymbol{\gamma}}^*$,the optimal objective to OPT-$\hat{\boldsymbol{\gamma}}$.

Lemma 1 $\sum_{i=1}^N \hat{\gamma}_i^* \le \log_2 e\cdot\sum_{i=1}^N l_i$.

Proof We proof Lemma 1 by contradiction. Suppose $\sum_{i=1}^N \hat{\gamma}_i^* > \log_2 e\cdot\sum_{i=1}^N l_i$. Then for any $\hat{\boldsymbol{\gamma}}$ that is feasible to OPT-$\hat{\boldsymbol{\gamma}}$, we have $\sum_{i=1}^N \hat{\gamma}_i > \log_2 e\cdot\sum_{i=1}^N l_i$.

For any $x > 0$, we define two vectors $\hat{\boldsymbol{\gamma}}^x = [\hat{\gamma}_1^x, \hat{\gamma}_2^x, \cdots, \hat{\gamma}_N^x]$ and $\boldsymbol{\gamma}^x = [\gamma_1^x, \gamma_2^x, \cdots, \gamma_N^x]$ as:

$$\hat{\gamma}_i^x = \min\{1, x\cdot 2^{\lceil\log_2(\frac{l_i}{x})\rceil}\},\ i=1,2,\cdots,N. \tag{26}$$

$$\gamma_i^x = x\cdot 2^{\lceil\log_2(\frac{l_i}{x})\rceil},\ i=1,2,\cdots,N. \tag{27}$$

It can be verified that $\hat{\boldsymbol{\gamma}}^x$ is feasible to OPT-$\hat{\boldsymbol{\gamma}}$. Therefore, for any $x > 0$ we have

$$\sum_{i=1}^N \hat{\gamma}_i^x > \log_2 e\cdot\sum_{i=1}^N l_i. \tag{28}$$

For each i, we have $\gamma_i^x \ge \hat{\gamma}_i^x$. So for any $x > 0$, we have

$$\sum_{i=1}^N \gamma_i^x > \log_2 e\cdot\sum_{i=1}^N l_i. \tag{29}$$

Define $g(x) = \frac{1}{x}$. We have

$$\int_{0.5}^1 g(x)dx = \frac{1}{\log_2 e}. \tag{30}$$

Multiplying the two sides of (29) and (30) respectively, we have

$$\int_{0.5}^{1} g(x)dx \cdot \sum_{i=1}^{N} \gamma_i^x > \sum_{i=1}^{N} l_i. \tag{31}$$

Substituting $g(x) = \frac{1}{x}$ and (27) into (31), we have

$$\sum_{i=1}^{N} \int_{0.5}^{1} 2^{\lceil \log_2(\frac{l_i}{x}) \rceil} dx > \sum_{i=1}^{N} l_i. \tag{32}$$

It can be shown that the LHS of (32) equals to $\sum_{i=1}^{N} l_i$. So we have

$$\sum_{i=1}^{N} l_i > \sum_{i=1}^{N} l_i. \tag{33}$$

Clearly, (33) can not hold (i.e., a contradiction). This completes our proof of Lemma 1. ∎

With Lemma 1 in hands, we can further prove Lemma 2 w.r.t. β^*, the optimal solution to OPT-β. Note that Lemma 2 holds when ϵ is sufficiently small (otherwise, Algorithm 4 will terminate too early).

Lemma 2 *When ϵ is sufficiently small, we can always have $\beta^* \geq \frac{1}{\log_2 e}$.*

Proof We prove Lemma 2 by contradiction. Suppose $\beta^* < \frac{1}{\log_2 e}$. Then for $\hat{\boldsymbol{\gamma}}^*$, the optimal objective to OPT-$\hat{\boldsymbol{\gamma}}$, we have

$$\sum_{i=1}^{N} \hat{\gamma}_i^* \geq W, \tag{34}$$

otherwise β^* is not the optimal objective to OPT-β. On the other hands, by Lemma 1, we have

$$\sum_{i=1}^{N} \hat{\gamma}_i^* \leq \log_2 e \cdot \sum_{i=1}^{N} \beta^* \cdot \gamma_i^{\text{LB}}. \tag{35}$$

Since $\boldsymbol{\gamma}$ is feasible to OPT-LB, we have $\sum_{i=1}^{N} \gamma_i^{\text{LB}} \leq W$. Considering (35) and $\beta^* < \frac{1}{\log_2 e}$, we have

$$\sum_{i=1}^{N} \hat{\gamma}_i^* < W, \tag{36}$$

which contradicts to (34). This completes our proof of Lemma 2. ∎

With Lemma 2 in hands, now we can offer a proof of Theorem 1. Note that in Algorithm 4, we must set ϵ small enough to make sure $\beta^* \geq \frac{1}{\log_2 e}$.

Proof Considering $\bar{A}_{\text{LB}} \leq \bar{A}^*$, to prove (16), it's sufficient to prove

$$\sum_{i=1}^{N} w_i \bar{A}_i \leq (1 + p_{\max}) \log_2 e \cdot \bar{A}_{\text{LB}}. \tag{37}$$

Define $f(x) = 2\lfloor \frac{1}{x} \rfloor + 1 - x(\lfloor \frac{1}{x} \rfloor^2 + \lfloor \frac{1}{x} \rfloor)$. Then we have $q_i(r_i) = \frac{1+p_i}{1-p_i} f(r_i) + \frac{1}{2}$. Following the discussions in Section III-C, it can be shown that $q_i(r_i) \geq \bar{A}_i$ for each i. Therefore, for the AUS with $\boldsymbol{r}^*$, to prove (37), it's sufficient to prove

$$\sum_{i=1}^{N} \frac{w_i(1+p_i)}{2(1-p_i)} f(r_i) + \sum_{i=1}^{N} \frac{w_i}{2} \leq (1+p_{\max}) \log_2 e \cdot \bar{A}_{\text{LB}}. \tag{38}$$

Considering (13), to prove (38), it's sufficient to prove

$$\frac{w_i(1+p_i)}{2(1-p_i)} f(r_i^*) + \frac{w_i}{2} \leq (1+p_{\max}) \log_2 e \cdot \left(\frac{w_i}{2(1-p_i) r_i^{\text{LB}}} + \frac{w_i}{2}\right).$$

for each $i = 1, 2, \cdots, N$. Since $p_{\max} \geq p_i$, it's sufficient to prove

$$\frac{1+p_i}{1-p_i} f(r_i^*) + 1 \leq (1+p_i) \log_2 e \cdot \left(\frac{1}{(1-p_i) r_i^{\text{LB}}} + 1\right).$$

By Lemma 2, we have $r_i^* = \beta^* \cdot r_i^{\text{LB}} \geq \frac{1}{\log_2 e} \cdot r_i^{\text{LB}}$. So it's sufficient to prove

$$\frac{1+p_i}{1-p_i} f(r_i^*) + 1 \leq \frac{1+p_i}{(1-p_i) r_i^*} + (1+p_i) \log_2 e,$$

which is equivalent to

$$f(r_i^*) - \frac{1}{r_i^*} \leq (1-p_i) \log_2 e - \frac{1-p_i}{1+p_i}. \tag{39}$$

Recall $f(x) = 2\lfloor \frac{1}{x} \rfloor + 1 - x(\lfloor \frac{1}{x} \rfloor^2 + \lfloor \frac{1}{x} \rfloor)$. It can be proved that $f(x) - \frac{1}{x} \leq 3 - 2\sqrt{2}$ for all $0 < x < 1$ (the equality holds when $x = \frac{\sqrt{2}}{2}$). So the LHS of (39) is no greater than $3 - 2\sqrt{2} = 0.172$. On the other hand, the RHS of (39) is no less than $0.193 * \log_2 e - \frac{0.193}{1.807} = 0.172$ when $p_i \leq 0.807$. Therefore, (39) always holds. This completes our proof. ∎

Algorithm 5 A Solution to OPT-SD (Min-BW)

```
Input: $\boldsymbol{d}$.
Output: $W$ and $\boldsymbol{r}^*$.
1: Sort $\boldsymbol{d}$ in non-decreasing order.
2: Set $l_i = \frac{1}{d_i}$ for each $i = 1, 2, \cdots, N$. Solve OPT-$\hat{\boldsymbol{\gamma}}$ (by
   Algorithm 2 in [25]) and obtain an optimal solution $\hat{\boldsymbol{\gamma}}^*$.
3: Set $W = \lceil \sum_{i=1}^{N} \hat{\gamma}_i^* \rceil$. Set $\boldsymbol{\gamma}^* = \hat{\boldsymbol{\gamma}}^* \cdot W / (\sum_{i=1}^{N} \hat{\gamma}_i^*)$.
4: Re-sort $\boldsymbol{\gamma}^*$ and get $\boldsymbol{r}^*$.
```

C. An Algorithm to Solve OPT-SD (Min-BW)

Similar what we we did in A, to solve OPT-SD (Min-BW), we first transform $\boldsymbol{r}$ to its sorted version $\boldsymbol{\gamma}$. Here we sort $\boldsymbol{d}$ in non-decreasing order. Then we solve problem OPT-$\hat{\boldsymbol{\gamma}}$ by letting $l_i = \frac{1}{d_i}$ for each $i = 1, 2, \cdots, N$ and let $W^* = \lceil \sum_{i=1}^{N} \hat{\gamma}_i^* \rceil$.

A pseudocode for a solution to solve OPT-SD (Min-BW) is given in Algorithm 5.

D. A Proof of Theorem 2

First we develop a lower bound for the optimal objective W^*. For each source node i, to satisfy (17) for all t, there must be at least one transmission within any consecutive d_i time slots. Therefore,to satisfy (17), the transmission rate r_i must satisfy:

$$r_i \geq \frac{1}{d_i}. \tag{40}$$

Considering (40) and (4), we have

$$W^* \geq \sum_{i=1}^{N} \frac{1}{d_i}. \tag{41}$$

Then we derive an upper bound for the bandwidth W found by Eywa. By Lemma 1, we have

$$\sum_{i=1}^{N} \hat{\gamma}_i^* \leq \log_2 e \cdot \sum_{i=1}^{N} \frac{1}{d_i}. \tag{42}$$

From Step 3 in Algorithm 5, we have

$$W \leq \lceil \log_2 e \cdot \sum_{i=1}^{N} \frac{1}{d_i} \rceil. \tag{43}$$

Combining (41) and (43), we can complete the proof of Theorem 2. ∎

E. Performance Guarantee for Aion in [26]

In [26, Theorem 2], Liu *et al.* proved that

$$\frac{W_{\text{Aion}}}{W^*} \leq \frac{\left\lceil \sum_{i=1}^{N} \frac{1}{l_i^*} \right\rceil}{\left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil}, \tag{44}$$

where W_{Aion} is the bandwidth achieved by their proposed algorithm Aion and the vector $\boldsymbol{l}^* = [l_1^* \ l_2^* \ \cdots \ l_N^*]$ is the optimal solution of the following problem:

$$\begin{aligned}
\min_{l_1, l_2, \cdots, l_N \in \mathbb{R}} \quad & \sum_{i=1}^{N} \frac{1}{l_i} && \text{(45a)}\\
\text{s.t.} \quad & 1 \leq l_i \leq d_i, \ \text{for } i \in \{1, 2, \cdots, N\}, && \text{(45b)}\\
& \frac{l_i}{l_{i-1}} \in \mathbb{Z}^+, \ \text{for } i \in \{2, 3, \cdots, N\}, && \text{(45c)}\\
& l_i \in \mathbb{R}^+, \ \text{for } i \in \{1, 2, \cdots, N\}. && \text{(45d)}
\end{aligned}$$

We note that it is easy to prove

$$\frac{W_{\text{Aion}}}{W^*} \leq \frac{\left\lceil \sum_{i=1}^{N} \frac{1}{l_i^*} \right\rceil}{\left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil} \leq 2 \tag{46}$$

as follows: Consider the vector $\boldsymbol{l}' = [l_1' \ l_2' \ \cdots \ l_N']$ where $l_i' = 2^{\lfloor \log_2 d_i \rfloor}$ for each $i = 1, 2, \cdots, N$. Clearly, $\boldsymbol{l}'$ is a feasible solution of (45). As $\boldsymbol{l}^*$ is the optimal solution of (45), it holds that

$$\begin{aligned}
\frac{\left\lceil \sum_{i=1}^{N} \frac{1}{l_i^*} \right\rceil}{\left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil} &\leq \frac{\left\lceil \sum_{i=1}^{N} \frac{1}{l_i'} \right\rceil}{\left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil} = \frac{\left\lceil \sum_{i=1}^{N} \frac{1}{2^{\lfloor \log_2 d_i \rfloor}} \right\rceil}{\left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil}\\
&\leq \frac{\left\lceil \sum_{i=1}^{N} \frac{1}{2^{(\log_2 d_i - 1)}} \right\rceil}{\left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil} = \frac{\left\lceil 2 \cdot \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil}{\left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil} \leq 2,
\end{aligned}$$

which completes the proof. ∎

Algorithm 6 A Solution to DEC-SD

```
Input: $\boldsymbol{d}$, $W$.
Output: Whether or not a feasible $\boldsymbol{r}$ exists. If yes, find it.
1: Sort $\boldsymbol{d}$ in non-decreasing order.
2: Set $l_i = \frac{1}{d_i}$ for each $i = 1, 2, \cdots, N$. Solve OPT-$\hat{\boldsymbol{\gamma}}$ (by Algorithm 2 in [25]) and obtain an optimal solution $\hat{\boldsymbol{\gamma}}^*$.
3: if $\sum_{i=1}^{N} \hat{\gamma}_i^* \leq W$ then
4:     Determine that there exist a feasible $\boldsymbol{r}$. Construct a step-down $\boldsymbol{\gamma}$ with $\sum_{i=1}^{N} \gamma_i = W$ by letting some leading elements in $\hat{\boldsymbol{\gamma}}^*$ be 1. Re-sort $\boldsymbol{\gamma}$ and get the feasible $\boldsymbol{r}$ to DEC-SD.
5: else
6:     Determine that there does not exist a feasible $\boldsymbol{r}$.
7: end if
```

F. A Proof of Theorem 3

First we develop a lower bound for the optimal objective W^*. Recall that for any scheduler, we have (12). So to satisfy (20), we must have

$$r_i \geq \frac{1}{(1-p_i)(2\alpha_i - 1)}, \ i = 1, 2, \cdots, N. \tag{47}$$

Considering (47) and (4), we have

$$W^* \geq \sum_{i=1}^{N} \frac{1}{(1-p_i)(2\alpha_i - 1)}. \tag{48}$$

Then we derive an upper bound for the bandwidth W found by Eywa. Define function $g(\cdot)$ as:

$$h(x) = \frac{2\lfloor x \rfloor + 1 - x}{\lfloor x \rfloor^2 + \lfloor x \rfloor}. \tag{49}$$

It can be shown that for all $x > 1$, we have

$$h(x) \leq \frac{9}{8x}, \tag{50}$$

where the equality holds when $x = \frac{3}{2}$. In OPT-$\hat{\boldsymbol{\gamma}}$, we have

$$\sum_{i=1}^{N} l_i = \sum_{i=1}^{N} h\big(\frac{(1-p_i)(2\alpha_i - 1)}{1 + p_i}\big), \tag{51}$$

so we have

$$\sum_{i=1}^{N} l_i \leq \frac{9}{8} \cdot \sum_{i=1}^{N} \frac{(1-p_i)(2\alpha_i - 1)}{1 + p_i}. \tag{52}$$

By Lemma 1, we have

$$\sum_{i=1}^{N} \hat{\gamma}_i^* \leq \log_2 e \cdot \frac{9}{8} \cdot \sum_{i=1}^{N} \frac{(1-p_i)(2\alpha_i - 1)}{1 + p_i}. \tag{53}$$

From Step 3 in Algorithm 5, we have

$$W \leq \lceil \log_2 e \cdot \frac{9}{8} \cdot \sum_{i=1}^{N} \frac{(1-p_i)(2\alpha_i - 1)}{1 + p_i} \rceil. \tag{54}$$

Combining (48) and (54), we can complete the proof of Theorem 3. ∎

G. An Algorithm to Solve DEC-SD

To solve DEC-SD, we first transform $\boldsymbol{r}$ to its sorted version $\boldsymbol{\gamma}$. Here we sort $\boldsymbol{d}$ in non-decreasing order. Then we solve problem OPT-$\hat{\boldsymbol{\gamma}}$ by letting $l_i = \frac{1}{d_i}$ for each $i = 1, 2, \cdots, N$ and obtain an optimal solution $\hat{\boldsymbol{\gamma}}^*$. IF $\sum_{i=1}^{N} \hat{\gamma}_i^* \leq W$, then there exists a feasible $\boldsymbol{r}$ to DEC-SD. Otherwise, there does not.

A pseudocode for a solution to solve DEC-SD is given in Algorithm 6.