Hybrid Status Update Systems With Dedicated and Shared Servers

Sahan Liyanaarachchi[1], Sennur Ulukus[1], and Nail Akar[2]

[1]University of Maryland, College Park, MD, USA
[2]Bilkent University, Ankara, Türkiye

***Abstract*—Use of multi-path network topologies has become a prominent technique to assert timeliness in terms of age of information (AoI) and to improve resilience to link disruptions in communication systems. However, establishing multiple dedicated communication links among network nodes is a costly endeavor. Therefore, quite often, these secondary communication links are shared among multiple entities. Moreover, these multi-path networks come with the added challenge of out-of-order transmissions. In this paper, we study an amalgamation of the above two aspects, i.e., multi-path transmissions and link sharing. In this setting, we devise age-minimal scheduling schemes for a status update system composed of multiple sources sharing a single-server while having a separate dedicated server for each source so as to improve its timeliness.**

I. Introduction

Timeliness has become an indispensable feature that needs to be integrated into communication systems spanning from internet of things (IoT) applications [1] to cislunar communications [2], [3]. Vehicular networks used in autonomous driving, remote surgery systems, uncrewed lunar landing missions are a few avenues where timely communication is critical [2]–[5]. Therefore, the design of network infrastructure and development of sampling and scheduling policies to improve the timeliness of communication has been a broadly pursued research direction in the recent literature.

Age of information (AoI) has become a prominent metric of interest to quantify timeliness in communication systems [6]. A majority of the existing literature on timeliness of communication revolves around minimizing the time averaged AoI of status update systems. Recently, some of these efforts have shifted towards the analysis and optimization of status update systems with path diversity, i.e., systems with sources receiving service from multiple servers. Path diversity is a simple but effective design technique to improve upon the timeliness of communication [7]. However, despite their efficacy, these multi-server architectures suffer from a phenomenon known as *out-of-order transmissions* since the packets may not have to be received in the same order they were generated due to the availability of multiple paths for a single information source. This makes the AoI analysis an arduous task. Therefore, most recent works have been limited to either the AoI analysis of single-source multi-server systems or the construction of scheduling policies for multi-source single-server systems.

This work is done when N. Akar is on sabbatical leave as a visiting professor at University of Maryland, MD, USA, which is supported in part by the Scientific and Technological Research Council of Türkiye (Tübitak) 2219-International Postdoctoral Research Fellowship Program.

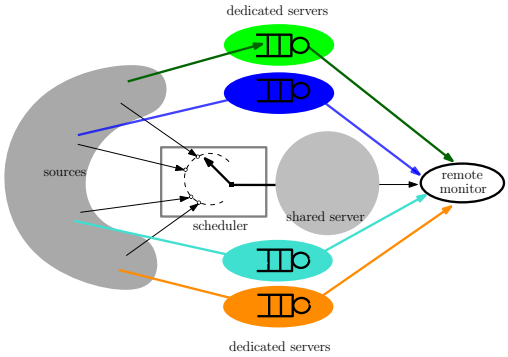

Fig. 1: Shared server status update system with N sources.

In this work, different from the approaches so far, we study a status update system that brings together the essence of single-source multi-server and multi-source single-server systems. In particular, we envision a system where multiple sources are scheduled through a single shared server, but in addition, each source has its own dedicated server with its unique service rate which may be different than others; see Fig. 1. These types of systems arise naturally in applications such as wireless relay networks [8] and mobile edge computing systems [9].

We study this hybrid status update system with *generate-at-will* (GAW) transmissions and present several *age-agnostic* scheduling policies for source transmissions across the shared channel (link). We analyze the AoI of the hybrid system under *cyclic* and *probabilistic* schedulers where we employ an absorbing Markov chain (AMC) formulation to tackle the complications arising from *out-of-order* transmissions. We provide the optimal probabilistic scheduler and present several heuristic-based cyclic schedulers which have superior AoI performance compared to the optimum probabilistic scheduler.

II. Related Work

One of the earliest works in the domain of multi-server systems roots back to [10] which uses a stochastic hybrid system (SHS) formulation to derive the expression for the average AoI of queues in tandem and later this analysis was expanded to queues in parallel in [11]. Both of these works revolved around the *random arrival* (RA) model where the status generation is governed by a Poisson arrival process.

ISBN 978-3-903176-65-2 © 2024 IFIP

SHS method was used in [12] to obtain the average AoI of dual queue/server status update system under the GAW model introduced in [13] for status generation. In [14], the AMC method was introduced as an alternative to SHS and it was used to derive the exact distribution of the steady state random variable associated with the AoI process of a single-server queue under a GAW model. This method was later utilized in [15] to develop a freeze and preemption policy to further improve the AoI of the dual-server status update system introduced in [12]. The effect of multiple parallel transmissions on the timeliness of status update systems was studied in [16] by employing the SHS method. All these works demonstrate the value of path diversity in networks for timely communication, resulting in substantial reduction in the AoI.

The closest to our work is [9], where the authors study a system of multiple sources where each source probabilistically opts to either process the data locally or through a shared edge server. This scenario was modeled as a push-based queuing system with multiple parallel servers and a single shared server under an RA status generation process. Moreover, the shared server is assumed to follow a *last-come-first-serve with preemption* policy which allows the use of the SHS method to find the average AoI and employ a mean field game model to optimize for the processing power and the selection probabilities of individual sources in the large population regime. This work does not encapsulate the essence of a scheduling problem but rather that of a game theory problem, where each source tries to maximize a local objective while accounting for the interference from other sources. In contrast to [9], in this work, we study a pull-based status update system with non-preemptive servers under a GAW model, explicitly focusing on the regime of age-agnostic scheduling schemes.

III. System model

Consider the multi-source dual-server (one dedicated server and one shared server for each source) status update system shown in Fig. 1, where each source has a dedicated channel to the remote monitor along with access to a single channel which is shared among all the sources based on a suitable scheduling policy. All of the channels are modeled with exponentially distributed service times. We assume a GAW model for the status generation process where each server immediately sends a pull request to one of the sources once it has finished transmitting (serving) the previous source packet. Thus, in this system, we have work-conserving servers that never idle.

Let S_n for $n \in \{1, 2, \ldots, N\}$ denote the direct channel (dedicated server) between source-n and the remote monitor with an exponentially distributed service time with parameter μ_n. Let S denote the shared channel (shared server) among the sources whose exponentially distributed service time for any of the sources has the service rate μ. We denote by $\Delta_n(t)$ the associated AoI process of source-n updates, which is a random process which linearly increases with time until the remote monitor receives an update from source-n with a fresher timestamp upon which the process drops to the service delay of the freshest source-n update. Since each source has two possible pathways for transmission, the updates that arrive at the monitor, can occasionally be out-of-order. In the event that we receive an older timestamp due to out-of-order transmissions, that particular update will simply be discarded by the monitor without decreasing the age.

We define $\Delta = \sum_{n=1}^{N} w_n \Delta_n$ as the weighted AoI where $\sum_{n=1}^{N} w_n = 1$ and w_n is the source specific weight assigned to source-n. Here, $\Delta_n(t)$ is the AoI process for source-n maintained at the remote monitor and Δ_n is the steady-state random variable associated with the process $\Delta_n(t)$. The goal of this work is to devise scheduling policies for source transmissions across the shared server so as to minimize the expected weighted AoI of the status update system.

IV. Scheduling Policies

To minimize the expected weighted AoI, we need to schedule the source transmissions through the shared server appropriately. Since we consider a GAW model with work-conserving servers, the scheduler must decide which source it must schedule once the current transmission through the shared server has finished. Due to the lack of communication between the scheduler and the dedicated servers (for each server/channel, we assume that the remote monitor only provides feedback for their respective channel transmissions and nothing more), the scheduler does not have the necessary information to decide whether the current data packet it is transmitting is obsolete (packets which are ultimately discarded) or not. This along with non-preemptive servers ensures that the packets can be discarded only once they reach the remote monitor. This renders any estimates of the AoI processes made at the scheduler side inaccurate and makes the use of age-dependent scheduling policies infeasible.

Therefore, in this paper, we consider only *age-agnostic policies* due to their simplicity, and also the lack of a need in such policies to maintain an exact replica of the AoI processes at the scheduler. Next, we describe the two age-agnostic scheduling schemes studied in this paper.

A. Probabilistic Scheduler (PS)

In the probabilistic scheduler (PS), once the shared server becomes free, the next source transmission is selected based on a probability mass function (pmf). Let $\{p_1, p_2, \ldots, p_N\}$ denote this pmf. In particular, source-n is scheduled for transmission with probability p_n once the current transmission is over.

B. Cyclic Scheduler (CS)

In the cyclic scheduler (CS), source transmissions are scheduled based on a fixed finite pattern which repeats itself. For example, let $N = 4$ and also let $C = [1, 2, 2, 3]$ be the cyclic pattern and C_n be a sample obtained from source-n. Then, the source transmissions on the shared server will be $C_1, C_2, C_2, C_3, C_1, C_2, C_2, C_3, \ldots$. Note that, even though we have four sources, based on the pattern C, we will never allocate a scheduling instance for the fourth source. This is to highlight the fact that any pattern which schedules at least one source schedule instance is a feasible cyclic schedule. In other words, the shared server need not cater to all sources.

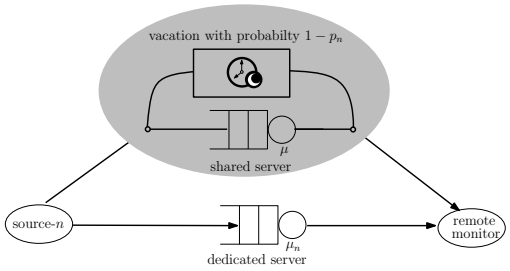

Fig. 2: Dual-server sub-problem for probabilistic scheduler (PS).

V. AoI Analysis

The AoI of source-n depends only on the transmissions emanating from its dedicated server and the shared server. When the shared server is occupied by another source transmission, from the perspective of source-n, it is as if the shared server is taking a vacation with rate μ. Therefore, one can reduce the N-source, $N+1$-server problem into N single-source dual-server problems, where the shared server takes a vacation based on either some probability (for PS) or according to a cyclic schedule (for CS) from the perspective of source-n. Fig. 2 illustrates the dual-server sub-problem for the PS.

Finding the average AoI of the dual-server problem is not a simple task. Due to the out-of-order arrival of packets at the remote monitor, the traditional graphical area based computation is not feasible in this scenario. Therefore, we employ the absorbing Markov chain (AMC) formulation, which was introduced in [14], for this purpose.

The key idea of the AMC method is to model the sample path followed by a newly joining packet into the system until it is successfully received (without being obsolete) by the remote monitor. Note that a single AoI cycle begins upon the successful reception of a packet and will continue to linearly increase until the next successful reception. Let P be a packet that just entered the system. At this time instance, we will initiate our AMC and we let it evolve until P is successfully received. The time at which P becomes successful will correspond to the beginning of a typical AoI cycle (see Fig. 3). In this case, we let the AMC further evolve until the next successful reception of a packet and consider this to be one absorbing state of our AMC. This absorbing state corresponds to a successful reception of a new packet following the successfully-received packet P, and therefore, is termed as the successful absorbing state. In the event that P becomes obsolete because a packet with a later timestamp has arrived at the remote server before P, then we consider that the AMC gets absorbed into another absorbing state termed as the unsuccessful absorbing state.

To compute the average AoI using the AMC method, we require the generator matrix of the AMC and the initial probability vector of the transient states of the AMC. Therefore, the AoI analysis using an AMC involves three main steps for a given source-n:

1) construction of the AMC,
2) construction of a recurrent Markov chain (RMC) to compute the initial probabilities of the AMC,

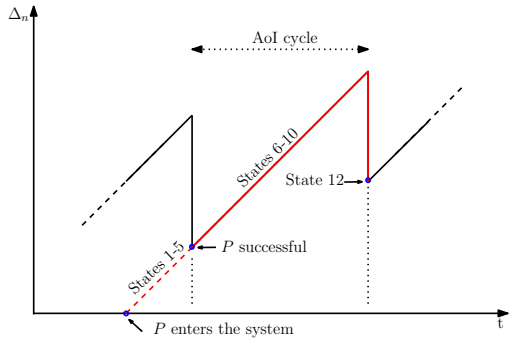

Fig. 3: Sample path of the AoI of source-n with the subsequent states of the PS AMC.

3) computation of the mean of Δ_n.

This procedure needs to be repeated for all sources.

Let $\boldsymbol{Q}$ be the generator matrix of the AMC obtained in the first step above, which is defined as follows,

$$\boldsymbol{Q} = \left[\begin{array}{c|c} \boldsymbol{U} & \boldsymbol{V} \\ \hline \boldsymbol{0} & \boldsymbol{0} \end{array}\right], \tag{1}$$

where $\boldsymbol{U}$ is the sub-generator matrix governing transitions among the transient states, $\boldsymbol{V}$ is the sub-generator matrix representing the transitions from the transient states to the two absorbing states, and $\boldsymbol{0}$ stands for a matrix of zeros of appropriate size. Let $\boldsymbol{\sigma}$ be the initial probability row vector of the transient states. Then, it is shown in [15] that the average AoI for source-n can be found as follows,

$$\mathbb{E}[\Delta_n] = -\frac{\boldsymbol{\sigma}\boldsymbol{U}^{-2}\boldsymbol{\theta}}{\boldsymbol{\sigma}\boldsymbol{U}^{-1}\boldsymbol{\theta}}, \tag{2}$$

where $\boldsymbol{\theta}$, termed as the transient vector, is a binary column vector of the same size as $\boldsymbol{\sigma}$ with ones in the entries corresponding to the transient states during which the packet P (which initiated the AMC in the first place) had already been successfully received by the remote monitor. On the other hand, $\boldsymbol{\sigma}$ is zero for states during which the original packet P is still in the system. The distribution and higher-order moments of Δ_n can also be obtained using the method of [14] but our focus in this paper is only on the mean AoI for source-n.

Now, we will present the AMC construction for the dual-server sub-problems for PS and CS.

A. Probabilistic Scheduler (PS) AoI

We analyze the AoI of PS for the dual-server sub-problem first. For PS, once the shared server becomes free, it will pull a packet from source-n with probability p_n, or will go on a vacation with probability $q_n = (1-p_n)$. Let P be the packet that initiates the AMC and let P_n and P_s be typical packets in server S_n and server S, respectively. Let the timestamps of the packets in S_n and S be denoted by T_n and T_s. Based on at which server a newly arriving packet P joins the system, the corresponding timestamps of packets in the servers, and whether S is on a vacation or not, we identify 12 states for the evolution of the AMC. These states are given in Table I.

The transitions between the states occur when either the dedicated server becomes free and pulls a new source-n packet

State	Description
1	P on S_n, S on vacation
2	P on S_n, S busy, $T_n \geq T_s$
3	P on S_n, S busy, $T_n < T_s$
4	P on S, S_n busy, $T_n \leq T_s$
5	P on S, S_n busy, $T_n > T_s$
6	P_n on S_n up to date, S on vacation
7	P_n on S_n up to date, P_s on S obsolete
8	P_n on S_n obsolete, P_s on S up to date
9	P_n on S_n up to date, P_s on S up to date
10	P_n on S_n obsolete, S on vacation
11	Unsuccessful absorbing state
12	Successful absorbing state

TABLE I: States of AMC for PS sub-problem for source-n.

into the system or when the shared server becomes free and decides to go on a vacation or pull a new source-n packet. We provide detailed descriptions for a few states transitions of the above AMC. The rest of the transitions are summarized in Table II and follow similar arguments.

- When in state 1, the packet P, which initiated the AMC, will be successfully received by the remote monitor with rate μ_n. In this case, a new up-to-date (not obsolete) packet will be pulled from source-n into the dedicated server, and hence, the ACM transitions to state 6 with rate μ. In state 1, the shared server, which is on vacation, will pull a new packet from source-n with rate $p_n\mu$. This new packet will have a later timestamp than P, and hence, the AMC transitions to state 3 with rate $p_n\mu$.
- In state 3, P on S_n will be successful and a new fresher packet will be pulled into server S_n with rate μ_n. In this case, since P has a earlier timestamp than the packet in S, the packet in S will be up-to-date even after the reception of P. Hence, the AMC will transition to state 9 with rate μ_n. Moreover, in state 3, the packet in S will be successful with rate μ, which will make the packet P on S_n obsolete since it has an older timestamp. Since P becomes obsolete, the AMC will be absorbed to state 11 with rate μ.
- In state 4, P on S will be successful and a new packet will be pulled onto S with rate $p_n\mu$ or P on S will be successful, and S will go onto a vacation with rate $q_n\mu$. In either case, once P becomes successful, the packet in S_n which has an older timestamp than P will become obsolete. Therefore, the AMC will transition to state 8 with rate $p_n\mu$ and to state 10 with rate $q_n\mu$. Additionally, when in state 4, the packet in S_n will be successful with rate μ_n and in this case a new packet with a fresher timestamp will be pulled into S_n. Therefore, the AMC will transition to state 5 with rate μ_n.
- When in states 6 to 10, the corresponding packet P which initiated the AMC has been successful. In these states, once an up-to-date packet finishes its transition, the AMC will be absorbed into state 12. If an obsolete packet finishes its transition, it will be discarded and a new packet will replace the obsolete packet in the corresponding server. If S is on a vacation, then a new packet will be pulled into S with rate $p_n\mu$.

Transition rates			Transition rates		
From	To	Rate	From	To	Rate
1	6	μ_n	6	12	μ_n
	3	$p_n\mu$		9	$p_n\mu$
2	7	μ_n	7	12	μ_n
	3	$p_n\mu$		9	$p_n\mu$
	1	$q_n\mu$		6	$q_n\mu$
3	9	μ_n	8	9	μ_n
	11	μ		12	μ
4	5	μ_n	9	12	$\mu_n + \mu$
	8	$p_n\mu$	10	6	μ_n
	10	$q_n\mu$		8	$p_n\mu$
5	11	μ_n			
	9	$p_n\mu$			
	6	$q_n\mu$			

TABLE II: Transition rates of AMC for PS sub-problem.

State	Description
$R1$	S_n busy, S on vacation
$R2$	S_n busy, S busy, $T_n \geq T_s$
$R3$	S_n busy, S busy, $T_n < T_s$

TABLE III: States of RMC for the PS sub-problem.

From Table II we can obtain the sub-generator matrix $\boldsymbol{U}$ that is required for the AoI computation. Next, we need to find the initial probability vector of the transient states of the AMC. For this purpose, we construct an RMC whose states represent the states of the system from the perspective of a newly joining packet. Based on the timestamps of the packets in each server and whether the server S is on vacation or not, we can define three states for PS. These states are given in Table III and transition rates of this RMC are shown in Fig. 4.

Let $\boldsymbol{\pi} = [\pi_1, \pi_2, \pi_3]$ be the stationary distribution of the above RMC. By algebraic manipulations, one can write $\boldsymbol{\pi}$ as,

$$\boldsymbol{\pi} = \begin{bmatrix} q_n & \frac{p_n\mu_n}{\mu_n+\mu} & \frac{p_n\mu}{\mu_n+\mu} \end{bmatrix}. \tag{3}$$

Using $\boldsymbol{\pi}$, we can find $\boldsymbol{\sigma}$ as follows. Let the net rate at which a new source-n packet joins the system be denoted by f. Note that a new packet enters the system with rate $\mu_n + p_n\mu$ when in any of the states of the RMC. Therefore, $f = \mu_n + p_n\mu$. Now, we establish the relation between the AMC and the RMC.

- When in state $R1$, a new source-n packet will join server S_n with rate μ_n and to server S with rate $p_n\mu$. The former event corresponds to the packet P initiating the AMC from state 1 and the latter corresponds to initiating the AMC starting from state 4.
- When in state $R2$ or $R3$, similar to state $R1$, a new source-n packet will join server S_n with rate μ_n and to server S with rate $p_n\mu$. If the new packet joins S_n, it would correspond to the event that the packet P initiated the AMC evolution starting from state 2 and if the new

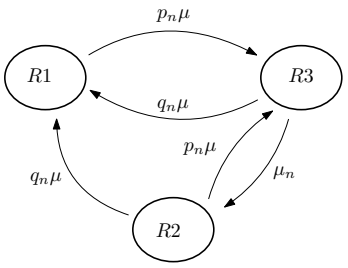

Fig. 4: RMC for PS sub-problem.

packet joins S, P would start its AMC from state 4.

Based on the above observations, it is clear that the AMC would be kicked off only from states 1, 2 or 4. Then, using $\boldsymbol{\pi}$, f and the established relations, the non-zero elements of the initial probability vector $\boldsymbol{\sigma}$ can be written as,

$$\sigma_1 = \frac{\mu_n \pi_1}{f}, \tag{4}$$

$$\sigma_2 = \frac{\mu_n(\pi_1 + \pi_2)}{f}, \tag{5}$$

$$\sigma_4 = \frac{p_n\mu(\pi_1 + \pi_2 + \pi_3)}{f}. \tag{6}$$

Thus, we write the row vector $\boldsymbol{\sigma}$ explicitly as follows,

$$\boldsymbol{\sigma} = \left[\tfrac{q_n\mu_n}{\mu_n+p_n\mu} \quad \tfrac{p_n\mu_n}{\mu_n+p_n\mu} \quad 0 \quad \tfrac{p_n\mu}{\mu_n+p_n\mu} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\right]. \tag{7}$$

Since states 6 through 10 are preceded by the event that P, which initiated the AMC, was successfully received by the remote monitor, the transient vector $\boldsymbol{\theta}$ can be written as,

$$\boldsymbol{\theta}^T = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1\end{bmatrix}. \tag{8}$$

Note that the states corresponding to the non-zero values of the vector $\boldsymbol{\theta}$ are essentially the states that exactly coincide with the AoI curve as shown in Fig. 3.Then by substituting $\boldsymbol{U}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\theta}$ in (2), the average age of source-n can be written in the following closed-form,

$$\mathbb{E}[\Delta_n] = \frac{p_n^2\mu^2(2\mu_n+\mu) + p_n\mu(2\mu_n+\mu)^2 + 2\mu_n(\mu_n+\mu)^2}{(\mu_n+\mu)^2(p_n\mu+\mu_n)^2}. \tag{9}$$

B. Cyclic Scheduler (CS) AoI

Now, we will present the AoI analysis for the CS sub-problem. Here, the shared server scheduling decisions will be based on a CS where at source-n scheduling instances, the shared server will pull a packet from source-n once it is free, and would be on vacation at other scheduling instances. Let P, P_n, T_n and T be as defined in Section V-A. Let $\tilde{C} = [c_1, c_2, \ldots, c_m]$ be the binary cyclic pattern with $c_i \in \{1, 2\}$ where $c_i = 1$ corresponds to scheduling a source-n transmission and $c_i = 2$ corresponds to taking a vacation.

Next, we will describe the construction of the AMC for CS. For brevity, we will reuse the state-space described in Table I to define a two-dimensional state vector for the transient states of the CS AMC. We define the transient states of this AMC as the pairs (i, j), where $i \in \{1, 2, \ldots, m\}$ denotes the ith scheduling instance of the pattern $\tilde{C}$ and $j \in \{1, 2 \ldots, 10\}$ corresponds to one of the states that was described in Table I.

Transition rates			
c_i	From	To	Rate
1	$(i,2)$	$(i,7)$	μ_n
		$(i+1,3)$ if $c_{i+1}=1$	μ
		$(i+1,1)$ if $c_{i+1}=2$	μ
	$(i,3)$	$(i,9)$	μ_n
		11	μ
	$(i,4)$	$(i,5)$	μ_n
		$(i+1,8)$ if $c_{i+1}=1$	μ
		$(i+1,10)$ if $c_{i+1}=2$	μ
	$(i,5)$	11	μ_n
		$(i+1,9)$ if $c_{i+1}=1$	μ
		$(i+1,6)$ if $c_{i+1}=2$	μ
	$(i,7)$	12	μ_n
		$(i+1,9)$ if $c_{i+1}=1$	μ
		$(i+1,6)$ if $c_{i+1}=2$	μ
	$(i,8)$	$(i,9)$	μ_n
		12	μ
	$(i,9)$	12	$\mu_n+\mu$
2	$(i,1)$	$(i,6)$	μ_n
		$(i+1,3)$ if $c_{i+1}=1$	μ
		$(i+1,1)$ if $c_{i+1}=2$	μ
	$(i,6)$	12	μ_n
		$(i+1,9)$ if $c_{i+1}=1$	μ
		$(i+1,6)$ if $c_{i+1}=2$	μ
	$(i,10)$	$(i,6)$	μ_n
		$(i+1,8)$ if $c_{i+1}=1$	μ
		$(i+1,10)$ if $c_{i+1}=2$	μ

TABLE IV: Transition rates for CS sub-problem.

Note that $c_i = 1$ implies that the shared server S is currently occupied by a source-n packet and therefore j can only take values in $\{2, 3, 4, 5, 7, 8, 9\}$. Similarly, $c_i = 2$ implies that the shared server S is currently on vacation and therefore j can only take values in $\{1, 6, 10\}$. As before, let the unsuccessful and successful absorbing states be denoted by states 11 and state 12, respectively. Thus, altogether we will have $7k+3(m-k)+2$ states for the AMC where k is the number of source-n occurrences within the pattern $\tilde{C}$. The transition rates of the CS AMC is summarized in Table IV where c_{m+1} is treated as c_1. The details of the state transitions are similar to the PS sub-problem with the only exception being that that whenever the shared server finishes its transmission, i changes from $i \rightarrow i+1$ (m changes to 1).

Next, we give the states of the RMC to compute the initial probabilities for the above AMC. Again, we will reuse the states of the RMC from the PS to define a two dimensional state vector for RMC of the CS. Let (i, R_j) be the states of the RMC, where $i \in \{1, 2, \ldots, m\}$ represents the ith scheduling instance of $\tilde{C}$ and R_j for $j = 1$ to 3 denote the status of the packets as defined in Table III. Here, $c_i = 1$ indicates that the shared server S is occupied by a source-n pattern and therefore the packet states can either be in $R2$ or $R3$. If $c_i = 2$, then

Transition states			
c_i	To	From	Rate
1	$(i, R2)$	$(i+1, R3)$ if $c_{i+1}=1$	μ
		$(i+1, R1)$ if $c_{i+1}=2$	μ
	$(i, R3)$	$(i, R2)$	μ_n
		$(i+1, R3)$ if $c_{i+1}=1$	μ
		$(i+1, R1)$ if $c_{i+1}=2$	μ
2	$(i, R1)$	$(i+1, R3)$ if $c_{i+1}=1$	μ
		$(i+1, R1)$ if $c_{i+1}=2$	μ

TABLE V: Transition rates of CS RMC.

that indicates that the S is currently on vacation, and therefore, packet status can only correspond to $R1$. Thus, the CS RMC comprises $2k+(m-k)$ recurrent states in total. The associated transition rates of the RMC are given in Table V.

Now, we need to find the total rate f_c at which a new packet enters the system. Note that, if we are in state (i, R_j) and $c_{i+1}=1$, then a new source-n packet would enter the system with rate $\mu+\mu_n$, and if $c_{i+1}=2$, then the rate would be μ_n. Let $\phi_{i,j}$ denote the stationary probability of the state (i, R_j). Then, we can write the quantity f as,

$$f_c = \sum_{(i,j)} \phi_{i,j}(\mu_n + \mu \mathbf{1}_{\{c_{i+1}=1\}}), \tag{10}$$

where the summation is over the state-space of the RMC, $c_{m+1}=c_1$ and $\mathbf{1}_{\{\cdot\}}$ is the indicator function. The corresponding relations between the AMC and RMC for the CS are as follows:

- When in state (i, R_1) and if $c_{i+1}=1$, a newly joining source-n packet entering server S_n would correspond to initiating the AMC starting from state $(i, 1)$. If the packet joins the shared server S instead, then the AMC will be initiated starting from state $(i+1, 4)$. These events would occur with rates μ_n and μ, respectively. If $c_{i+1}=2$, then only the former event would occur.
- When in state $(i, R2)$ or $(i, R3)$, if the new source-n packet joins server S_n, then the AMC would start from state $(i, 2)$ and this event occurs with rate μ_n. If $c_{i+1}=1$, and the packet joins server S, then the AMC would start from state $(i+1, 4)$ and this event occurs with rate μ.

The above relations indicate that the AMC can only start evolving from the states $(i, 2)$, $(i, 4)$ and $(i, 1)$. Let $\boldsymbol{\sigma}_c = \{\sigma_{i,j}\}$ represent the initial probability vector of the CS AMC where $\sigma_{i,j}$ denotes the probability of being in state (i, j) of the CS AMC. Using the above relations and the rate f_c, we can define the initial probabilities of the CS AMC as follows,

$$\sigma_{i,j} = \begin{cases} \frac{\mu_n \phi_{i,1}}{f_c}, & \text{if } c_i = 2, j = 1, \\ \frac{\mu_n(\phi_{i,2}+\phi_{i,3})}{f_c}, & \text{if } c_i = 1, j = 2, \\ \frac{\mu \phi_{i-1,1}}{f_c}, & \text{if } c_i = 1, c_{i-1} = 2, j = 4, \\ \frac{\mu(\phi_{i-1,2}+\phi_{i-1,3})}{f_c}, & \text{if } c_i = 1, c_{i-1} = 1, j = 4, \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where $c_0 = c_m$. The associated transient vector $\boldsymbol{\theta}_c$ will be given by,

$$\theta_{i,j} = \begin{cases} 1, & \text{if } c_i = 1, j \in \{7, 8, 9\}, \\ 1, & \text{if } c_i = 2, j \in \{6, 10\}, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Then, as before, by using (2), we can find the average AoI of source-n for the CS. However, since the matrices and vectors involved depend on the pattern considered, we do not provide a closed-form expression for this scheduler.

Next, we present an interesting relationship between the probabilistic and cyclic schedulers in Theorem 1. The proof follows from an sample path argument and is omitted here due to space limitations.

Theorem 1 *The optimal cyclic scheduler achieves a lower weighted AoI than the optimal probabilistic scheduler.*

VI. Optimal Probabilistic Scheduler (PS)

In this section, we pose the construction of the optimal PS as a convex optimization problem and obtain the optimal values for p_n. Using (2), we obtain the expected weighted AoI as,

$$\begin{aligned} \mathbb{E}[\Delta] = &\sum_n w_n \cdot \frac{(2\mu_n+\mu)\mu y_n + \mu\mu_n(\mu_n+\mu)}{(\mu_n+\mu)^2 y_n^2} \\ &+ \sum_n w_n \cdot \frac{(2\mu_n+\mu)}{(\mu_n+\mu)^2}, \end{aligned} \tag{13}$$

where $y_n = p_n\mu + \mu_n$. Since the first summation is a linear sum of functions $\frac{1}{y_n}$ and $\frac{1}{y_n^2}$ which are both convex for $y_n \geq 0$, the weighted AoI is a convex function with respect to y_n's. Therefore, finding the optimal probabilities reduces to the following convex problem:

$$\begin{aligned} \min_{y_n} \quad & \sum_n \left(\frac{(2\mu_n+\mu)\mu}{(\mu_n+\mu)^2} \cdot \frac{w_n}{y_n} + \frac{\mu\mu_n}{(\mu_n+\mu)} \cdot \frac{w_n}{y_n^2} \right) \\ \text{s.t.} \quad & \sum_n y_n = \mu + \sum_n \mu_n, \\ & y_n \geq \mu_n. \end{aligned} \tag{14}$$

Let a_n and b_n denote the coefficients of $\frac{1}{y_n}$ and $\frac{1}{y_n^2}$ terms in the objective function, respectively. We define the Lagrangian of the above optimization problem as follows,

$$\mathcal{L} = \sum_n \frac{a_n}{y_n} + \frac{b_n}{y_n^2} + \lambda\Big(\sum_n y_n - \eta\Big) + \sum_n \gamma_n(u_n - y_n) \tag{15}$$

where $\boldsymbol{y} = \{y_1, \ldots, y_n\}$, $\eta = \mu + \sum_n \mu_n$, and $\{\gamma_1, \ldots, \gamma_n\}$, λ are the Lagrange multipliers of the inequality and equality constraints, respectively. Since this satisfies the Slater's condition, the Karush-Kuhn-Tucker (KKT) conditions will yield the following sufficient conditions for optimality,

$$a_n y_n + 2b_n = (\lambda - \gamma_n) y_n^3, \tag{16}$$

$$\gamma_n(u_n - y_n) = 0, \tag{17}$$

$$\sum_n y_n = \eta. \tag{18}$$

Algorithm 1 An algorithm to find the optimal PS

Require: μ, $\{w_n,\mu_n\}_{n=1}^{N}$, λ_u sufficiently large,λ_l, ϵ sufficiently small

$a_n = \frac{w_n(2\mu_n+\mu)\mu}{(\mu_n+\mu)^2}$, $b_n = \frac{\mu\mu_n}{(\mu_n+\mu)} \quad \forall n$
$\eta = \mu + \sum_n \mu_n$
$y_n = 0 \quad \forall n$
while $|\eta - \sum_n y_n| > \epsilon$ **do**
  $\lambda = \frac{\lambda_u+\lambda_l}{2}$
  $f_n(y) \coloneqq \lambda y^3 - a_n y - 2b_n \quad \forall n$
  $z_n \leftarrow$ positive real root of $f_n(y_n) = 0$
  $y_n = \max\{z_n, u_n\}$
  if $\sum_n y_n > \eta$ **then**
    $\lambda_l = \lambda$
  else
    $\lambda_u = \lambda$
  end if
end while
Output: $p_n = \frac{y_n-\mu_n}{\mu} \quad \forall n$

Since $\gamma_n \geq 0$ and any feasible y_n satisfies $y_n \geq \mu_n$, we can conclude that $\lambda > 0$. From (17), we have that if $\gamma_n > 0$, then $y_n = u_n$ and if $y_n > u_n$, then γ_n will be zero. If $\gamma_n = 0$, then note that for a fixed λ, the y_n that satisfies (16) is simply the intersection of the straight line $a_n y_n + b_n$ and the cubic polynomial λy_n^3. Since $\lambda, a_n, b_n > 0$, a simple geometric interpretation reveals that (16) has only one positive real root in this particular scenario. Moreover, this root will decrease as we increase λ and can be found using Cardano's formula. To find the optimal λ, we can use a simple bisection search. Algorithm 1 gives the pseudo-code for finding the optimal probabilities using the above steps.

VII. Cyclic Scheduler (CS) Construction

In this section, we present two heuristic-based approaches for the construction of the cyclic scheduler where in one we refurbish the Insertion Search (IS) algorithm introduced in [17] and in the other we utilize the optimal probabilities for the probabilistic scheduler computed in Section VI.

A. Insertion Search (IS) Algorithm

Variants of the IS algorithm have been readily used in the past and have been shown to perform well for cyclic scheduler construction to minimize the weighted AoI [17]–[19]. We use this algorithm with slight modifications to fit to our setting. The IS algorithm constructs the cyclic schedule in an iterative manner by improving upon the current best schedule obtained in each iteration. Starting from an empty schedule, in each iteration, we select the source-position pair that achieves the lowest weighted AoI to construct a new cyclic schedule. Then, this new cyclic schedule is passed on to the next iteration where the same steps are repeated. At any iteration, if the weighted AoI does not improve, we will continue to explore a few more iterations and stop if no further improvement is observed. Then, we select the best schedule from among all the cyclic schedules that were constructed at each iteration.

B. Probability Aided Cyclic (PAC) Scheduler

Despite the versatility of the IS algorithm, it comes with a high computational complexity. Within each iteration of IS, the weighted AoI needs to be computed several times and this involves inverting a large matrix each time. This limits its applicability for systems with a large number of sources, especially in situations where the server rates or our estimates of the server rates are prone to change frequently. To overcome this drawback, we introduce the PAC scheduler which is constructed using the optimal probabilities p_n^* of the PS. In the PAC scheduler, we aim to construct the cyclic schedule such that the proportion of time dedicated for source-n tallies with p_n^* of the optimal PS. Next, we select a schedule length K and find the number of instances K_n that should be allocated for source-n such that $\sum_n K_n = K$. Then, we try to uniformly spread the source instances within the schedule to construct the PAC scheduler. Both these problems, i.e., finding K, K_n and uniformly spreading the scheduling instances, have been studied in the past. [18] introduces a deficit round robin (DRR) algorithm which constructs an almost uniform schedule given the optimal source frequencies within the schedule. Thus, by filtering out the sources with $p_n* > 0$ and replacing the optimal frequencies with p_n^*, we propose to use the DRR algorithm introduced in [18] to construct our PAC scheduler.

VIII. Numerical Results

In this section, we evaluate and compare the performances of the three scheduling schemes. In this first experiment, we consider a system with three sources for which we fix the weights given to the sources, and vary the dedicated service rate of source-1. As shown in Fig. 5, the cyclic scheduling schemes outperform the optimal PS, and IS has only a slight edge over the PAC scheduler. This validates the rationale behind selecting the optimal scheduling probabilities as a suitable substitute for the optimal source frequencies when constructing the cyclic schedulers in PAC.

In the next experiment, we consider a system of four sources, where we fix the service rates of the sources and vary the weight given to the first source. The variation of the weighted AoI for this scenario is shown in Fig. 6. As depicted, cyclic scheduling schemes exhibit superior AoI performance compared to the probabilistic scheme. Moreover, we observe that, the weighted AoI first increases and then starts to decrease with w_1. This is because, as w_1 increases initially, its contribution to the weighted AoI increases, but as w_1 increases further, the shared channel tends to favor source-1 more, and hence, decreases the average AoI of source-1, which in turn reduces the weighted AoI of the system.

In the next experiment, we evaluate how the optimal probabilities of PS vary with the service rate of the shared server for a simple three source system whose weights are the same but with different dedicated service rates. As shown in Fig. 7, when the shared server rate is small compared to the dedicated server rates, the shared server tries to give more scheduling priority to the source with the lowest dedicated rate. However, as μ is comparatively large compared to the

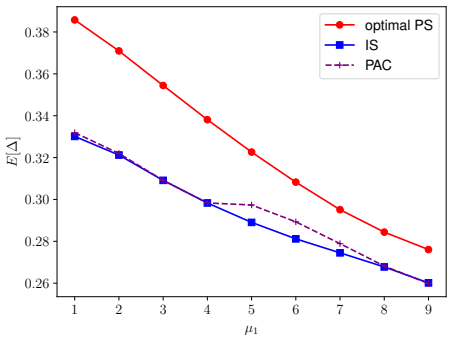

Fig. 5: Variation of the weighted AoI with μ_1 for $w_1 = 0.3$, $w_2 = 0.5$, $w_3 = 0.3$, $\mu = 8$, $\mu_2 = 2$ and $\mu_3 = 3$.

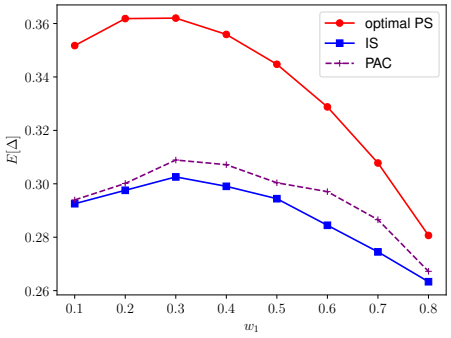

Fig. 6: Variation of the weighted AoI with w_1 for $N = 4$, $w_n = \frac{1-w_1}{3}$, $\mu = 10$ and $\mu_n = n$.

dedicated service rates, equal priority is given to all the sources. In this case, it is as if the dedicated servers are non-existent from the perspective of the shared server, and almost all packets served through the dedicated links will turn out to be obsolete. Thus, when μ is large when compared to the dedicated rates, our problem simplifies to to a single-server multi-source scheduling problem.

IX. Conclusion

We provided an AMC formulation to derive expressions for the weighted AoI of a status update system comprising multiple dedicated servers along with a single shared server. Through rigorous convex optimization, we have provided the optimal probabilistic scheduler along with several heuristics for developing cyclic schedulers which are shown to be superior in performance. As discussed in the numerical results section, when the service rate of the shared server is high, the packets through the dedicated servers would turn out to be obsolete most of the time. Therefore, the problem of selecting an appropriate service rate for the shared server so as to improve efficiency of the system (i.e., to minimize the number of obsolete packets) is a interesting future research direction for the system studied in this work.

References

[1] M. A. Abd-Elmagid, N. Pappas, and H. S. Dhillon. On the role of age of information in the internet of things. *IEEE Communications Magazine*, 57(12):72–77, December 2019.

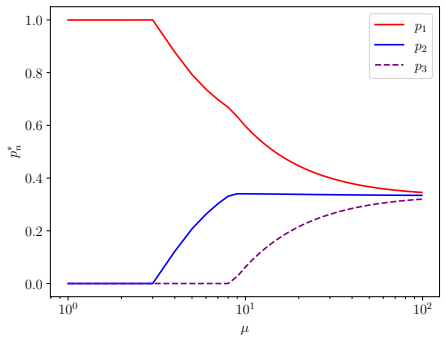

Fig. 7: Variation of the optimal scheduling probabilities with μ (log scale) for $N = 3$, $w_n = \frac{1}{3}$, $\mu_1 = 4$, $\mu_2 = 7$ and $\mu_3 = 10$.

[2] A. Yuan, Z. Hu, Q. Zhang, Z. Sun, and Z. Yang. Towards the age in cislunar communication: an aoi-optimal multi-relay constellation with heterogeneous orbits. *IEEE Journal on Selected Areas in Communications*, 42(5):1420–1435, May 2024.

[3] S. Liyanaarachchi, S. Mitrolaris, P. Mitra, and S. Ulukus. 6G at $\frac{1}{6}g$: The future of cislunar communications. Available online at arXiv:2407.16672.

[4] C. Xu, Q. Xu, J. Wang, K. Wu, K. Lu, and C. Qiao. AoI-centric task scheduling for autonomous driving systems. In *IEEE Infocom*, May 2022.

[5] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis. 5G-enabled tactile internet. *Jour. Sel. Areas Commun*, 34(3):460–473, March 2016.

[6] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus. Age of information: An introduction and survey. *IEEE Journal on Selected Areas in Communications*, 39(5):1183–1210, May 2021.

[7] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides. Effect of message transmission path diversity on status age. *Journal on Selected Areas in Communications*, 62(3):1360–1374, March 2016.

[8] A. Maatouk, M. Assaad, and A. Ephremides. The age of updates in a simple relay network. In *IEEE ITW*, November 2018.

[9] S. Aggarwal, M. A. uz Zaman, M. Bastopcu, S. Ulukus, and T. Başar. A mean field game model for timely computation in edge computing systems. Available online at arXiv:2404.02898.

[10] R. D. Yates. Age of information in a network of preemptive servers. In *IEEE Infocom*, April 2018.

[11] R. D. Yates. Status updates through networks of parallel servers. In *IEEE ISIT*, June 2018.

[12] K. Lang Z. Chen, H. H. Yang, N. Pappas, M. Wang, and T. Q. S. Quek. Age of information of a dual queue status update system: A stochastic hybrid systems method. *IEEE Communications Letters*, 27(7):1714–1718, July 2023.

[13] R. D. Yates. Lazy is timely: Status updates by an energy harvesting source. In *IEEE ISIT*, June 2015.

[14] N. Akar and E. O. Gamgam. Distribution of age of information in status update systems with heterogeneous information sources: An absorbing Markov chain-based approach. *IEEE Communications Letters*, 27(8):2024–2028, August 2023.

[15] N. Akar and S. Ulukus. Age of information in a single-source generate-at-will dual-server status update system. Available online at arXiv:2404.01229.

[16] Z. Chen, K. Lang, N. Pappas, H. H. Yang, M. Wang, Z. Tian, and T. Q. S. Quek. Timeliness of status update system: The effect of parallel transmission using heterogenous updating devices. Available online at arXiv:2405.16965.

[17] E. O. Gamgam, N. Akar, and S. Ulukus. Cyclic scheduling for age of information minimization with generate at will status updates. In *CISS*, March 2024.

[18] N. Akar, S. Liyanaarachchi, and S. Ulukus. Scalable cyclic schedulers for age of information optimization in large-scale status update systems. In *IEEE Infocom*, May 2024.

[19] S. Liyanaarachchi, S. Ulukus, and N. Akar. Minimizing the age of two heterogeneous sources with packet drops via cyclic schedulers. In *Asilomar Conference on Signals, Systems and Computers*, October 2024.