Timely and Energy-Efficient Multi-Step Update Processing

Vishakha Ramani, Ivan Seskar, Roy D. Yates
WINLAB, Rutgers University
Email: {vishakha, seskar, ryates}@winlab.rutgers.edu

***Abstract*—This work explores systems where source updates require multiple sequential processing steps. We model and analyze the Age of Information (AoI) performance of various system designs under both parallel and series server setups. In parallel setups, each processor executes all computation steps with multiple processors working in parallel, while in series setups, each processor performs a specific step in sequence. In practice, processing faster is better in terms of age but it also consumes more power. We identify the occurrence of wasted power in these setups, which arises when processing efforts do not lead to a reduction in age. This happens when a fresher update finishes first in parallel servers or when a server preempts processing due to a fresher update from preceding server in series setups. To address this age-power trade-off, we formulate and solve an optimization problem to determine the optimal service rates for each processing step under a given power budget. We focus on a special case where updates require two computational steps.**

I. Introduction

Emerging mobile real-time applications such as Augmented Reality (AR) and Mixed Reality (MR) mandate a comprehensive understanding of the surroundings. Sensors supporting such applications generate abundant data that induces a computationally intensive workload not feasible on resource-constrained mobile devices. This challenge is addressed by offloading the computation to a nearby edge node [1]. Examples include personalized AR tours and pedestrian safety systems at smart city intersections [2], both requiring low latency and timely responses.

Typically, sensor (source) update processing consists of sequential computation steps. For example, the object detection involves sequential tasks like pre-processing on input images, feature extraction, followed by object classification. For any edge-computing platform there can be different modes of processing a source update. These modes are usually dictated by system design choices in terms of number of processors deployed to process an update, the underlying communication mechanism between processors as well as energy consumption constraint.

One approach involves using n loosely coupled processors to process an update requiring n computational steps. In this configuration, each processor performs one step in the update's processing pipeline. This setup introduces an asynchronous pipeline mechanism, where the output of a processor serves as the input to the subsequent processor. From a queueing theory perspective, this can be modeled as a tandem queue (also known as a series queue) with n servers (as depicted in Fig. 1). In contrast, another processing paradigm involves the use of multiple parallel processors, where each processor independently executes all n computation steps, as illustrated in Fig. 2.

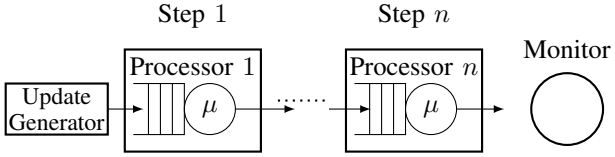


Fig. 1. Processors in series for source update processing. Processor i is responsible for computation step i.

A. AoI and Multi-Step Processing Systems

Regardless of the mode of processing, it is imperative that the system delivers timely processed updates such that the age at the end user, hereafter referred to as *monitor*, is minimized. Despite its importance, the study of timely multi-step update processing remains unexplored in AoI literature. This work aims to bridge that gap by investigating the age performance of two-step ($n = 2$) update processing systems, a fundamental building block for more complex processing systems. Even within this seemingly simple two-step framework, rudimentary questions arise. For instance, which configuration-—series or parallel processing—-proves more effective in maintaining timeliness? Answering this question, however, is far from straightforward and poses a considerable challenge.

In a setup with two servers in series, modeled as a tandem queue, each service facility may operate under different service disciplines, such as lossless First-Come-First-Served (FCFS) or lossy Last-Come-First-Served (LCFS) with preemption in either waiting or service steps. While there is an extensive literature on age performance of fresh arrivals in single-source single-server queues employing various service disciplines (see [3] and the references therein), in our work however, updates arrive at server 2 from server 1 with some existing age, which must be accounted for by the analysis.

Furthermore, the analysis of parallel processor setups with only two servers is equally non-trivial. A server may be "late" in delivering an update, while another parallel server has already delivered an update with lesser age, rendering the former's delivery inconsequential in terms of age reduction. This scenario underscores the necessity of developing a novel analytical framework to properly evaluate such systems.

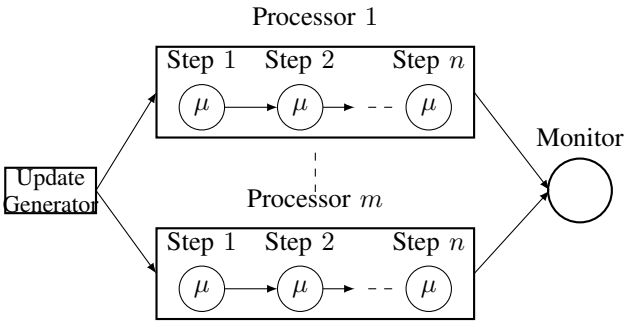


Fig. 2. Parallel processors setup for source update processing. Each processor executes all n computation steps.

B. Age-Power Trade-off

We argue that it is not straightforward to determine which setup – series or parallel servers – performs better solely based on age. While faster processing improves age performance, it also consumes more power. A deeper analysis reveals that both configurations are susceptible to a phenomenon we term *wasted power*, where computational resources are expended on processing updates that ultimately do not contribute to reducing the age.

In parallel server setups, wasted power occurs when one server completes processing a fresher update before others, thereby rendering the subsequent efforts of the remaining servers, still processing older updates, inconsequential in terms of age reduction. The resources dedicated to these outdated updates are thus wasted. In series server setups, wasted power may occur when a server preempts its current update upon receiving a fresher update from the preceding server. The computational resources expended in processing the preempted update can be identified as wasted. Similarly, in some scenarios, a server may discard updates received from the preceding server, nullifying the work performed on discarded update.

An additional inefficiency in series setups is server idleness. Servers may remain inactive while awaiting the completion of the preceding server's processing task. This idle time represents lost computational potential, as these resources could have been employed to process other updates, potentially improving age performance.

C. Paper Outline and Main Contributions

In this work, we address the age-power trade-off and pose the following question: How can we optimize the allocation of computational resources to minimize wasted power while simultaneously reducing age? We explore this trade-off by focusing primarily on identifying the optimal service rates for each step that minimize age when the system is subjected to a total power consumption constraint. We start with Section II which introduces the system parameters, the power consumption model, and the optimization problem formulation.

We focus on series server setups in Section III. Specifically, we analyze the power consumption and age performance of various preemptive and non-preemptive tandem queue models. These models differ in the queuing discipline employed at server 2, while server 1 generates updates at will. Particularly, we study four models under series server setup: M/M/1*, M/M/1/2*, M/M/1/1, Synchronous Sequential Service (SSS). In the M/M/1* model, server 2 employs preemption in service, discarding the current update upon the arrival of a new one. In the M/M/1/2* model, server 2 includes a waiting room of capacity 1, where updates in the waiting room can be preempted by new arrivals. In the M/M/1/1 model, server 2 blocks and discards incoming updates when it is busy. In SSS model, a new update begins service only when both servers are idle.

In Section IV, our focus turns to parallel server setups. We begin by analyzing the power consumption and age in the baseline model, Parallel Synchronous Sequential Service (P-SSS), where the two servers function independently. We introduce a novel approach using the Stochastic Hybrid Systems (SHS) methodology to derive a system of linear equations for calculating the average age at the monitor. Subsequently, we analyze power consumption and age performance of two additional policies, Parallel Coordinated Alternating Freshness (P-CAF) and Parallel Shared Intermediate Update (P-SIU), where servers synchronize their operations by leveraging information about each server's current step of processing.

In Section V, we numerically solve the optimization problem for various series and parallel server models introduced in Sections III and IV. This includes a comparative analysis of the minimum achievable age for each model, obtained by optimizing the computational step rates subject to a given power constraint. In Section VI, we conclude by discussing several open problems emerging from this work and propose directions for future research.

D. Related Work

In the Age-of-Information literature, various studies have focused on age in network of queues [4]–[6]. For a line network model of last-come-first served (LCFS) queue with preemption in service, it was shown that node i with service rate μ_i contributes $1/\mu_i$ to the age at the monitor [6]. Authors in [7] derived average age for two first-come-first served non-preemptive queues in tandem. [8] models the communication and computation delay in edge computing framework and derives the PDF of Peak Age-of-Information (PAoI) for M/M/1-M/D/1 and M/M/1-M/M/1 tandem queues. [9] develops a recursive framework to derive the mean peak age of information for N heterogeneous servers in tandem. [10] obtains the distribution of the age and peak age in a system of two tandem queues connected in series with packet prioritization in the second queue.

Age for M/M/2 and M/M/∞ systems was studied in [11] to demonstrate the advantage of having the message transmission path diversity for status updates. [12] studies the age-delay trade-off in G/G/∞ queue. [13] observes that a single M/M/1 queue has better age performance than the independent parallel M/M/1 queues with the same total capacity. [14] analyzed age in network of parallel finite identical and memoryless

servers, where each server is an LCFS queue with preemption in service. However, our work deviates from [11], [14] in that we relax the assumption of memoryless processing times for updates. This key difference renders the SHS analysis used in [14] inapplicable to our scenario.

On the other hand, with respect to general queuing theory, the problem of optimal service rate control has been extensively studied across various types of queuing networks, ranging from single-queue single server model [15]–[17], multiple queue single server model [18], to multiple server, multiple queue model [19]. In studies focused on single-server queue systems, the general setting involves a nondecreasing cost of service and holding costs that are nondecreasing functions of queue length, with rewards associated with customers entering the queue. The arrival rate, λ, and/or the service rate, μ, are subject to control. The objective in these studies is typically to minimize the expected total discounted cost or the long-run average cost. In various systems, the authors establish optimality of monotone policies i.e. optimal arrival rates are non-increasing in number of arrivals and optimal service rates are non-decreasing in queue length as observed in [20].

Several authors have considered tandem queue systems with Poisson arrivals at rate λ and two memoryless servers, serving at rates μ_1 and μ_2 at first and second queue respectively. The first study on optimal service control in tandem queues was conducted by Rosberg et al. [21]. In this study, the authors examined a setting where the service rate at server 1 is selected as a function of the system's state, defined as the tuple of queue lengths at each server, while the service rate at server 2 is held constant. Considering only holding cost and no operating cost, the authors established the optimality of switchover policies, where the optimal rate at server 1 is determined by a switching function of the queue length at server 2.

Authors in [20] considered a cyclic queue system where a number of ·/M/1 queues are arranged in a cycle. Considering a system cost comprising of both holding and operating costs, the authors determined the optimal policy has a transition-monotone decision rule, where when a customer moves from queue i to the following queue, the optimal service rate at queue i does not increase, and optimal service rate at queue $j, j \neq i$ does not decrease. Optimal control of service rates of a tandem queue under power constraints is studied in [22]. The authors assume that the service rate is linear to the power allocated to that server and the sum of service rates must not exceed the given power budget. An iterative algorithm is proposed to find the optimal service rates.

II. System Model Overview

A. Power Consumption Model

Power dissipation in digital CMOS circuits is primarily attributed to dynamic power, short circuit losses, and transistor leakage currents [23]. Among these, dynamic power consumption is currently the main component in high-performance microprocessors. Dynamic power, driven by the periodic switching of capacitors, can be approximated by the well-known formula

$$P = AC_L V^2 f, \tag{1}$$

where A and C_L denote the Activity Factor (AF) and loading capacitance, respectively, V is the supply voltage, and f is the clock frequency [24]. According to alpha-power law MOS model [25], $f \propto V^{\alpha_c - 1}$, where α_c, called the velocity saturation index, is a technology dependent factor, typically ranging between 1 and 2. This implies that voltage can be expressed as $V = k_V f^{1/(\alpha_c - 1)}$, with k_V denoting a constant of proportionality. This results in a power model for a processor running at frequency f as

$$P = k_P f^{\alpha}, \tag{2}$$

where $\alpha = (1 + \alpha_c)/(\alpha_c - 1) \geq 3$ and $k_P = AC_L k_V^2$ is a constant that incorporates the constants A and C_L from (1). According to [26], for a 25 μm technology, α_c is likely to be in range $[1.3, 1.5]$. For our numerical evaluations, we fix the velocity saturation index at $\alpha_c = 1.5$, which corresponds to $\alpha = 5$.

B. Processing Speed and Power Consumption

We assume that processing a source update involves a sequence of two computational steps. Further, we consider that the system uses two servers, which may operate in series or in parallel configurations.

In general, each step i in update processing involves a random computation workload C_i, measured in CPU cycles. This randomness is due to many reasons such as CPU being shared by many applications, background daemons, garbage collection and other system-level activities [27]. While ideally, the workload in terms of CPU cycles required to process the same update would remain consistent across servers, variations occur due to these random factors.

For simplicity, we assume that the workloads for the two steps are identically distributed, with means $\mathrm{E}[C_1] = \mathrm{E}[C_2] = \mathrm{E}[C]$. Each step i is executed at a constant processing frequency f_i (CPU cycles per unit time), resulting in an execution time $T_i = C_i/f_i$. The average service rate for step i is therefore

$$\mu_i = \frac{1}{\mathrm{E}[T_i]} = \frac{f_i}{\mathrm{E}[C_i]} = \frac{f_i}{\mathrm{E}[C]}. \tag{3}$$

While executing step i, the processor consumes power as described by (2), resulting in instantaneous power consumption: $P_i = k_P f_i^{\alpha} = k_P(\mathrm{E}[C]\mu_i)^{\alpha}$. To simplify the relation between power consumption and processor speed, we assume a power unit such that $k_P = 1$. Thus, in our analysis, the expected power consumed by a processor while executing step i simplifies to $P_i = (\mathrm{E}[C]\mu_i)^{\alpha}$. Finally, we assume that an idle processor consumes negligible power.

We assume that the execution time for step i follows an independent exponential distribution with rate parameter μ_i, i.e., $T_i \sim \exp(\mu_i)$. Let $\mathcal{Q}$ represent the discrete state space of a given update processing system. That is, a state $q \in \mathcal{Q}$ specifies for each processor whether it is idle or executing

update step 1 or step 2. With π_q representing the stationary probability of state $q \in \mathcal{Q}$, the power consumption associated with step i execution in state q is represented as $r_{q,i}(\mu_i)$. For example, if in state q, both processors are executing step 1 for a particular update, $r_{q,1}(\mu_1) = 2\,\mathrm{E}[C]^\alpha \mu_1^\alpha$ while $r_{q,2}(\mu_2) = 0$ because no processor is executing step 2. In general, by defining $n_{q,i}$ as the number of processors working on step i in state q,

$$r_{q,i}(\mu_i) = n_{q,i}(\mathrm{E}[C]\mu_i)^\alpha \tag{4}$$

The average power consumption associated with step i is then $\sum_{q\in\mathcal{Q}} \pi_q r_{q,i}(\mu_i)$.

C. Problem Formulation

We enforce that the total power consumption at the two servers is limited by a power budget. Specifically, with P representing the total power budget and $\pi_q(\mu_1, \mu_2)$ elucidating the dependence of π_q on μ_1 and μ_2, the power consumption at the two servers must satisfy the constraint

$$\sum_{q\in\mathcal{Q}} \pi_q(\mu_1,\mu_2)[r_{q,1}(\mu_1) + r_{q,2}(\mu_2)] \le P. \tag{5}$$

The age at the monitor, denoted by $\Delta(\mu_1, \mu_2)$, is a function of the service rates μ_1 and μ_2, and is influenced by the policy at each server. Our objective is to minimize the age $\Delta(\mu_1, \mu_2)$ at the monitor by controlling the service rates μ_1 and μ_2, subject to the power constraint (5). From (4) and (5), the optimization problem can be stated as:

$$\text{minimize} \quad \Delta(\mu_1,\mu_2) \tag{6a}$$

$$\text{subject to} \quad \sum_{q\in\mathcal{Q}} \pi_q(\mu_1,\mu_2) \sum_{i=1}^{2} n_{q,i}\mu_i^\alpha \le P/\,\mathrm{E}[C]^\alpha, \tag{6b}$$

$$\mu_1, \mu_2 \ge 0. \tag{6c}$$

To solve the optimization problem (6), our strategy is to define $\rho = \mu_1/\mu_2$, and focus on a class of systems in which the stationary probabilities π_q can be expressed as functions of ρ i.e., $\pi_q(\mu_1, \mu_2) \equiv \pi_q(\rho)$. In the sequential service system of Figure 1, ρ represents the total offered load from server 1 to server 2. In general, ρ characterizes the effort that processors make on step 1 relative to step 2

Substituting $\mu_1 = \rho\mu_2$ into the constraint (6b) results in

$$\sum_{q\in\mathcal{Q}} \pi_q(\rho)(n_{q,1}\rho^\alpha\mu_2^\alpha + n_{q,2}\mu_2^\alpha) \le P/\,\mathrm{E}[C]^\alpha. \tag{7}$$

With the observation that

$$\overline{N}_i(\rho) = \sum_{q\in\mathcal{Q}} \pi_q(\rho) n_{q,i} \tag{8}$$

is the average number of processors working on step i, we see that (7) simplifies to the upper bound on the step 2 service rate

$$\mu_2^\alpha \le \frac{P}{\mathrm{E}[C]^\alpha(\rho^\alpha\overline{N}_1(\rho) + \overline{N}_2(\rho))}. \tag{9}$$

Defining the Power-Weighted Processor Activity (PWPA)

$$\overline{N}(\rho) \equiv \rho^\alpha\overline{N}_1(\rho) + \overline{N}_2(\rho), \tag{10}$$

the optimization problem (6) can be reformulated as

$$\text{minimize} \quad \Delta(\mu_2,\rho) \tag{11a}$$

$$\text{subject to} \quad \mu_2^\alpha \le \frac{P}{\mathrm{E}[C]^\alpha\overline{N}(\rho)}. \tag{11b}$$

$$\mu_2 \ge 0, \text{and } \rho \ge 0. \tag{11c}$$

The parameter α is fixed by the technology, while the power budget P and CPU demand C are system parameters. For a fixed ρ, the service rates μ_2 and $\mu_1 = \rho\mu_2$ determine the average age at the monitor. We will see in the systems we study that the age is typically minimized by choosing μ_2 as large as possible subject to the upper bound (11b). What remains is choosing the right value of ρ. A larger ρ implies that step 1 is executed faster relative to step 2, resulting in more energy being allocated to step 1. This results in fresher updates reaching step 2, but this could be wasting the energy used in step 1, as updates from step 1 could either be discarded or updates in step 2 could be preempted. On the other hand, a smaller ρ means that step 1 is slower relative to step 2. In this case, step 1 processed updates arrive at step 2 with higher age with the system not feeding enough updates to step 2.

For a system design choice, finding an optimal ρ^* allows us to determine the corresponding optimal service rate μ_2^* using the constraint in (11b), which in turn yields the optimal age $\Delta(\mu_2^*, \rho^*)$. The subsequent sections analyze the system models studied in this work. Specifically, we detail the relevant Markov Chain for each model, including its state space $\mathcal{Q}$, stationary probabilities π_q, and transitions. Analytical expressions for $\Delta(\mu_2, \rho)$ and $\overline{N}(\rho)$ are provided for each model, enabling the formulation of optimization problem (11) for each model.

D. SHS Overview

We use Stochastic Hybrid Systems (SHS) [28] to evaluate AoI of processed updates. The SHS based approach for AoI evaluation was first introduced in [29] and has been since employed in AoI evaluation of a variety of status updating systems [6], [30]–[36]. An SHS has a state-space with two components – a discrete component $q(t) \in \mathcal{Q} = \{0, 2, \ldots, M\}$ that is a continuous-time finite-state Markov Chain and a continuous component $\mathbf{x}(t) = [x_0(t), \ldots, x_n(t)] \in \mathbb{R}^{n+1}$. In AoI analyses using SHS, each $x_j(t) \in \mathbf{x}(t)$ describes an age process of interest. Each transition $l \in \mathcal{L}$ is a directed edge (q_l, q_l') with a transition rate $\lambda^{(l)}$ in the Markov chain. The age process vector evolves at a unit rate in each discrete state $q \in \mathcal{Q}$, i.e., $\frac{d\mathbf{x}}{dt} = \dot{\mathbf{x}}(t) = \mathbf{1}_n$. A transition l causes a system to jump from discrete state q_l to q_l' and resets the continuous state from $\mathbf{x}$ to $\mathbf{x}'$ using a linear transition reset map $\mathbf{A}_l \in \{0,1\}^{(n\times n)}$ such that $\mathbf{x}' = \mathbf{x}\mathbf{A}_l$. For simple queues, examples of transition reset mappings $\{\mathbf{A}_l\}$ can be found in [29].

For a discrete state $\bar{q} \in \mathcal{Q}$, let

$$\mathcal{L}_{\bar{q}} = \{l \in \mathcal{L} : q_l' = \bar{q}\}, \qquad \mathcal{L}'_{\bar{q}} = \{l \in \mathcal{L} : q_l = \bar{q}\}. \tag{12}$$

denote the respective sets of incoming and outgoing transitions. Age analysis using SHS is based on the expected value

processes $\{\mathbf{v}_q(t)\colon q \in \mathcal{Q}\}$ such that $\mathbf{v}_q(t) = \mathrm{E}[\mathbf{x}(t)\delta_{q,q(t)}]$, with $\delta_{i,j}$ denoting the Kronecker delta function. For the SHS models of age processes considered here, each $\mathbf{v}_q(t)$ will converge to a fixed point $\bar{\mathbf{v}}_q$. The fixed points $\{\bar{\mathbf{v}}_q\colon q \in \mathcal{Q}\}$ are the solution to a set of age balance equations. The following theorem provides a simple way to calculate the age-balance fixed point and then the average age.

Theorem 1. *[29, Theorem 4] If the discrete-state Markov chain $q(t) \in \mathcal{Q} = \{0, \ldots, M\}$ is ergodic with stationary distribution $\bar{\boldsymbol{\pi}} = [\bar{\pi}_0 \cdots \bar{\pi}_M] > 0$ and there exists a non-negative vector $\bar{\mathbf{v}} = [\bar{\mathbf{v}}_0 \cdots \bar{\mathbf{v}}_M]$ such that*

$$\bar{\mathbf{v}}_{\bar{q}} \sum_{l \in \mathcal{L}_{\bar{q}}} \lambda^{(l)} = \mathbf{1}\bar{\pi}_{\bar{q}} + \sum_{l \in \mathcal{L}'_{\bar{q}}} \lambda^{(l)} \bar{\mathbf{v}}_{q_l} \mathbf{A}_l, \quad \bar{q} \in \mathcal{Q}, \tag{13}$$

then the average age vector is $\mathrm{E}[\mathbf{x}] = \lim_{t\to\infty} \mathrm{E}[\mathbf{x}(t)] = \sum_{\bar{q}\in\mathcal{Q}} \bar{\mathbf{v}}_{\bar{q}}$.

III. Problem Formulation: Servers in Series

In this section, we analyze update processing models with two servers arranged in series, as depicted in Fig. 1 for the case of $n = 2$. Server 1 (Processor 1) performs the first step of processing, while server 2 (Processor 2) handles the second step. We assume a generate-at-will with zero-wait scenario at server 1 such that it can generate a fresh (age zero) update whenever it wishes. However, we consider variations on service disciplines at server 2. There may be a single queue to save updates from server 1 when server 2 is busy. Since, the queuing (if any) is only at server 2, we name our sub-models based on the queuing discipline at server 2.

With generate-at-will with zero-wait strategy and memoryless service times, the departure process at server 1 is a Poisson process with rate μ_1. Consequently, the inter-arrival times of updates at server 2 follow an exponential distribution with parameter μ_1. The service time at server 2 is also exponential, with rate μ_2.

We adopt Kendall's notation to denote the queuing discipline at server 2, following the convention used in the AoI literature [3], [37]. For example, an M/M/1/1 submodel implies a queueing system that blocks and clears a new arrival while server 2 is busy. We use the notation M/M/1* to indicate preemption in service at server 2, and M/M/1/2* to denote a system with a waiting room having an update capacity of 1, with preemption in waiting. We now describe the analysis of various preemptive and non-preemptive system models for servers in series configuration.

A. M/M/1*

In this model, server 1 generates a fresh update immediately upon completing the processing of the previous update. The update is then passed to server 2 at a rate μ_1. Server 2 employs preemption in service, allowing a new arrival from server 1 to preempt an update currently being serviced at server 2. Consequently, an update departing from server 1 immediately enters service at server 2, and any preempted update at server 2 is discarded. Since there is no queuing at server 2, it is either idle or actively serving an update. The discrete state space of M/M/1* is $\mathcal{Q} = \{0, 1\}$ where 0 corresponds to server 2 being idle and state 1 represents busy server 2. The stationary probabilities are

$$\pi_0 = \frac{1}{1+\rho}, \quad \text{and} \quad \pi_1 = \frac{\rho}{1+\rho}. \tag{14}$$

Our M/M/1* model is analogous to the line network studied in [6], [34], where it was demonstrated that the age at the monitor for a two-server line network, applicable to our model as well, is given by

$$\Delta_{\text{M/M/1}^*}(\mu_1, \mu_2) = \frac{1}{\mu_1} + \frac{1}{\mu_2}, \tag{15}$$

Alternatively, we can express the age in terms of μ_2 and ρ as

$$\Delta_{\text{M/M/1}^*}(\mu_2, \rho) = \frac{1}{\mu_2}\left(1 + \frac{1}{\rho}\right). \tag{16}$$

Since server 1 is perpetually busy with step 1, $\overline{N}_1(\rho) = 1$. Since server 2 works on step 2 only in state 1, (14) implies

$$\overline{N}_2(\rho) = \sum_{q\in\mathcal{Q}} \pi_q(\rho) n_{q,2} = \pi_1(\rho) = \frac{\rho}{1+\rho}. \tag{17}$$

It then follows from (10) that the power-weighted processor activity $\overline{N}(\rho)$ for M/M/1* is

$$\overline{N}_{\text{M/M/1}^*}(\rho) = \rho^\alpha \overline{N}_1(\rho) + \overline{N}_2(\rho) = \rho^\alpha + \frac{\rho}{1+\rho}. \tag{18}$$

Thus, the M/M/1* speed constraint (11b) takes the form

$$\mu_2^\alpha \le \frac{P}{\mathrm{E}[C]^\alpha \overline{N}_{\text{M/M/1}^*}(\rho)}. \tag{19}$$

B. M/M/1/2*

Server 1 generates a fresh update as soon as it finishes processing the previous update. The step 1 update is then sent to the waiting room of server 2, which has a capacity of 1. In this waiting room, a new arrival from server 1 preempts any existing update. If the waiting room is empty, an arrival from server 1 goes into service at server 2. When server 2 completes processing its current update, it either sits idle if the waiting room is empty or begins processing the next update from the waiting room. The age of processed update can be analysed using the SHS Markov chain and table of state transitions depicted in Fig. 3. The continuous state age vector is $\mathbf{x} = [x_0, x_1, x_2, x_3]$, where x_0 is the age of the processed update at the monitor, x_1 and x_2 are the ages of the update at server 1 and server 2 respectively, and x_3 is the age of

l	$q_l \to q'_l$	$\lambda^{(l)}$	$\mathbf{x}\mathbf{A}_l$
1	$0 \to 1$	μ_1	$[x_0, 0, x_1, x_1]$
2	$1 \to 0$	μ_2	$[x_2, x_1, x_2, x_3]$
3	$1 \to 2$	μ_1	$[x_0, 0, x_2, x_1]$
4	$2 \to 2$	μ_1	$[x_0, 0, x_2, x_1]$
5	$2 \to 1$	μ_2	$[x_2, x_1, x_3, x_3]$

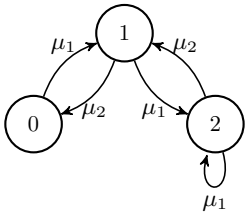


Fig. 3. The SHS transition/reset maps and Markov chain corresponding to M/M/1/2* model.

the update at server 2's waiting room. The discrete state is $\mathcal{Q} = \{0, 1, 2\}$, where state 0 corresponds to server 2 being idle, and states 1 and 2 correspond to server 2 being busy with no update in the queue and one update waiting in the queue, respectively.

We now describe SHS transitions enumerated in the table in Fig. 3.

- $l = 1$: Server 1 finishes step 1 and sends the update to idle server 2. Server 2 receives an update of age x_1, thus $x_2' = x_1$. A fresh update is generated at server 1, thus $x_1' = 0$. Age at the monitor remains unchanged, hence $x_0' = x_0$.
- $l = 2$: Server 2 finishes step 2, and delivers the update to monitor, making $x_0' = x_2$. The waiting room is empty, and server 2 waits for an update from server 1, resulting in no change in x_2.
- $l = 3, 4$: Update from server 1 arrives in the waiting room and preempts the update (if any), resetting the age in the waiting room to $x_3' = x_1$. Server 1 generates a fresh update, hence $x_1' = 0$.
- $l = 5$: Server 2 finishes step 2 and delivers update to the monitor, resulting in $x_0' = x_2$. Since there is an update waiting in server 2's buffer with age x_3, server 2 starts processing this update, thus age at server 2 is reset to the age of update in the waiting room i.e., $x_2' = x_3$.

The Markov chain in Fig. 3 has stationary probabilities $\boldsymbol{\pi}$ with normalization constant C_π given by

$$\boldsymbol{\pi} = [\pi_0 \ \pi_1 \ \pi_2] = C_\pi^{-1}[1 \ \rho \ \rho^2], \tag{20a}$$

$$C_\pi = 1 + \rho + \rho^2. \tag{20b}$$

We now use Theorem 1 to solve for $\bar{\mathbf{v}} = [\bar{\mathbf{v}}_0 \ \bar{\mathbf{v}}_1 \ \bar{\mathbf{v}}_2]$, where $\mathbf{v}_q = [v_{q0} \ v_{q1} \ v_{q2} \ v_{q3}], \forall q \in \mathcal{Q}$. With $\mu_1 = \rho\mu_2$, this yields

$$\rho\mu_2\bar{\mathbf{v}}_0 = \mathbf{1}\bar{\pi}_0 + \mu_2\bar{\mathbf{v}}_1\mathbf{A}_2, \tag{21a}$$

$$\mu_2(1+\rho)\bar{\mathbf{v}}_1 = \mathbf{1}\bar{\pi}_1 + \rho\mu_2\bar{\mathbf{v}}_0\mathbf{A}_1 + \mu_2\bar{\mathbf{v}}_2\mathbf{A}_5, \tag{21b}$$

$$\mu_2(1+\rho)\bar{\mathbf{v}}_2 = \mathbf{1}\bar{\pi}_2 + \rho\mu_2\bar{\mathbf{v}}_2\mathbf{A}_4 + \rho\mu_2\bar{\mathbf{v}}_1\mathbf{A}_3. \tag{21c}$$

The age at the monitor $\Delta_{\text{M/M/1/2}^*}$, is then calculated as $\Delta_{\text{M/M/1/2}^*} = v_{0,0} + v_{1,0} + v_{2,0}$. Some algebra yields[1]

$$\Delta_{\text{M/M/1/2}^*}(\mu_2, \rho) = \frac{1}{\mu_2}\Big(\frac{2}{\rho} + \frac{2\rho^2}{1+\rho+\rho^2} + \frac{(1+2\rho)(1+3\rho+\rho^2)}{(1+\rho)^4}\Big). \tag{22}$$

Similar to M/M/1*, server 1 is always busy with step 1. As such for each state $q \in \mathcal{Q}$, $n_{q,1} = 1$. The average number of processors working on step 1 is

$$\overline{N}_1(\rho) = \sum_{q\in\mathcal{Q}} \pi_q(\rho) n_{q,1} = 1. \tag{23}$$

[1]Note that the M/M/1/2* model here is equivalent to the end-to-end update processing model in the edge computing scenario of [38]. With arrival rate $\lambda = \mu_1$ and service rate $\mu = \mu_2$, the age expression [38, Theorem 1, Equation (9)] derived with sawtooth waveform analysis can be shown to be identical to (22).

l	$q_l \to q_l'$	$\lambda^{(l)}$	$\mathbf{x}\mathbf{A}_l$
1	$0 \to 1$	μ_1	$[x_0, 0, x_1]$
2	$1 \to 1$	μ_1	$[x_0, 0, x_2]$
3	$1 \to 0$	μ_2	$[x_2, x_1, x_2]$

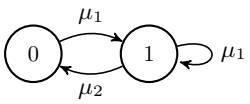


Fig. 4. The SHS transition/reset maps and Markov chain corresponding to Model $M/M/1/1$.

Only server 2 works on step 2 in states $q = 1$ and $q = 2$. No server works on step 2 in state $q = 0$. Hence,

$$\overline{N}_2(\rho) = \sum_{q\in\mathcal{Q}} \pi_q(\rho) n_{q,2} = \pi_1 + \pi_2 = \frac{\rho(1+\rho)}{1+\rho+\rho^2}. \tag{24}$$

From (10), (23) and (24), the power-weighted processor activity $\overline{N}(\rho)$ takes the form

$$\overline{N}_{\text{M/M/1/2}^*}(\rho) = \rho^\alpha\overline{N}_1(\rho) + \overline{N}_2(\rho) = \rho^\alpha + \frac{\rho(1+\rho)}{(1+\rho+\rho^2)}. \tag{25}$$

For M/M/1/2*, (11b) and (25) imply

$$\mu_2^\alpha \le \frac{P}{\mathrm{E}[C]^\alpha \overline{N}_{\text{M/M/1/2}^*}(\rho)}. \tag{26}$$

C. M/M/1/1

In this model as well, server 1 generates a fresh source update immediately after processing the previous one. Server 1 then sends updates to server 2 at rate μ_1. If server 2 is busy when a new update arrives, the new update is discarded. Consequently, Server 2 only accepts updates when it is idle.

The age at the monitor for M/M/1/1 model $\Delta_{\text{M/M/1/1}}$ can be described by the SHS Markov chain and table of state transitions shown in Fig. 4. The continuous age state vector is $\mathbf{x} = [x_0, x_1, x_2]$, where x_0 is the age of the processed update at the monitor, x_1 and x_2 are ages of the update at server 1 and server 2 respectively. For this model, discrete states are $\mathcal{Q} = \{0, 1\}$, where 0 and 1 correspond to server 2 being idle and busy respectively. The stationary probabilities are:

$$\pi_0 = \frac{1}{1+\rho}, \quad \text{and} \quad \pi_1 = \frac{\rho}{1+\rho}. \tag{27}$$

The SHS transitions are self-explanatory. Employing Theorem 1, we calculate age at the monitor as $\Delta_{\text{M/M/1/1}} = v_{0,0} + v_{1,0}$. Some algebraic manipulation gives

$$\Delta_{\text{M/M/1/1}}(\mu_2, \rho) = \frac{2}{\mu_2}\Big(1 + \frac{1}{\rho}\Big). \tag{28}$$

Server 1 is always busy with step 1, $\overline{N}_1(\rho) = 1$. Server 2 working on step 2 is only active in state $q = 1$. Using (27),

$$\overline{N}_2(\rho) = \sum_{q\in\mathcal{Q}} \pi_q(\rho) n_{q,2} = \pi_1 = \frac{\rho}{1+\rho}. \tag{29}$$

The power weighted processor activity for M/M/1/1 is

$$\overline{N}_{\text{M/M/1/1}}(\rho) = \rho^\alpha\overline{N}_1(\rho) + \overline{N}_2(\rho) = \rho^\alpha + \frac{\rho}{1+\rho} \tag{30}$$

with (11b) equivalent to

$$\mu_2^\alpha \le \frac{P}{\mathrm{E}[C]^\alpha \overline{N}_{\text{M/M/1/1}}(\rho)}. \tag{31}$$

l	$q_l \to q_l'$	$\lambda^{(l)}$	$\mathbf{x}\mathbf{A}_l$
1	$1 \to 2$	μ_1	$[x_0, x_1, x_1]$
2	$2 \to 1$	μ_2	$[x_2, 0, x_2]$

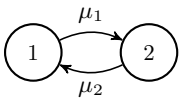


Fig. 5. The SHS transition/reset maps and Markov chain for synchronous sequential servers.

D. Synchronous Sequential Service (SSS)

In this model, servers work synchronously, meaning server 1 generates a fresh update after server 2 finishes step 2 on previous update. Consequently, processing on source update starts when both servers are idle. Here, only one server is busy at any instance. The age analysis for this model can be approached using either the sawtooth waveform method or the SHS method. For consistency with previous analyses, we apply the SHS method to evaluate the AoI at the monitor.

Fig. 5 illustrates SHS Markov Chain and table of state transitions for synchronous servers model. The continuous age state vector is $\mathbf{x} = [x_0, x_1, x_2]$, where x_0 is the age of the processed update at the monitor, and x_1 and x_2 are the ages of the update at server 1 and server 2 respectively. For this model, discrete states are $\mathcal{Q} = \{1, 2\}$, where 1 and 2 correspond to server 1 and server 2 being busy respectively. We skip explaining the SHS transitions due to space constraints, however we do note that unlike in M/M/1/2* and M/M/1/1 models, $x_1' = 0$ occurs at the transition corresponding to μ_2. With $\rho = \mu_1/\mu_2$, the Markov Chain in Fig. 5 has stationary probabilities

$$\pi_1(\rho) = \frac{1}{1+\rho}, \quad \text{and} \quad \pi_2(\rho) = \frac{\rho}{1+\rho}. \tag{32}$$

The age at the monitor, $\Delta_{\text{sync}}(\mu_1, \mu_2)$, is calculated as $v_{10} + v_{20}$, resulting in

$$\Delta_{\text{SSS}}(\mu_2, \rho) = \frac{1}{\mu_2}\Big(2 + \frac{1}{\rho} + \frac{1}{\rho(1+\rho)}\Big). \tag{33}$$

Server 1 works on step 1 only in state 1 and consequently from (32)

$$\overline{N}_1(\rho) = \sum_{q\in\mathcal{Q}} \pi_q(\rho) n_{q,1} = \pi_1 = \frac{1}{1+\rho}. \tag{34}$$

Similarly, server 2 works on step 2 in state $q = 2$. Thus,

$$\overline{N}_2(\rho) = \sum_{q\in\mathcal{Q}} \pi_q(\rho) n_{q,2} = \pi_2 = \frac{\rho}{1+\rho}. \tag{35}$$

It follows from (10), (34) and (35) that

$$\overline{N}_{\text{SSS}}(\rho) = \rho^\alpha \overline{N}_1(\rho) + \overline{N}_2(\rho) = \frac{\rho^\alpha + \rho}{1+\rho}. \tag{36}$$

For SSS, (11b) and (36) imply

$$\mu_2^\alpha \le \frac{P}{\mathrm{E}[C]^\alpha \overline{N}_{\text{SSS}}(\rho)}. \tag{37}$$

IV. Problem Formulation: Parallel Servers

In this section, we identify two-step update processing models involving two parallel servers, as illustrated in Fig. 2 for the case of $n = 2$. Unlike the series server setup, here each server is allowed to execute both computation steps. For each model, we identify the system state set $\mathcal{Q}$, the stationary probabilities π_q, age $\Delta(\mu_2, \rho)$, $\overline{N}(\rho)$ and consequently, the upper bound on step 2 service rate μ_2.

We observe that in parallel server systems, the quantity $\rho = \mu_1/\mu_2$ does not always accurately represent the total offered load from step 1 to step 2. This is because one server may transition from step 1 to step 2 while the other remains in step 1. Moreover, as we will see in certain parallel server policies, even within a single server, the concept of offered load from steps 1 to step 2 can break down, diverging from its conventional interpretation in general queueing theory. Therefore, we refrain from referring to ρ as the offered load in parallel setup. Instead, we treat ρ purely as the ratio between the service rates of step 1 and step 2.

A. Parallel SSS (P-SSS)

In this mode, two identical servers process updates independently, in parallel. Each server works on a distinct update and executes both computation steps. After processing an update, a server generates new update and immediately starts processing this update. The total service time for an update is a random variable $T = T_1 + T_2 = C_1/f_1 + C_2/f_2$. With $T_i \sim \exp(\mu_i)$, the service time T is a two-parameter hypoexponential distribution with parameters μ_1 and μ_2.

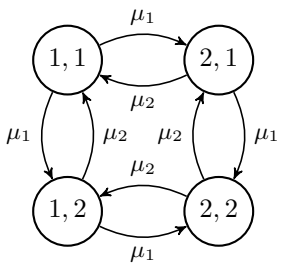
The age analysis for parallel server setup is non-trivial even with two parallel servers. Unlike the series server setup, not every update delivered by the servers resets the monitor's age i.e. not every update delivery is *useful*. The issue stems from variability in processing times across servers. For instance, one server might take longer to process an update, while the other server, working on a fresher update, finishes its task earlier and resets the monitor's age. Consequently, when the older update from the first server eventually arrives, it fails to reset the monitor's age.

When server $i, i \in \{1, 2\}$, sends an update with age x_i to the monitor, the monitor accepts a processed update only if it is fresher than its current update. Consequently, the resulting age at the monitor, denoted by x_0, is updated as

$$x_0' = \min(x_0, x_i). \tag{38}$$

Therefore, in the SHS-based age analysis, it is essential to track variables such as $\min(x_0, x_1)$, $\min(x_0, x_2)$, and $\min(x_0, x_1, x_2)$ to accurately account for the acceptance of fresh updates and discarding of outdated ones. This approach to tracking age variables is inspired by the methodology proposed in [39]. We now proceed to describe the SHS analysis for the P-SSS model in detail.

The age of processed update for Parallel SSS model can be analyzed using the SHS Markov chain and table of state transitions shown in Fig. 6. The discrete state set is $\mathcal{Q} =$



(a)

l	$q_l \to q_l'$	$\lambda^{(l)}$	$\mathbf{x}\mathbf{A}_l$
1	$(1,1) \to (2,1)$	μ_1	$[x_0, x_1, x_2, x_3, x_4, x_5]$
2	$(1,1) \to (1,2)$	μ_1	$[x_0, x_1, x_2, x_3, x_4, x_5]$
3	$(1,2) \to (2,2)$	μ_1	$[x_0, x_1, x_2, x_3, x_4, x_5]$
4	$(2,1) \to (2,2)$	μ_1	$[x_0, x_1, x_2, x_3, x_4, x_5]$
5	$(2,1) \to (1,1)$	μ_2	$[x_3, 0, x_2, 0, x_5, 0]$
6	$(2,2) \to (1,2)$	μ_2	$[x_3, 0, x_2, 0, x_5, 0]$
7	$(1,2) \to (1,1)$	μ_2	$[x_4, x_1, 0, x_5, 0, 0]$
8	$(2,2) \to (2,1)$	μ_2	$[x_4, x_1, 0, x_5, 0, 0]$

(b)

Fig. 6. (a) Markov Chain, and (b) SHS transition maps corresponding to Parallel Sequential Synchronous Service (P-SSS) system.

$\{(1,1),(1,2),(2,1),(2,2)\}$, where each tuple $(i,j) \in \mathcal{Q}$ represents the current step of server 1 and server 2, respectively. The continuous state age vector is $\mathbf{x} = [x_0, x_1, x_2, x_3, x_4, x_5]$, where x_0 is the age at monitor, x_1 and x_2 are the ages of the update at server 1 and server 2 respectively, $x_3 = \min(x_0, x_1)$, $x_4 = \min(x_0, x_2)$, and $x_5 = \min(x_0, x_1, x_2)$.

The SHS transitions are enumerated in the table in Fig. 6 and can be understood as follows:

- $l = 1, 2, 3, 4$: In these transitions, the servers only change steps; one of the servers finishes step 1 and begins step 2. Consequently, there is no reset in the age of updates at servers 1 and 2. Since no update is delivered to the monitor, the age at the monitor remains unchanged. As a result, the age variables x_3, x_4, and x_5 also remain unchanged.
- $l = 5, 6$: These transitions occur when server 1 finishes processing and delivers the update to the monitor. Server 1 generates a fresh update, consequently, $x_1' = 0$. Since server 2 continues to work on its update, $x_2' = x_2$. The age at the monitor is reset according to $\min(x_0, x_1)$, which is tracked by age variable x_3. Hence, $x_0' = x_3$. Since $x_1' = 0$, thus $x_3' = 0$. The transition for age variable x_4 is more complex. We have $x_4' = \min(x_0', x_2') = \min(x_3, x_2) = \min(\min(x_0, x_1), x_2) = \min(x_0, x_1, x_2) = x_5$. Further, $x_5' = \min(x_0', x_1', x_2') = \min(x_3, 0, x_2) = 0$.
- $l = 7, 8$: These transitions occur when server 2 finishes processing and delivers the update to the monitor. Now server 2 generates a new update, thus $x_2' = 0$. Since server 1 continues to work on its update, $x_1' = x_1$. The age at the monitor is reset according to $\min(x_0, x_2)$, which is tracked by age variable x_4. Hence, $x_0' = x_4$. Since $x_2' = 0$, $x_4' = 0$. Next, we have $x_3' = \min(x_0', x_1') = \min(x_4, x_1) = \min(\min(x_0, x_2), x_1) = \min(x_0, x_1, x_2) = x_5$. Additionally, $x_5' = \min(x_0', x_1', x_2') = \min(x_4, x_1, 0) = 0$.

The Markov chain in Fig. 6 has stationary probabilities $\boldsymbol{\pi}$

$$\boldsymbol{\pi} = [\pi_{(1,1)}, \pi_{(1,2)}, \pi_{(2,1)}, \pi_{(2,2)}] = \frac{1}{(1+\rho)^2}[1, \rho, \rho, \rho^2]. \quad (39)$$

The age balance equations are

$$2\rho\mu_2 \bar{\mathbf{v}}_{1,1} = \mathbf{1}\bar{\pi}_{1,1} + \mu_2 \bar{\mathbf{v}}_{2,1}\mathbf{A}_5 + \mu_2 \bar{\mathbf{v}}_{1,2}\mathbf{A}_7, \quad (40a)$$

$$(1+\rho)\mu_2 \bar{\mathbf{v}}_{2,1} = \mathbf{1}\bar{\pi}_{2,1} + \rho\mu_2 \bar{\mathbf{v}}_{1,1}\mathbf{A}_1 + \mu_2 \bar{\mathbf{v}}_{2,2}\mathbf{A}_8, \quad (40b)$$

$$(1+\rho)\mu_2 \bar{\mathbf{v}}_{1,2} = \mathbf{1}\bar{\pi}_{1,2} + \rho\mu_2 \bar{\mathbf{v}}_{1,1}\mathbf{A}_2 + \mu_2 \bar{\mathbf{v}}_{2,2}\mathbf{A}_6, \quad (40c)$$

$$2\mu_2 \bar{\mathbf{v}}_{2,2} = \mathbf{1}\bar{\pi}_{2,2} + \rho\mu_2 \bar{\mathbf{v}}_{1,2}\mathbf{A}_4 + \rho\mu_2 \bar{\mathbf{v}}_{1,2}\mathbf{A}_3. \quad (40d)$$

It follows that the age at the monitor can be calculated as $\mathrm{E}[x_3] = v_{(1,1),0} + v_{(2,1),0} + v_{(1,2),0} + v_{(2,2),0}$. Thus,

$$\Delta_{\text{P-SSS}}(\mu_2, \rho) = \frac{1}{\mu_2}\left(1 + \frac{1}{\rho} + \frac{1+\rho+\rho^2}{4\rho(1+\rho)} + \frac{\rho(1+2\rho)(2+\rho)}{4(1+\rho)^5}\right). \quad (41)$$

Further, observe that as $\rho \to \infty$ $\Delta_{\text{P-SSS}} \to 1.25/\mu_2$. This result aligns with the following intuition: When $\rho \to \infty$ and μ_2 is finite , step 1 is almost instantly completed, and each server delivers an update with an average age $1/\mu_2$. If there were only one server, this would correspond to an average age of $2/\mu_2$ at the monitor, since each update is delivered on average after a duration of $1/\mu_2$. However, if a single server were running step 2 at twice the speed ($2\mu_2$), the age at the monitor would be $1/\mu_2$ as $\rho \to \infty$. However, in the P-SSS setup, we are not running one server at double speed but rather operating two parallel servers. Each server processes an update that is slightly older, resulting in an average age of $1.25/\mu_2$ rather than $1/\mu_2$.

Next, we observe that in states $(1,2)$ and $(2,1)$, one server works on step 1 while the other works on step 2. In state $(1,1)$, both servers work on step 1, and in state $(2,2)$, no servers work on step 1, as both are executing step 2. Using (39), the average number of processors working on step 1 is given by

$$\overline{N}_1(\rho) = 2\pi_{(1,1)} + \pi_{(1,2)} + \pi_{(2,1)} = \frac{2}{1+\rho}. \quad (42)$$

Similarly, the average number of processors working on step 2 is

$$\overline{N}_2(\rho) = \pi_{(1,2)} + \pi_{(2,1)} + 2\pi_{(2,2)} = \frac{2\rho}{1+\rho}. \quad (43)$$

Hence, $\overline{N}(\rho)$ for P-SSS is

$$\overline{N}_{\text{P-SSS}}(\rho) = \rho^\alpha \overline{N}_1(\rho) + \overline{N}_2(\rho) = \frac{2(\rho^\alpha + \rho)}{1+\rho}. \quad (44)$$

With $\overline{N}_{\text{P-SSS}}(\rho)$ defined in (44), the upper bound on the service rate is then

$$\mu_2^\alpha \leq \frac{P}{\mathrm{E}[C]^\alpha \overline{N}_{\text{P-SSS}}(\rho)}. \quad (45)$$

l	$q_l \to q_l'$	$\lambda^{(l)}$	$\mathbf{x}\mathbf{A}_l$
1	$1 \to 2$	$2\mu_1$	$[x_0, 0, x_1]$
2	$2 \to 2$	μ_1	$[x_0, 0, x_1]$
3	$2 \to 1$	μ_2	$[x_2, 0, x_2]$

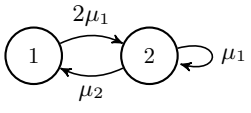


Fig. 7. The SHS transition/reset maps and Markov chain corresponding to Parallel Coordinated Alternating Freshness (P-CAF) policy for parallel servers.

l	$q_l \to q_l'$	$\lambda^{(l)}$	$\mathbf{x}\mathbf{A}_l$
1	$1 \to 2$	$2\mu_1$	$[x_0, x_1]$
2	$2 \to 1$	$2\mu_2$	$[x_1, 0]$

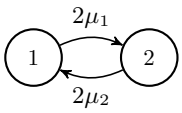


Fig. 8. The SHS transition/reset maps and Markov chain corresponding to Parallel Shared Intermediate Update (P-SIU) policy for parallel servers.

B. Parallel Coordinated Alternating Freshness (P-CAF)

When both servers i and j are in step 1, they process the same fresh update concurrently. If server i transitions to step 2, then server j restarts step 1 with a fresh update. If server j reaches step 2 with its fresher update before server i completes its processing, then server i will abort its current task and restart in step 1 with a fresh update. In this policy, the update in step 1 is always the freshest and only one server is allowed to work on step 2 of update processing at a time.

The Markov state space is defined as $\mathcal{Q} = \{1, 2\}$, where state 1 corresponds to both servers working step 1, while state 2 indicates that one server is in step 2 and the other in step 1. The continuous age vector is $\mathbf{x} = [x_0, x_1, x_2]$, where x_0 corresponds to age at the monitor, and x_1 denotes the age of update currently in step 1, and x_2 denotes the age of update being processed in step 2.

Note that, in contrast to the P-SSS model where both servers work independently and may deliver stale updates, the P-CAF policy allows coordination among the servers to ensure that only the freshest update that has completed two steps of processing is delivered. Mathematically, the analysis is simplified since we don't need to track age variables such as $\min(x_0, x_i)$ as described in (38).

The SHS Markov chain and table of state transitions are shown in Fig. 7.

- $l = 1$: Transition from state 1 to state 2 at rate $2\mu_1$. In state 1, both servers are in step 1. The time until one server finishes step 1 is the minimum of two independent exponential distributions with rate μ_1, resulting in a departure rate of $2\mu_1$ from state 1. Upon transition, one server moves to step 2, so $x_2' = x_1$, while the other server restarts in step 1 with a fresh update, thus $x_1' = 0$. The age at the monitor, x_0, remains unchanged since no update is delivered, so $x_0' = x_0$.
- $l = 2$: Server in step 1 finishes service to reach step 2, and the server in step 2 (with an older update) restarts in step 1 with a fresh update. The age of the update in step 1 is reset to 0, so $x_1' = 0$. The age of the update now being processed in step 2 is updated to the age of the previous update in step 1, so $x_2' = x_1$. The age at the monitor, x_0, remains unchanged, so $x_0' = x_0$.
- $l = 3$: Server in step 2 finishes service and delivers the processed update to the monitor. The age at the monitor is updated to the age of the update that was in step 2, so $x_0' = x_2$. The server that finished in step 2 and the server that was in step 1 restart with a fresh update, thus $x_1' = 0$.

The Markov Chain in Fig. 7 has stationary probabilities

$$\pi_1 = \frac{1}{1+\rho}, \quad \text{and} \quad \pi_2 = \frac{\rho}{1+\rho}. \tag{46}$$

The age balance equations are

$$2\rho\mu_2[v_{10}\ v_{11}\ v_{12}] = [\pi_1\ \pi_1\ \pi_1] + \mu_2[v_{22}\ 0\ v_{22}], \tag{47a}$$

$$(1+\rho)\mu_2[v_{20}\ v_{21}\ v_{22}] = [\pi_2\ \pi_2\ \pi_2] + \rho\mu_2[v_{20}\ 0\ v_{21}] + 2\rho\mu_2[v_{10}\ 0\ v_{11}]. \tag{47b}$$

Solving the set of equations in (47), we can obtain v_{qj}. The age at the monitor is $\Delta_{\text{P-CAF}} = v_{10} + v_{20}$, which yields

$$\Delta_{\text{P-CAF}} = \frac{1}{\mu_2}\Big(\frac{3}{2(1+\rho)} + \frac{2\rho}{1+2\rho} + \frac{1+\rho+\rho^2}{\rho(1+\rho)^2}\Big). \tag{48}$$

In the P-CAF policy, in state $q = 1$, both servers execute step 1, while in state $q = 2$, only one server works on step 1. Using π_q from (46), the average number of servers executing step 1 is

$$\overline{N}_1(\rho) = \sum_{q\in\mathcal{Q}} \pi_q(\rho) n_{q,1} = 2\pi_1 + \pi_2 = \frac{2+\rho}{1+\rho}. \tag{49}$$

Since there is only one state ($q = 2$) where a server is executing step 2, we have

$$\overline{N}_2(\rho) = \sum_{q\in\mathcal{Q}} \pi_q(\rho) n_{q,2} = \pi_2 = \frac{\rho}{1+\rho}. \tag{50}$$

It follows from (10), (49) and (50) that

$$\overline{N}_{\text{P-CAF}}(\rho) = \rho^\alpha \overline{N}_1(\rho) + \overline{N}_2(\rho) = \frac{\rho^\alpha(2+\rho)}{1+\rho} + \frac{\rho}{1+\rho}. \tag{51}$$

The upper bound on μ_2 is then given by

$$\mu_2^\alpha \le \frac{P}{\mathrm{E}[C]^\alpha \overline{N}_{\text{P-CAF}}(\rho)}. \tag{52}$$

C. Parallel Shared Intermediate Updates (P-SIU)

This policy leverages server-to-server communication to share intermediate processing results, enabling parallel execution of each step. Under this policy, both servers initially work on step 1 of the same update. Once one server completes step 1, it shares the intermediate result with the other server. Subsequently, both servers begin step 2 processing on this intermediate update simultaneously. When either server completes step 2, both servers reset and start processing a fresh update.

The P-SIU policy effectively transforms the parallel server system into a system that behaves like a single server. In this equivalent single-server system, the service times for step i are

exponential with rate $2\mu_i$. This simplification arises because the policy ensures that both servers are always working in parallel on the same update, whether in step 1 or step 2.

The discrete state space of the system is defined as $\mathcal{Q} = \{1, 2\}$, where state 1 corresponds to both servers in step 1, and state 2 correspond to both servers in step 2. The continuous age vector is $\mathbf{x} = [x_0, x_1]$, where x_0 is the age of the update at the monitor, and x_1 is the age of the update being processed by the servers. The SHS Markov Chain and transitions are illustrated in Fig. 8 and are self-explanatory. Additionally, the Markov Chain in Fig. 8 has stationary probabilities

$$\pi_1 = \frac{1}{1+\rho}, \quad \text{and} \quad \pi_2 = \frac{\rho}{1+\rho}. \tag{53}$$

The age at the monitor is $\Delta_{\text{P-SIU}} = v_{10} + v_{20}$, which is expressed as

$$\Delta_{\text{P-SIU}} = \frac{1}{\mu_2}\left[1 + \frac{1}{2\rho}\left(1 + \frac{1}{1+\rho}\right)\right]. \tag{54}$$

In state 1, both servers execute step 1, while in state 2, both servers execute step 2. The average number of servers executing step 1 and step 2 is then

$$\overline{N}_1(\rho) = \sum_{q\in\mathcal{Q}} \pi_q(\rho) n_{q,1} = 2\pi_1 = \frac{2}{1+\rho}, \tag{55}$$

and

$$\overline{N}_2(\rho) = \sum_{q\in\mathcal{Q}} \pi_q(\rho) n_{q,2} = 2\pi_2 = \frac{2\rho}{1+\rho}, \tag{56}$$

From (10), (55) and (56), we derive the $\overline{N}(\rho)$ for P-SIU as

$$\overline{N}_{\text{P-SIU}}(\rho) = \rho^\alpha \overline{N}_1(\rho) + \overline{N}_2(\rho) = \frac{2(\rho^\alpha + \rho)}{1+\rho}, \tag{57}$$

which gives us the upper bound

$$\mu_2 \le \frac{P}{\text{E}[C]^\alpha \overline{N}_{\text{P-SIU}}(\rho)}. \tag{58}$$

V. Numerical Evaluation

In this section, we address the optimization problem presented in (11) for systems identified in Section III and Section IV. A key observation here is that all the systems we examine end up having a formulation where the age metric is proportional to $1/\mu_2$ times some factor in terms of ρ. This observation is consistent across the models, as reflected in (16), (22), (28), (33), (41), (48), and (54). Thus, in the context of optimization problem (11), age in all the models is minimized by maximizing μ_2 within the limits imposed by the right side of power constraint (11b). Specifically, for a given power budget P, expected CPU cycles required for each step $\text{E}[C]$, and scaling parameter α, denote

$$P_2(\rho) = \frac{P}{\text{E}[C]^\alpha \overline{N}(\rho)}. \tag{59}$$

Then it follows from (11b) and (59) that the optimal service rate for step 2 is

$$\mu_2^* = (P_2(\rho))^{1/\alpha}. \tag{60}$$

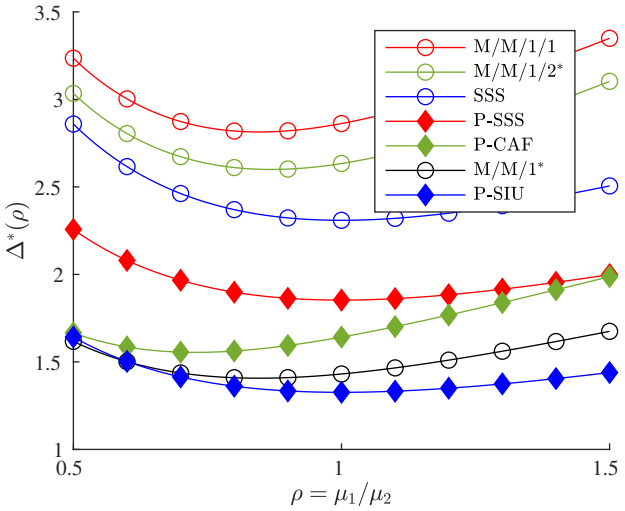


Fig. 9. Plot of objective function $\Delta^*(\rho)$ of unconstrained optimization as a function of ρ. Here $P = 8, \text{E}[C] = 1$, and $\alpha = 5$. The marker ○ represents servers in series, while ◆ represents parallel servers.

Using (60), the constrained optimization problem (11) can be reformulated as an unconstrained optimization problem with the objective function:

$$\Delta^*(\rho) = \Delta(\mu_2^*, \rho). \tag{61}$$

To illustrate the methodology, we begin with a detailed analysis of the M/M/1* system. Further, to avoid redundancy, we do not present explicit analyses for each model, and instead present the results of the numerical evaluation, which have been derived using the same methodology. To solve the M/M/1*, we use (16), (18) and (19). With $\alpha = 5$, the optimization problem in (11) can then be reformulated as:

$$\text{minimize} \quad \frac{1}{\mu_2}\left(1 + \frac{1}{\rho}\right) \tag{62a}$$

$$\text{subject to} \quad \mu_2^5 \le \frac{P}{\text{E}[C]^5\left(\rho^5 + \frac{\rho}{1+\rho}\right)}, \tag{62b}$$

$$\mu_2 \ge 0, \text{and } \rho \ge 0. \tag{62c}$$

We aim to solve (62) with respect to the variable ρ. Substituting the μ_2 upper bound (62b) for μ_2 in the objective function (62a), our goal is then to minimize

$$\Delta^*_{\text{M/M/1*}}(\rho) = \frac{\text{E}[C]}{P^{1/5}}\left(\rho^5 + \frac{\rho}{1+\rho}\right)^{1/5}\left(1 + \frac{1}{\rho}\right). \tag{63}$$

Setting $d\Delta^*_{\text{M/M/1*}}(\rho)/d\rho = 0$, we obtain $\rho^5(1+\rho) = 4/5$, yielding the optimal $\rho^* = 0.846$. Now $\rho^* < 1$, implies that server 2 should operate faster than server 1, as a slow server 2 would become a bottleneck in update processing.

Figure 9 illustrates the objective function $\Delta^*(\rho)$ as a function of ρ for each update processing model. We first observe, for all systems, that the optimal $\rho^* \le 1$, indicating that step 2 should be processed faster than step 1. For instance, faster step 2 processing in M/M/1/2* suggests that the queue at server 2 is cleared quickly, which is favorable

for minimizing the age. The optimal ρ^* for M/M/1/1 is the same as that for M/M/1* because, as shown in (18) and (30), $\overline{N}_{\text{M/M/1}^*}(\rho) = \overline{N}_{\text{M/M/1/1}}(\rho)$. Additionally, the age $\Delta_{\text{M/M/1/1}}(\mu_2, \rho) = 2\Delta_{\text{M/M/1}^*}(\mu_2, \rho)$, as evident from (28) and (16).

For the SSS, P-SSS and P-SIU models, $\rho^* = 1$ indicates that step 1 and step 2 processing should occur at the same rate. This is intuitive due to symmetry: if step 1 is slower, it delays step 2, while if step 2 is slower, the system waits longer to generate a new update. Both scenarios are suboptimal for minimizing the age.

We observe that, across all considered models, the optimal ρ^* is independent of the power constraint P. The illustrative example of M/M/1* mathematically justifies this, as minimizing the objective function (63) will be independent of P. A more conceptual reasoning is as as follows. In this work, we have considered a restrictive class of systems where increasing μ_2 and $\mu_1 = \rho\mu_2$ improves age performance. In the examined systems, when ρ is fixed, then increasing the service rate at server 2 is always age reducing as is evident from (16), (22), (28) (33), (41), (48) and (54). Therefore, the optimal μ_2 should be as large as possible while ensuring that the energy consumed by servers 1 and 2 satisfies the power constraint .

We note that this independence of ρ^* from P might not hold for all systems. For instance, consider a system where server 1 generates at will with zero wait and serves at rate μ_1, and updates are queued at server 2. The performance of server 2 is known if the updates arrive fresh [40]. However, in our case, updates arrive with some age from server 1. A longer inter-arrival time between updates can slightly empty the queue at server 2, but the updates arrive with higher age, as the inter-arrival times reflect the age of the updates. Hence, it is not straightforward to say that increasing μ_2 and $\mu_1 = \rho\mu_2$ will always minimize age, and as such there could be some optimal service rates ratio ρ^* which could depend on the power budget P.

Fig. 10 numerically compares optimal age performance $\Delta^* = \Delta(\mu_2^*, \rho^*)$ of all the models as a function of power constraint P. As expected, increasing P leads to a larger optimal μ_2^* resulting in a decrease in age due to the faster service rate. It is apparent from Fig. 10 that preemption in service yields better age performance among all servers in series models, which aligns with the existing view in the AoI literature that preemption of old updates by new ones is generally beneficial. An interesting and somewhat surprising finding is that synchronous service at servers performs better than asynchronous service, indicating that having a single update in service is more advantageous than having multiple updates in progress. This observation makes sense upon further reflection: synchronous servers prevent updates from lingering in the waiting queue at server 2 (as in the M/M/1/2* model), or causing server 2 to be idle more frequently which occurs when updates are frequently discarded (as in the M/M/1/1 model).

Fig. 10 demonstrates that the P-SSS system achieves better age performance compared to the SSS system, highlighting the advantages of parallel processing over serial processing.

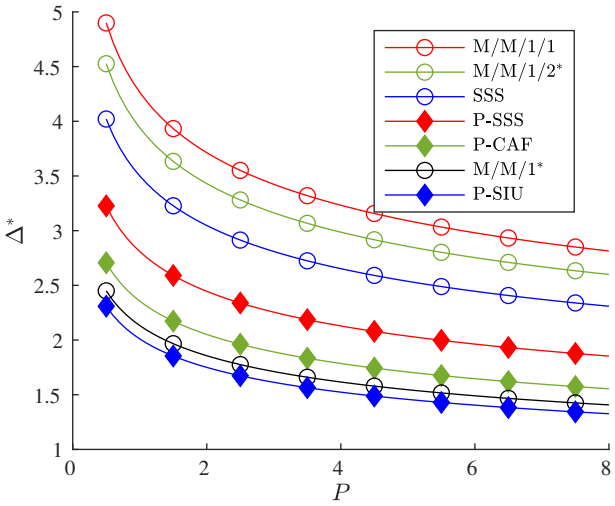


Fig. 10. Optimal age $\Delta^* = \Delta(\mu_2^*, \rho^*)$ for servers in series and parallel setups under power constraint P. Here, $\alpha = 5$ and, $\mathrm{E}[C] = 1$. The marker ○ represents servers in series, while ◆ represents parallel servers.

P-SSS can be viewed as a parallel version of SSS, where independent servers operate in a manner similar to SSS but with each server consuming power $P/2$, half of the total power budget. Additionally, the superior performance of P-CAF and P-SIU compared to P-SSS further underscores the benefits of incorporating additional information about the state of the other server.

VI. Open Problems: A discussion

An important and seemingly simple question emerging from this work is: *When is power wasted?* In parallel systems, useless work always wastes power. Specifically, if an update being processed is older than the age at the monitor, the work is considered useless. If the server continues this useless work, its efforts are deemed wasted. The P-SSS setup demonstrates an example of wasted effort. In this setup, the two servers work independently on processing updates, without knowledge of the other's progress. If a server working on an older update had information that the other server had already delivered a fresher update, it would refrain from continuing its work, recognizing that its efforts are redundant.

In contrast, the P-SIU and P-CAF policies appear to avoid any wasted power. Both policies involve sharing information about the state of the other server, ensuring that no useless work is being done. In policies like P-CAF, a server may abandon processing its current update to restart with a new one because the other server was the first to reach step 2. While this update discarding might initially seem wasteful, we argue that it is not wasted since the expected age will be less at the monitor.

Power can also be wasted even if the work doesn't meet the definition of useless. In series server setups, the effort of server 1 is wasted if server 2 continues processing an older update despite the availability of a fresher update from server 1. This wasted power is evident in the M/M/1/1 and M/M/1/2* models.

In M/M/1/1, if server 2 is busy when server 1 completes processing, the fresher update from server 1 is discarded, wasting its effort while server 2 continues with its current update. In M/M/1/2*, server 1 may deliver an update that gets queued at server 2 but is later preempted and discarded by a newer arrival from server 1, as server 2 chooses to continue processing an older update.

In the M/M/1* model, wasted effort is avoided as server 2 always prioritizes the update from server 1. If server 2 is busy when a new update arrives, it preempts its current task to process the fresher update and such preemption is not considered a waste of server 2's effort as it contributes to overall age reduction.

However, we acknowledge that the definition of wasted power becomes less clear in systems with non-exponential service times. For instance, if service times follow a uniform distribution over the interval $[a, b]$, the hazard function $h(t) = 1/(b - t)$ increases as t approaches b, reflecting that the likelihood of update finishing service becomes increasingly certain as time nears the upper bound b. Preempting or discarding the update at this stage would not be prudent, as the update is nearly complete and could reset the monitor's age. In such cases, discarding the update would indeed seem like a waste of effort.

On the other hand, there are significant opportunities for age optimization. Our analysis assumes a generate-at-will scenario with zero-wait at servers. In the existing literature on optimal waiting strategies [41], [42], it is typically assumed that there is just a single update in the service facility. Upon delivery of this update, the decision to wait is then considered. However, in our system, we allow multiple updates to be in process simultaneously. Consequently, the optimality of the known non-zero wait strategies, such as setting a threshold based on prior service time, is unresolved.

Another approach to age optimization is studying an online policy where service rates are dynamically adjusted based on the system's state. For example in P-CAF system, when one server transitions to step 2, should the other server, now working on a fresher update in step 1, increase its service rate? Alternatively, if the age at the monitor exceeds a certain threshold, the servers could speed up their processing (i.e., age-based service rates). Such adaptive strategies could be studied using a Markov Decision Process (MDP) framework.

Moreover, this work has a natural extension where each processing step has a general service time distribution. The current SHS methodology has a limitation of being applicable to systems with memoryless regimes. Developing a novel SHS analysis for a general service time will not only be useful to this work, but in general to the AoI community.

VII. Conclusion

This work explored the timely processing of updates that require a sequence of computational steps. We specifically identified various parallel and series server models for update processing, with a focus on understanding the age-power trade-off in the special case of two-step update processing. To achieve this, we formulated and solved optimization problems that determine the optimal service rates for each step, constrained by a total power budget, to minimize the average age. The analysis revealed that step 2 should generally be faster than step 1 for optimal power efficiency and minimum age. We also observed that processing by parallel servers has better age performance than servers in series.

References

[1] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[2] T. Gandhi and M. M. Trivedi, "Pedestrian protection systems: Issues, survey, and challenges," *IEEE Transactions on intelligent Transportation systems*, vol. 8, no. 3, pp. 413–430, 2007.

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

[4] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Age-optimal information updates in multihop networks," in *Proc. IEEE Int'l. Symp. Info. Theory (ISIT)*, 6 2017, pp. 576–580.

[5] R. Talak, S. Karaman, and E. Modiano, "Minimizing age-of-information in multi-hop wireless networks," in *55th Annual Allerton Conference on Communication, Control, and Computing*, 10 2017, pp. 486–493.

[6] R. D. Yates, "Age of Information in a Network of Preemptive Servers," in *IEEE Conference on Computer Communications (INFOCOM) Workshops*, Apr. 2018, pp. 118–123, arXiv preprint arXiv:1803.07993.

[7] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Modeling the age of information in emulated ad hoc networks," in *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, 10 2017, pp. 436–441.

[8] F. Chiariotti, O. Vikhrova, B. Soret, and P. Popovski, "Peak Age of Information Distribution for Edge Computing With Wireless Links," *IEEE Transactions on Communications*, vol. 69, no. 5, pp. 3176–3191, 2021.

[9] A. Sinha, S. Singhvi, P. D. Mankar, and H. S. Dhillon, "Peak Age of Information under Tandem of Queues," 2024. [Online]. Available: https://arxiv.org/abs/2405.02705

[10] O. Vikhrova, F. Chiariotti, B. Soret, G. Araniti, A. Molinaro, and P. Popovski, "Age of Information in Multi-hop Networks with Priorities," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[11] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides, "Effect of Message Transmission Path Diversity on Status Age," *IEEE Trans. Info. Theory*, vol. 62, no. 3, pp. 1360–1374, Mar. 2016.

[12] R. Talak and E. H. Modiano, "Age-Delay Tradeoffs in Queueing Systems," *IEEE Transactions on Information Theory*, vol. 67, no. 3, pp. 1743–1758, 2021.

[13] M. Fidler, J. P. Champati, J. Widmer, and M. Noroozi, "Statistical Age-of-Information Bounds for Parallel Systems: When Do Independent Channels Make a Difference?" *IEEE Journal on Selected Areas in Information Theory*, vol. 4, pp. 591–606, 2023.

[14] R. D. Yates, "Status updates through networks of parallel servers," in *Proc. IEEE Int'l. Symp. Info. Theory (ISIT)*, Jun. 2018, pp. 2281–2285.

[15] J. M. George and J. M. Harrison, "Dynamic control of a queue with adjustable service rate," *Operations research*, vol. 49, no. 5, pp. 720–731, 2001.

[16] T. B. Crabill, "Optimal control of a maintenance system with variable service rates," *Operations Research*, vol. 22, no. 4, pp. 736–745, 1974.

[17] S. Stidham, "Optimal control of admission to a queueing system," *IEEE Transactions on Automatic Control*, vol. 30, no. 8, pp. 705–713, 1985.

[18] M. Hofri and K. W. Ross, "On the Optimal Control of Two Queues with Server Setup Times and Its Analysis," *SIAM Journal on Computing*, vol. 16, no. 2, pp. 399–420, 1987. [Online]. Available: https://doi.org/10.1137/0216029

[19] N. Lee and V. G. Kulkarni, "Optimal arrival rate and service rate control of multi-server queues," *Queueing Systems*, vol. 76, pp. 37–50, 2014.

[20] R. R. Weber and S. Stidham, "Optimal control of service rates in networks of queues," *Advances in applied probability*, vol. 19, no. 1, pp. 202–218, 1987.

[21] Z. Rosberg, P. Varaiya, and J. Walrand, "Optimal control of service in tandem queues," *IEEE Transactions on Automatic Control*, vol. 27, no. 3, pp. 600–610, 1982.

[22] L. Xia, D. Miller, Z. Zhou, and N. Bambos, "Service rate control of tandem queues with power constraints," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5111–5123, 2017.

[23] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, 1992.

[24] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, "Interconnect-power dissipation in a microprocessor," in *Proceedings of the 2004 International Workshop on System Level Interconnect Prediction*, ser. SLIP '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 7–13. [Online]. Available: https://doi.org/10.1145/966747.966750

[25] T. Sakurai and A. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 2, pp. 584–594, 1990.

[26] R. Gonzalez, B. Gordon, and M. Horowitz, "Supply and threshold voltage scaling for low power CMOS," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 8, pp. 1210–1216, 1997.

[27] J. Dean and L. A. Barroso, "The Tail at Scale," *Communications of the ACM*, vol. 56, pp. 74–80, 2013. [Online]. Available: http://cacm.acm.org/magazines/2013/2/160173-the-tail-at-scale/fulltext

[28] J. P. Hespanha, "Modelling and analysis of stochastic hybrid systems," *IEE Proceedings-Control Theory and Applications*, vol. 153, no. 5, pp. 520–535, 2006.

[29] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Transactions on Information Theory*, vol. 65, no. 3, pp. 1807–1827, 2018.

[30] S. Farazi, A. G. Klein, and D. R. Brown, "Average age of information for status update systems with an energy harvesting server," in *IEEE Conference on Computer Communications (INFOCOM) Workshops*, Apr. 2018, pp. 112–117.

[31] A. Maatouk, M. Assaad, and A. Ephremides, "Minimizing The Age of Information: NOMA or OMA?" in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 102–108.

[32] S. Kaul and R. Yates, "Age of Information: Updates With Priority," in *Proc. IEEE Int'l. Symp. Info. Theory (ISIT)*, Jun. 2018, pp. 2644–2648.

[33] A. Maatouk, M. Assaad, and A. Ephremides, "On the Age of Information in a CSMA Environment," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2020.

[34] R. D. Yates, "The Age of Information in Networks: Moments, Distributions, and Sampling," *IEEE Transactions on Information Theory*, vol. 66, no. 9, pp. 5712–5728, 2020.

[35] M. Moltafet, M. Leinonen, and M. Codreanu, "Moment Generating Function of the AoI in a Two-Source System With Packet Management," *IEEE Wireless Communications Letters*, vol. 10, no. 4, pp. 882–886, 2021.

[36] ——, "Source-Aware Packet Management for Computation-Intensive Status Updating: MGF of the AoI," in *2021 17th International Symposium on Wireless Communication Systems (ISWCS)*, 2021, pp. 1–6.

[37] M. Costa, M. Codreanu, and A. Ephremides, "On the Age of Information in Status Update Systems With Packet Management," *IEEE Trans. Info. Theory*, vol. 62, no. 4, pp. 1897–1910, Apr. 2016.

[38] J. Gong, Q. Kuang, X. Chen, and X. Ma, "Reducing age-of-information for computation-intensive messages via packet replacement," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2019, pp. 1–6.

[39] R. D. Yates, "The Age of Gossip in Networks," in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 2984–2989.

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

[41] R. Yates, "Lazy is Timely: Status Updates by an Energy Harvesting Source," in *Proc. IEEE Int'l. Symp. Info. Theory (ISIT)*, Jun. 2015, pp. 3008–3012.

[42] Y. Sun, E. Uysal-Biyikoglu, R. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," in *Proc. IEEE INFOCOM*, 4 2016.