Age of Information for Multiple-Source Multiple-Server Networks

Alireza Javani®, Student Member, IEEE, Marwen Zorgui®, Student Member, IEEE, and Zhiying Wang®, Member, IEEE

Abstract—Having timely and fresh knowledge about the current status of information sources is critical in a variety of applications, where the status update arrives at the destination later than its generation time due to processing and communication delays. The freshness of the status update at the destination is captured by the notion of the age of information. In this study, we analyze a multiple sensing network with multiple sources, multiple servers, and a monitor (destination). Each source corresponds to an independent piece of information, and its age is individually measured. Given a particular source, the servers independently sense the source of information and send the status update to the monitor. We assume that updates arrive at the servers according to Poisson random processes. Each server sends its updates to the monitor through a direct link, which is modeled as a queue. The service time to transmit an update is considered to be an exponential random variable. We examine both homogeneous and heterogeneous service and arrival rates for the single-source case, and heterogeneous arrival and service rates for the multiple-source case. We derive a closed-form expression for the average age of information under a last-come-first-serve (LCFS) queue for a single source and an arbitrary number of homogeneous servers. Using a recursive method, we derive the explicit average age of information for any number of sources and homogeneous servers. We also investigate heterogeneous servers and a single source, and present efficient algorithms for finding the average age of information. Optimal update scheduling strategies are also investigated in several scenarios, providing insights into enhancing the system performance in terms of update freshness.

Index Terms—Age of Information, wireless sensor network, status update, queuing analyses, monitoring network.

I. INTRODUCTION

In this paper, we introduce a novel approach to address the challenges of quantifying and minimizing the Age of Information (AoI) in a multiple-sensing network. The widespread use of sensor networks in applications like health monitoring [1], the Internet of Things (IoT) [2], stock market trading, and vehicular networks [3] emphasizes the need for up-to-date information. Outdated information in the monitoring facility can lead to undesirable consequences, making it crucial to ensure the freshness of the received status updates.

To tackle this issue, the concept of AoI was introduced in [4]. More specifically, the sources generate updates and transmit them to the monitoring facility through servers. The goal is to minimize AoI, defined as the time elapsed from the generation of the update to its arrival at the monitoring facility. AoI captures the timeliness of status updates, which is different from other standard communication metrics, such as delay, and throughput. It is affected by the inter-arrival time of updates, and the delay that is caused by queuing during update processing and transmission.

In this paper, we present a comprehensive investigation of AoI in a multiple-sensing network, taking into consideration both homogeneous and heterogeneous arrival and service rates. Consider a network with a number of shared information sources, whose status updates are transmitted to the monitor by n independent servers. For example, the sources of information can be some shared environmental parameters, and independently operated sensors in the surrounding area obtain such information. As another example, the sources of information can be the prices of several stocks, which are transmitted to the user by multiple independent service providers. Throughout this paper, a sensor or a service provider is called a server, since it is responsible for serving the updates to the monitor.

We assume that the arrival of status updates at the servers are independent events following Poisson random processes. The server itself is represented as a queue, with the service time for each update being modeled as an exponential distribution. We assume that information sources are independent and are sensed by n independent servers. We categorize the network into two types: homogeneous and heterogeneous, based on the information arrival rates and service rates. For homogeneous rates, both single-source and multiple-source networks are investigated. For heterogeneous rates, we focus on the case of a single source. We mainly consider the Last-Come-First-Serve with preemption in service (in short, LCFS) queue model. Namely, upon the arrival of a new update, the server immediately starts to serve it and drops any old update being served.

One special case of our model is the single-source multiple-server homogeneous network, which was inspired by the model in [5]. However, there are notable differences in the update assignment and the corresponding analysis between our model and [5]. In our model, the new update is submitted to each server independently based on an individual Poisson
random process. An equivalent view can be that the update is generated at a total rate, and then assigned to one of the $n$ servers uniformly at random. In [5], the new update is also generated at a total rate, but it is assigned to an idle server if it exists, and otherwise to the server with the largest age. Unlike the approach proposed by [5], our model does not require server cooperation, allowing for a simpler distributed sensing network. Additionally, our work distinguishes itself by thoroughly investigating the multiple-source setup and considering both homogeneous and heterogeneous server rates. To the best of our knowledge, this is the first study to comprehensively explore these aspects, setting it apart from previous research.

In this paper, we leverage the techniques of stochastic hybrid system (SHS) models, as developed in [6], to analyze the age of information. Despite the effectiveness of SHS models in simplifying AoI calculations by solving a set of linear equations, a critical challenge arises as the number of equations increases rapidly, sometimes exponentially, with the network size. To address this complexity, we exploit the symmetry and the recursive structure of the equations, which, in some cases, allows us to derive an explicit expression for AoI. When explicit derivation is not feasible, we develop algorithms to recursively solve the set of equations. These algorithms are specifically designed to efficiently manage a large number of equations by reducing the problem’s dimension, enabling systematic AoI derivation. We prove the correctness of these algorithms through nested induction across several cases. By combining explicit derivation with algorithmic approaches, we can analyze AoI within the multiple-sensing framework established by SHS models, and obtain insights into the system’s dynamics and behavior.

A. Contribution and Paper Organization

Our research progressively builds upon simpler models, delving into more sophisticated network configurations. Through our investigation of AoI, we derive closed-form expressions or develop novel algorithms, and advance the understanding of average AoI computation in multiple-sensing setups with varying degrees of heterogeneity.

• Our research paper starts with the formal definition of the multiple-sensing network model and a brief review of the notions of AoI and SHS in Section II. Our model is a novel distributed network and does not require server cooperation, which is beneficial for implementation in real-world distributed sensing networks. While previous studies with SHS mostly work with scenarios with a tractable number of states, our work allows more complex and realistic network configurations with an exponentially growing number of states.

• To gain insights into the system’s performance in terms of information freshness, in Section III-A we focus on a single-source multiple-server homogeneous network operating under the LCFS policy. We derive a closed-form expression for the average AoI, shedding light on the average AoI and its relationship with the network parameters.

• Expanding our investigation to more complex scenarios, we examine LCFS with multiple sources and multiple servers in a homogeneous network in Section III-B. We develop a recursive algorithm that efficiently solves the linear equations derived from SHS. We also specifically explore the cases of two and three servers, and derive the exact expressions for the average AoI. For the case of two servers, we find the best arrival rates of all the sources, which corresponds to the optimal update scheduling strategy. This approach provides deeper insights into optimizing the system’s performance.

• To address the challenges posed by heterogeneous networks with a single source and multiple servers, we propose a novel and low-complexity algorithm for calculating the average AoI in Section IV. Our approach exhibits a linear complexity in the number of system states, surpassing the quadratic to cubic complexity of general-purpose algorithms. Additionally, for the case of two servers, we develop exact expressions for the average AoI, and find the optimal arrival rate at each server given the service rates, offering insights into efficient update scheduling in heterogeneous systems. Simulation results demonstrate that the AoI under optimal update scheduling in heterogeneous systems is improved when the heterogeneity level of service rates is increased.

B. Related Work

In [4], the authors considered the single-source, single-server, and first-come-first-serve (FCFS) queue model and determined the arrival rate that minimizes AoI. A series of works afterward investigated average AoI minimization under various system models with multiple sources and/or servers. Different cases of multiple-source single-server under FCFS and LCFS were considered in [6] and [7] and the region of feasible age was derived. In [5] and [8], the system is modeled as a source that submits status updates to a network of parallel and serial servers, respectively, for delivery to a monitor, and AoI is evaluated. The parallel-server network is also studied in [9] when the number of servers is 2 or infinite, and the average AoI for the FCFS queue model was derived. The authors in [10] also considered a system with multiple sources, where packets are sent to parallel queues. They compute the average AoI of a system with only two parallel servers and compare the average AoI with the case of a single queue. In [11], the authors considered a model with multiple sources, a single queue, and multiple destinations.

AoI has also been optimized for different network models as a performance metric for various communication systems that deal with time-sensitive information, such as cellular wireless networks [12], [13], [14], [15], source nodes powered by energy harvesting [16], [17], [18], [19], [20], [21], [22], wireless erasure networks and coding [23], [24], [25], [26], [27], scheduling in networks [28], [29], [30], [31], [32], [33], unmanned aerial vehicle (UAV)-assisted communication systems [34], [35], [36], correlated status updates [37], and multi-hop networks [38], [39], [40], [41].

Several studies have employed the stochastic hybrid systems method to analyze the age of information across different system configurations. References [10] and [42] introduce system models with multiple sources, analyzing joint distributional properties and deriving differential equations for joint moments and moment-generating functions. They also examine networks with parallel queues, buffering capabilities, and preemption, comparing the average AoI for systems with parallel and single queues. References [43] and [44] focus on minimizing AoI in systems with multiple receivers and dynamic environments. They explore strategies like
resource augmentation and packet management to improve performance, establishing the stability and regularity of age-dependent systems. Reference [45] investigates a dual-server status update system, deriving average AoI for policies like zero wait and freeze/preempt. Finally, [46] analyzes real-time status update systems powered by energy harvesting using SHS. They derive closed-form expressions for the moment-generating function of AoI under various queuing disciplines, highlighting the necessity of considering higher moments in system optimization.

C. Notation

In this paper, we use boldface for vectors and normal font with subscripts for its elements. For example, for a vector \( \mathbf{x} \), the \( j \)-th element is denoted by \( x_j \). For non-negative integers \( a \) and \( b \), \( b \geq a \), we define \( [a : b] = \{a, a + 1, \ldots, b\} \), and \( [a] = \{1, 2, \ldots, a\} \). If \( a > b \), define \( [a : b] = \emptyset \).

II. SYSTEM MODEL AND PRELIMINARIES

In this section, we first present our network model, and then briefly review the stochastic hybrid system analysis from [6]. The network consists of \( m \) information sources that are sensed by \( n \) independent servers, as illustrated in Figure 1. Updates from the information sources are aggregated at the monitor after going through separate links. Server \( j \) collects updates from source \( i \) following a Poisson random process with rate \( \lambda^{(i)}_j \), \( j \in [n], i \in [m] \). For Server \( j \), the service time is an exponential random variable with average \( 1/\mu_j \), independent of all other servers. We focus on the queuing model of last-come-first-serve with preemption in service, or in short, LCFS. In this model, a server starts to transmit the new update right upon its arrival, thus dropping the previous update being served regardless of its source, if any.

A network is called homogeneous if \( \lambda^{(i)}_j = \lambda^{(i)} \mu_j = \mu \), for all \( j \in [n], i \in [m] \); otherwise, it is heterogeneous. In the case of a single source in a homogeneous network, we denote \( \lambda^{(i)} \) simply by \( \lambda \).

Consider one particular source. Suppose the freshest update at the monitor at time \( t \) is generated at time \( u(t) \). The age of information at the monitor (in short, AoI) is defined as \( \Delta(t) = t - u(t) \), which is the time elapsed since the generation of the last received update [4]. From the definition, it is clear that AoI linearly increases at a unit rate with respect to \( t \), except for some reset jumps to a lower value at points when the monitor receives a fresher update from the source.

The age of information for our network is shown in Figure 2. For a particular source, let \( t_1, t_2, \ldots, t_N \) be the generation times of all transmitted updates at all servers in increasing order. The black dashed lines show the age of every update. Let \( T_1, T_2, \ldots, T_N \) be the receipt times of all updates. The red solid lines show AoI.

Note that due to the contention among different updates at the same server, some updates may be dropped and not delivered at all. In addition, a newly arrived update may not have any effect on AoI because a fresher update may have already been delivered by another server. As an example, from the 6 updates shown in Figure 2, only updates 1, 3, 4, and 6 are useful and change AoI.

The interest of this paper is the average AoI for each source at the monitor. The average AoI [4] is the limit of the average age over time:

\[
\Delta \triangleq \lim_{T \to \infty} \frac{1}{T} \int_0^T \Delta(t) dt,
\]

and for a stationary ergodic system, it is also the limit of the average age over the ensemble:

\[
\Delta = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^n \Delta_i.
\]

In the paper, we view our system as a stochastic hybrid system and apply a method first introduced in [6] in order to calculate AoI. SHS is defined by a stochastic differential equation. For status update systems, AoI calculations can be implemented using a simplified SHS method, which leads to a system of ordinary first-order differential equations describing the temporal evolution of the expected value of the age process. The resulting methodology makes AoI calculations as simple as solving a linear set of equations. In SHS, the state is composed of a discrete state and a continuous state. The discrete state \( q(t) \in \mathcal{Q} \), for a discrete set \( \mathcal{Q} \), is a continuous-time discrete Markov chain, and the continuous-time continuous state \( x(t) = (x_0(t), x_1(t), \ldots, x_n(t)) \in \mathbb{R}^{n+1} \) is a continuous-time stochastic process. For example, the discrete state can represent which server has the freshest update in the network. For another example, we can use \( x_0(t) \) to represent the age at the monitor, and \( x_j(t) \) for the age at the \( j \)-th server, \( j = 1, 2, \ldots, n \). We can derive a set of first-order differential equations for the first-order moments of the continuous state. Here we consider systems where \( q(t) \) is a continuous-time finite-state Markov chain that describes the occupancy of servers and \( x(t) \) describes the continuous-time evolution of a collection of age-related processes.

Graphically, we represent each State \( q \in \mathcal{Q} \) by a node. For the discrete Markov chain \( q(t) \), transitions happen from one state to another through a directed transition edge \( l \), and the time spent before the transition occurs is exponentially distributed with rate \( \lambda(l) \). Note that it is possible to transit from one state to itself. The transition occurs when an update arrives at a server, or an update is received at the monitor. Thus, the transition rate is the update arrival rate or the service
rate, \( \lambda(l) \in \{\lambda_1^{(1)}, \ldots, \lambda_{m}^{(m)}, \mu_1, \ldots, \mu_n\} \). Denote by \( L_q \) and \( L'_q \) the sets of incoming and outgoing transitions of State \( q \), respectively. When transition \( l \) occurs, we write that the discrete state transits from \( q_l \) to \( q_{l'} \). For a transition, we denote that the continuous state changes from \( x \) to \( x' \). In our problem, this transition is linear in the vector space of \( \mathbb{R}^{n+1} \), i.e., \( x' = xA_l \), for some real matrix \( A_l \) of size \((n+1) \times (n+1)\). Note that when we have no transition, the age grows at a unit rate for the monitor and relevant servers, and is kept unchanged for irrelevant servers. Hence, within the discrete State \( q \), \( x(t) \) evolves as a piecewise linear function in time, namely, \( \frac{dx(t)}{dt} = b_q \), for some \( b_q \in \{0,1\}^{n+1} \). To illustrate the concepts, we use an example below.

**Example 1:** Consider the case of 2 heterogeneous servers and 1 source. At each time, we keep track of the age of information in the continuous state \( x = (x_0, x_1, x_2) \). Here \( x_0 \) is the age at the monitor, \( x_1 \) is the age for the first server, and \( x_2 \) is the age for the second server. In this example, the discrete states are \( Q = \{1,2\} \). In State 1, Server 1 contains the fresher information, i.e., \( x_1 \leq x_2 \); and in State 2, Server 2 has the fresher information, namely, \( x_2 \leq x_1 \). Obviously, our system changes its state when servers receive new information or when the updates are sent to the monitor. For instance, there is a transition from State 1 to State 2 with the rate of \( \lambda_2 \), when a new update arrives at Server 2 and the fresher information was at Server 1 before that. Hence, \( q_1 = 1 \) and \( q_2 = 2 \), which shows that State 2 is an outgoing transition for State 1, and State 1 is an incoming transition for State 2. Moreover, the continuous state changes from \( x = (x_0, x_1, x_2) \) to \( x' = (x_0, x_1, 0) \). For another instance, there is a transition from State 1 to itself with rate \( \mu_2 \) when an update is received at the monitor from Server 2. The continuous state changes from \( x = (x_0, x_1, x_2) \) to \( x' = (x_2, x_1, x_2) \). Notice that keeping the last entry in \( x' \) unchanged does not affect the age of information of the monitor for all future time.

For our purpose, we consider the discrete state probability

\[
\pi_q(t) \triangleq \mathbb{E}[\delta_{q,t}(t)] = P[q(t) = q],
\]

(1)

and the correlation between the continuous state \( x(t) \) (age process) and the discrete state \( q(t) \):

\[
\nu_q = (\nu_{q_1}(t), \ldots, \nu_{q_n}(t)) \triangleq \mathbb{E}[x(t)\delta_{q,t}(t)].
\]

(2)

Here \( \delta_{q,t}(t) \) denotes the Kronecker delta function, i.e., it equals 1 if \( q(t) = q \), and it equals 0 otherwise. Following Dynkin’s formula, for a piecewise linear SHS, we can prove that \( \pi_q(t) \) and \( \nu_q(t) \) obey the system of first-order ordinary differential equations. Readers can refer to [6] and [47] for a more in-depth explanation of SHS and details of the differential equations. When the discrete state \( q(t) \) is ergodic, \( \pi_q(t) \) converges uniquely to the stationary probability \( \pi_q \), for all \( q \in Q \). We can find these stationary probabilities from the following set of equations knowing that \( \sum_{q \in Q} \pi_q = 1 \):

\[
\pi_q \sum_{l \in L_q} \lambda(l) = \sum_{l \in L'_q} \lambda(l)\pi_{q_l}, \quad q \in Q.
\]

(3)

Moreover, when the first-order differential equations are stable and converge, the following lemma from [6] gives us the set of equations for limit values of \( \nu_q(t) \). We use this lemma to develop AoI analysis for our LCFS queue model.

**Lemma 1 [6]:** If the discrete-state Markov chain \( q(t) \) is ergodic with stationary distribution \( \pi \) and we can find a non-negative solution of \( \{v_q, q \in Q\} \) such that

\[
v_q \sum_{l \in L_q} \lambda(l) = b_q\pi_q + \sum_{l \in L'_q} \lambda(l)v_{q_l} A_l, \quad q \in Q,
\]

(4)

then the average age of information is given by

\[
\Delta = \sum_{q \in Q} v_q q_0.
\]

(5)

**III. AOI IN HOMOGENEOUS NETWORKS**

**A. Single Source Multiple Servers**

In this section, we present AoI calculation with the LCFS queue for the single-source, \( n \)-server homogeneous network using SHS techniques. Note that to compute the average AoI, Lemma 1 requires solving \( \{Q\}(n+1) \) linear equations of \( \{v_q, q \in Q\} \). The complexity of obtaining explicit solutions for these equations grows with the number of discrete states. Since the discrete state typically represents the number of idle servers in the system for homogeneous servers, \( |Q| \) should be \( n+1 \). In the following, we introduce a method inspired by [5] to reduce the number of discrete states and efficiently describe the transitions.

We define our continuous state \( x \) at time \( t \), as follows: the 0-th element \( x_0 \) is AoI at the monitor, the first element \( x_1 \) corresponds to the freshest update among all updates in the servers, the second element \( x_2 \) corresponds to the second freshest update in the servers, etc. With this definition, we always have \( x_1 \leq x_2 \leq \cdots \leq x_n \), for any time \( t \). Note that the index \( i \) of \( x_i \) does not represent a physical server index, but the \( i \)-th smallest age of information among the \( n \) servers. The physical server index for \( x_i \) changes with each transition. We say that the server corresponding to \( x_i \) is the \( i \)-th *virtual* server.

A transition indexed by \( l \) is triggered by (i), the arrival of an update at a server, or (ii) the delivery of an update to the monitor. Recall that we use \( x \) and \( x' \) to denote the continuous state of AoI right before and after the transition \( l \).

When one update arrives at the monitor and the server delivering the update becomes idle, we introduce a *fake update* to the server using the method introduced in [5]. Thus we can reduce the calculation complexities and only have one discrete state indicating that all servers are virtually busy. We denote this state by \( q = 0 \). In particular, we put the current update that is in the monitor to an idle server until the next update reaches this server. This assumption does not affect our final calculation for AoI, because even if the fake update is delivered to the monitor, AoI at the monitor does not change. Moreover, serving the fake update does not affect the service of future actual updates because of preemption in service.

When an update is delivered to the monitor from the \( k \)-th virtual server, the server becomes idle, and as previously stated, receives the fake update. The age at the monitor becomes \( x'_0 = x_k \), and the age at the \( k \)-th server becomes \( x'_{k} = x'_0 = x_k \). In this scenario, consider the update at the \( j \)-th virtual server, for \( j > k \). Its delivery to the monitor does not affect AoI since it is older than the current update of the monitor, i.e., \( x_j \geq x_k = x'_0 \). Hence, we can adopt a *fake preemption* where the update for the \( j \)-th virtual server, for all \( k \leq j \leq n \), is preempted and replaced with the fake current update at the monitor. Therefore, we set \( x'_j = x'_0 = x_k \),
we have one state, $x_l$ monitor and the age of each update in the system grows at a $\lambda$.

We set $Q$ can work in a distributed manner. As a beneficial result, the servers do not need to cooperate and $k$ reassigned virtual server numbers. Specifically, after transition $\lambda$, virtual server $v$ becomes virtual Server 1 becomes virtual Server 1, virtual Server 1 becomes virtual Server 2, ..., and virtual Server $l$ becomes virtual Server $l+1$. The transition rate is the arrival rate of the update, $\lambda$. The matrix $A_l$ is

$$
\begin{pmatrix}
0 & 1 & 2 & \ldots & l+1 & l+2 & \ldots & n \\
1 & 0 & 1 & \ldots & l & l+1 & \ldots & n \\
& \vdots & & & \ddots & & & \vdots \\
& & & & & 1 & \ldots & l+1 \\
& & & & & & & l+2 \\
& & & & & & & \vdots \\
& & & & & & & n \\
\end{pmatrix}
$$

$\lambda$.
For $i \in \{2, 3, \ldots, n-1\}$, from (11), we obtain

$$w_i + (n-i) \lambda (v_{i+1} - v_i) = \lambda (n-i+1) (v_i - v_{i-1}).$$

Hence, $w_{i+1} = v_{i+1} - v_i = \frac{\lambda (n-i+1)}{(n+1)(n-i)} w_i$. Setting $i = 2$ in (11), we have

$$((n-1) \lambda + \mu) v_2 = 1 + \mu v_1 + (n-1) v_1.$$  \hspace{1cm} (12)

Simplifying (12), we obtain $w_2 = v_2 - v_1 = \frac{1}{(n-1) \lambda + \mu}$. Therefore, we write

$$w_j = \frac{1}{n \lambda} \prod_{i=1}^{j-1} \frac{\lambda (n-i+1)}{i \mu + (n-i) \lambda}, \hspace{1cm} 2 \leq j \leq n. \hspace{1cm} (13)$$

Finally, setting $i = n$ in (11),

$$(\lambda + (n-1) \mu) v_n = 1 + \mu \sum_{j=1}^{n-1} v_j + \lambda v_{n-1}.$$  \hspace{1cm} (14)

implies $\mu \sum_{i=1}^{n} v_i = \mu \sum_{j=1}^{n-1} v_j + \mu v_n = (\lambda (n-1) \mu) v_n + \mu v_n - 1 - \lambda v_{n-1}$. Hence,

$$\frac{1}{n} \sum_{i=1}^{n} v_i = \frac{\lambda}{n \mu} w_n + v_n - \frac{1}{n \mu}. \hspace{1cm} (15)$$

Combining (10) and (15), we obtain the average AoI as

$$AoI = v_0 = v_n + \frac{\lambda}{n \mu} w_n = \frac{\sum_{j=2}^{n} w_j}{n} + \frac{\lambda}{n \mu} w_n,$$  \hspace{1cm} (16)

which is simplified to (8) using (13). \hfill \Box

Figure 4 shows the average AoI for different numbers of servers ($n = 1, 2, 3, 4, 10$), where the total arrival rate $n \lambda$ is kept equal across different cases to ensure a fair comparison. We observe that for up to 4 servers, a significant decrease in the average AoI occurs with the increase of $n$. However, increasing the number of servers beyond 4 provides only a negligible decrease in AoI.

In Figure 5, the average AoI versus the total service rate for a fair comparison is demonstrated. We can see that, given the same total service rate, as the number of servers increases the average AoI decreases. In Figure 6, LCFS (with preemption in service), LCFS with preemption in waiting, and FCFS queue models are compared numerically through simulation. Preemption in waiting means that when a new update arrives, we drop any old updates that have not been served. As can be seen from the figure, LCFS outperforms the other two queue models, indicating the importance of considering preemption in scheduling policies to reduce AoI. It coincides with the intuition that exponential service time is memoryless, and

**B. Multiple Sources Multiple Servers**

In this subsection, we present the average AoI with the LCFS queue for the $m$-source $n$-server homogeneous network. The arrival rate of Source $i$ at any server is $\lambda_{i}^{(i)} = \lambda^{(i)}$, for all $i \in [n], j \in [m]$. The arrival rate of the sources other than Source $i$ is $\lambda^{(i)} = \sum_{i \neq j} \lambda_{i}^{(i)}, i \in [m]$. The service rate at any server is $\mu$. Our goal is to compute $\Delta_{i}$, the average AoI at the monitor for Source $i$, $i \in [n]$. Without loss of generality, we calculate $\Delta_{1}$ for Source 1. In the queue model, upon arrival of a new update from any source, each server immediately drops any previous update in service, regardless of its source, and starts to serve the new update.

The continuous state $x$ represents the age for Source 1, and similar to the single-source case, it is defined as follows: $x_0$ is AoI of Source 1 at the monitor, $x_1$ is the age of the freshest update among all updates of Source 1 in the servers, $x_2$ corresponds to the second freshest update in the servers, etc. Therefore, $x_1 \leq x_2 \leq \ldots \leq x_n$, for any time $t$. Using fake updates and fake preemption as explained in Section III-A, we obtain an SHS with a single discrete state and $3n$ transitions described below:

*Case I: $l \in [0 : n-1]$: A fresh update arrives at virtual Server $l + 1$ from Source 1. This update is the freshest update, so $x'_{l} = 0$. Now, the previous freshest update becomes the freshest update among all updates of Source 1.*

| n  | 1  | 2  | 4  | 10 | 50 |
|----|----|----|----|----|----|
| $\lambda$ | 0.5 | 0.5 | 0.525 | 0.53 | 0.529 |

**Table II**

**Optimal Individual Arrival Rate for FCFS Queue, $\mu = 1$**
second freshest update, that is \( x'_2 = x_1 \), and so on. Then, \( x' = (x_0, 0, x_1, \ldots, x_{l}+2, \ldots, x_n) \). The transition rate is \( \lambda^{(1)} \).

Case II: \( l \in [n : 2n - 1] \): A fresh update arrives at virtual Server \( l' \triangleq l + 1 - n \) from Source \( i \neq 1 \). The age at the monitor does not change, namely, \( x'_0 = x_0 \). The \( l' \)-th freshest update is preempted. Moreover, since the virtual Server \( l' \) drops the update for the source of interest (Source 1), with the fake update, the \( l' \)-th virtual server becomes the \( n \)-th virtual server with age \( x_0 \). Therefore, we have \( x' = (x_0, x_1, \ldots, x_{l'}-1, x_{l'+1}, \ldots, x_n, x_0) \). The transition rate is \( \lambda^{(1)} \).

Case III: \( l \in [2n : 3n - 1] \): the update of Source 1 in virtual Server \( h \triangleq l + 1 - 2n \) is delivered. The age \( x_h \) is reset to \( x_0 \) and the virtual Server \( h \) becomes idle. Using fake update and fake preemption, we reset \( x'_0 = x_h, h \leq j \leq n \). The transition rate is \( \mu \).

**Algorithm 1** The Average AoI \( \Delta_1 \) of Source 1 for the Multiple-Source \( n \)-Server Homogeneous Network

**Input:** \( n, \lambda^{(1)}, \lambda^{(1)}, \mu \)

**Output:** \( \Delta_1 \)

**Part 1. Base case for \( v_1 \).**
\[
c_2 = \frac{n\lambda^{(1)}}{\lambda^{(1)}} \quad d_2 = \frac{-1}{\lambda^{(1)}} \quad \text{and} \quad c_3 = \frac{n\lambda^{(1)}((n-1)\lambda^{(1)}+\mu)}{2\lambda^{(1)}}, \quad d_3 = \frac{-1}{2\lambda^{(1)}} - \frac{(n-1)\lambda^{(1)}+\mu}{2\lambda^{(1)}}.
\]

**Part 2. Recursion for \( v_j, j \geq 2; \) for \( j = 2: n \) do**
\[
\text{end for}
\]
\[
v_j = \frac{1}{\mu} \sum_{i=j}^{\infty} \frac{1}{i} - \frac{1}{\mu} \sum_{i=j}^{\infty} \frac{1}{i+1} b_j \left( \frac{-1}{n} \right)
\]

**Part 2.**

**Theorem 2:** Consider the \( m \)-source \( n \)-server homogeneous network, for \( n \geq 3 \). The average AoI for Source 1 can be computed in a recursive manner, as in Algorithm 1.

**Proof:** By applying Lemma 1, and dropping the index \( q = 0 \), the system of equations for \( \nu_0 = \nu = (v_0, v_1, \ldots, v_n) \) becomes:

\[
(n\lambda_1 + n\lambda^{(1)} + n\mu)(v_0, v_1, \ldots, v_n) = (1, 1, 1, \ldots, 1, 1, 1, 1) + \lambda_1(v_0, 0, v_2, v_3, \ldots, v_n) + \lambda_1(v_0, 0, v_1, v_3, \ldots, v_n) + \lambda_1(v_0, 0, v_1, v_2, \ldots, v_n)
\]

To find the average AoI \( \Delta_1 = \nu_0 \) we need to solve the system of equations in (17), and prove that the solution to \( v_i \), \( 0 \leq i \leq n \), is positive. Equations in (17) are equivalent to

\[
n\mu v_0 = 1 + \mu \sum_{i=1}^{n} v_i,
\]

\[
v_1(\lambda^{(1)} + n\lambda^{(1)}) = 1 + \lambda^{(1)} v_2.
\]

And for \( 2 \leq i \leq n \),

\[
n(\lambda + \mu)v_i = 1 + (i - 1)\lambda^{(1)} v_i + (n - i + 1)\lambda^{(1)} v_{i-1} + i\lambda^{(1)} v_{i+1} + \mu \sum_{j=1}^{i-1} v_j + (n - i + 1)\mu v_i,
\]

where \( v_{n+1} \triangleq v_0 \) and \( \lambda = \lambda^{(1)} + \lambda^{(1)} = \sum_{i=1}^{n} \lambda_i \).

Let us rewrite the equations using the difference of adjacent \( v_j \)'s. From (20), we have for \( 2 \leq i \leq n \),

\[
((n - i)\lambda^{(1)} + i\lambda^{(1)} + (i - 1)\mu) v_i
\]

\[
= 1 + \lambda^{(1)}(n - i + 1)v_{i-1} + i\lambda^{(1)} v_{i+1} + \mu \sum_{j=1}^{i-1} v_j.
\]

We plug in \( i + 1 \) in (21) and subtract the resulting equation from Equation (21). Therefore,

\[
((n - i)\lambda^{(1)} + i\lambda^{(1)} + \mu)(v_{i+1} - v_i)
\]

\[
= \lambda^{(1)}(n - i + 1)(v_{i-1} - v_{i-1}) + (i + 1)\lambda^{(1)}(v_{i+2} - v_{i+1})
\]

Let us define \( w_i = v_i - v_{i-1} \), then we have:

\[
((n - i)\lambda^{(1)} + \lambda^{(1)} + i\mu)w_{i+1}
\]

\[
= \lambda^{(1)}(n - i + 1)w_i + (i + 1)\lambda^{(1)} w_{i+2}.
\]

Define for each \( i \in \{2, \ldots, n - 1\} \), coefficients \( r_{i+2} = \frac{(n-i)\lambda^{(1)} + i\lambda^{(1)} + \mu}{(i+1)\lambda^{(1)}} \) and \( t_{i+2} = \frac{\lambda^{(1)}(n-i+1)}{(i+1)\lambda^{(1)}} \), then

\[
w_{i+2} = r_{i+2} w_{i+1} + t_{i+2} w_i, \quad \text{for} \ 2 \leq i \leq n - 1.
\]

To show that each \( v_j \) is positive, and also to determine its value, our proof is inductive. For the base case, we find the
value of $v_1$ and show it is positive. Then, using induction, assuming $v_1, \ldots, v_j$ are positive and we know their values, we find $v_{j+1}$ and prove that it is positive.

**Base case:** We will find $v_1$ and show that $v_1 > 0$. As we can see from the recursive equations in (23), we can express each $w_j$ for $j \in \{1, \ldots, n+1\}$ in terms of $w_2$ and $w_3$. Write such expressions as $w_j = x_j v_3 + y_j w_2$ where $x_4 = r_4$, $y_3 = t_4$, $x_5 = r_5 t_4 + t_5$, $y_4 = r_5 t_4$ and $x_6 = y_6 r_{x+1} + t_j x_{x+1}$, $y_{j+1} = r_{j+1} y_j + t_{j+1} y_{j-1}$ for $5 \leq j \leq n$. So far we can write $w_j$ for $4 \leq j \leq n+1$ as a linear function of $w_2$ and $w_3$ which are in fact linear functions of $v_1, v_2, v_3$ because $w_2 = v_2 - v_1$ and $w_3 = v_3 - v_2$. We also know from (19) and (21) for $i = 2$:

$$
\begin{align*}
((n-1)\lambda^{(1)} + 2\lambda^{(1)} + \mu)v_2 \\
= 1 + \lambda^{(1)}(n-1)v_1 + 2\lambda^{(1)}v_3 + \mu v_1.
\end{align*}
$$

Combining (19) and (24) together we reach the conclusion that we can write $v_2, v_3$, and all the $w_i$, $2 \leq i \leq n+1$, based on $v_1$. Hence for some coefficients $c_i, d_i$, we write

$$
w_i = c_i v_1 + d_i.
$$

Next, using (another) induction we will show that for $i \in \{2,3,\ldots, n+1\}$,

$$
c_i > 0 \text{ and } d_i < 0.
$$

For $i = 2, 3$, from equation (20) we have

$$
w_2 = \frac{v_1 n \lambda^{(1)} - 1}{\lambda^{(1)}}.
$$

$$
w_3 = \frac{((n-1)\lambda^{(1)} + \mu)w_2 - 1}{2\lambda^{(1)}}.
$$

Therefore, $c_2 = \frac{n \lambda^{(1)}}{\lambda^{(1)}}$, $d_2 = -\frac{1}{2\lambda^{(1)}}$ and $c_3 = \frac{n \lambda^{(1)}((n-1)\lambda^{(1)} + \mu)}{2\lambda^{(1)}}$, $d_3 = -\frac{1}{2\lambda^{(1)}} - \frac{((n-1)\lambda^{(1)} + \mu)}{2\lambda^{(1)}}$. Hence the claim in (25) holds.

Assume that (25) holds for $2, 3, \ldots, i$, where $3 \leq i \leq n$. We will prove that it also holds for $i+1$. We can rewrite Equation (21) as

$$
1 + i \lambda^{(1)} w_{i+1} = (n-i+1)\lambda^{(1)} w_i + \mu \sum_{k=1}^{i-1} (v_i - v_k)
$$

$$
= (n-i+1)\lambda^{(1)} w_i + \mu \sum_{j=1}^{i-1} \sum_{k=j}^{i} w_j
$$

$$
= c v_1 + d,
$$

for some constants $c > 0, d < 0$. The last equality follows from the induction hypothesis (25) and the fact that (29) consists of $w_j$’s where $j \leq i$. The above equation implies $c_{i+1} > 0, d_{i+1} < 0$. Therefore by induction the condition in (25) holds.

From (18),

$$
v_0 = \frac{1}{n \mu} + \sum_{i=1}^{n} \frac{v_i}{n} = \frac{1}{n \mu} + \sum_{j=2}^{n} \sum_{k=2}^{j} w_k + v_1
$$

$$
= \frac{1}{n \mu} + v_1 + \sum_{j=2}^{n} \frac{n - j + 1}{n} w_j.
$$

Moreover,

$$
v_0 = v_{n+1} = w_{n+1} + v_n = w_{n+1} + \sum_{j=2}^{n} w_j + v_1.
$$

Comparing (31), (32) and using $w_j = c_i v_1 + d_i$, we have

$$
\frac{1}{n \mu} = \sum_{j=2}^{n} \frac{1}{n} w_j = \sum_{j=2}^{n} \frac{1}{c_j} + \sum_{j=2}^{n} \frac{j-1}{n} d_j.
$$

We can obtain $v_1$ by

$$
v_1 = \frac{1 - \sum_{j=2}^{n} \frac{j-1}{n} d_j}{\sum_{j=2}^{n} \frac{1}{c_j}}.
$$

It can be seen that, by the condition of (25), $v_1$ is positive and we found its value in (34).

**Induction step:** We assume that we obtained the values of $v_1, \ldots, v_{j-1}$ and they are positive. We need to show that $v_j$ is positive and find its value. From now on, $v_1, \ldots, v_{j-1}$ are considered positive constants.

From (23) and considering that $v_1, \ldots, v_{j-1}$ are positive constants, it is obvious that we can write for $j \leq i \leq n$ and some constants $a_i, b_i$,

$$
w_i = a_i v_j + b_i.
$$

Next, We prove by (another) induction that for $j \leq i \leq n+1$,

$$
a_i > 0 \text{ and } b_i < 0.
$$

Since $w_j = v_j - v_{j-1}$ and also $v_{j-1}$ is assumed to be a positive constant, the condition in (35) is true for $j$.

We assume (35) holds for $j, j+1, \ldots, i$, and prove it for $i+1$. We make use of (28) again:

$$
1 + i \lambda^{(1)} w_{i+1} = (n-i+1)\lambda^{(1)} w_i + \mu \sum_{k=1}^{i-1} (v_i - v_k)
$$

$$
= (n-i+1)\lambda^{(1)} w_i + \mu \sum_{k=1}^{j-1} (v_i - v_k) + \mu \sum_{k=j}^{i} (v_i - v_k)
$$

$$
= (n-i+1)\lambda^{(1)} w_i + \mu \sum_{r=j+1}^{i} (w_r + v_j - v_k)
$$

$$
= \mu \sum_{k=j}^{i} w_r + \mu \sum_{k=j}^{i} v_j
$$

$$
= a_{i+1} v_j + b_{i+1},
$$

where $a_{i+1} > 0, b_{i+1} < 0$, are some constants. The last step holds because the right-hand side of (39) consists of $v_k$ (1 $\leq k \leq j-1$) and $w_r$ ($j + 1 \leq r \leq i$), and we assumed the condition (35) holds, and $v_k$ is a positive constant for $k < j$. From the above equation, the condition in (35) holds for $w_{i+1}$. Thus, we have proved the condition in (35) by induction.

Now, we intend to prove $v_j$ is positive and find its value recursively based on $v_1, \ldots, v_{j-1}$ and some constants. Similar to the way we found $v_1$ as in (31), (32), we write (18) as

$$
v_0 = \frac{1}{n \mu} + \sum_{i=1}^{n} \frac{v_i}{n} = \frac{1}{n \mu} + \frac{v_j}{n} + \sum_{r=j+1}^{n} \frac{\sum_{k=j+1}^{r} w_k + v_j}{n}.
$$

(41)
Moreover,
\[ v_0 = w_{n+1} + v_n = w_{n+1} + \sum_{k=j+1}^{n} w_k + v_j. \] (42)

Combining (41), (42), we have
\[ \frac{1}{n\mu} + \frac{\sum_{i=1}^{j-1} v_i}{n} = \frac{j - 1}{n} v_j + w_{n+1} + \sum_{i=j+1}^{n} \frac{i-1}{n} w_i. \] (43)

From (43) we can write
\[ v_j = \frac{1}{n\mu} + \sum_{i=1}^{j-1} \frac{v_i}{n} - \sum_{i=j+1}^{n} \frac{i-1}{n} b_i, \] (44)

where the denominator and the numerator are both positive, by condition (35). Therefore, we proved that \( v_j \) is positive and also found its value recursively. The solution to \( v_1, \ldots, v_n \) using the recursive calculation, and the age of information \( \Delta_1 = v_0 \) is summarized in Algorithm 1. \( \square \)

Let \( \Delta_i \) denote the average AoI at the monitor for Source \( i \). In the corollaries below, we state the average AoI for \( n = 2, 3 \) servers, which can be directly derived from Theorem 2.

Here, we define \( \lambda \triangleq \lambda_1^{(1)} + \lambda_2^{(2)} + \cdots + \lambda_m^{(m)} \), \( \rho^{(i)} = \frac{\lambda^{(i)}}{\mu} \), \( \rho = \sum_{i=1}^{n} \rho^{(i)} = \frac{\lambda}{\mu} \) and recall each server has service rate \( \mu \).

**Corollary 1:** For \( m \) information sources and \( n = 2 \) servers, we have
\[ \Delta_i = \frac{1}{2(\lambda + \mu)} + \frac{\lambda + \mu}{2\mu \lambda^{(i)}}, \quad 1 \leq i \leq m. \] (45)

**Corollary 2:** For \( m \) information sources and \( n = 3 \) servers, we have
\[ \Delta_i = \frac{1}{3\mu} \left( 5\rho^{(i)} + 2(\rho + 1)^2 \right) (\rho + 1), \quad 1 \leq i \leq m, \]
where \( \rho = \frac{\lambda}{\mu} \) and \( \rho^{(i)} = \frac{\lambda^{(i)}}{\mu} \).

Figure 7 examines the average AoI for two sources when the utilization factor \( \rho^{(1)} \) changes from 0 to 2.5, with \( \rho = 2.5 \), \( \mu = 1 \), and \( n = 2 \). While \( \mu \) represents a fixed system parameter determined by the server’s capability, the arrival rate \( \lambda^{(i)} \) represents an adjustable sampling rate or load at the source. Thus an update scheduling strategy is defined to be a particular partial derivative with respect to each \( \lambda^{(i)} \) to be zero.

\[ \frac{\partial}{\partial \lambda^{(i)}} (w_1 \Delta_1 + w_2 \Delta_2 + \cdots + w_m \Delta_m + a \sum_{j=1}^{m} \lambda^{(j)} - \lambda) = 0, \] (47)

for \( i \in [m] \). Here \( a \) is the Lagrange multiplier. Simplifying (47) results in:
\[ \frac{w_1}{(\lambda^{(1)})^2} = \frac{w_2}{(\lambda^{(2)})^2} = \cdots = \frac{w_n}{(\lambda^{(m)})^2} = a. \] (48)

Knowing the fact that \( \lambda^{(1)} + \lambda^{(2)} + \cdots + \lambda^{(m)} = \lambda \), we obtain the result in Theorem 3. \( \square \)

From Theorem 3, when there are 2 servers and the weights are all identical, i.e., \( w_1 = w_2 = \cdots = w_m \), the optimal arrival rate should be equal for all sources. In general, the optimal arrival rate is inversely proportional to the square root of the weight. Here, different choices of \( \lambda^{(i)} \) translates to different update scheduling policies.

**IV. AOI IN SINGLE-SOURCE HETEROGENEOUS NETWORKS**

**A. Overview**

In this section, we consider a single source and assume that the arrival rates and service rates of servers are arbitrary. We denote by \( \lambda_1^{(i)} \triangleq \lambda_j \) the arrival rate of a single source at Server \( j \), and \( \mu_j \) the service rate of Server \( j \in [n] \). In this setting, we can no longer use the technique of virtual servers used in the homogeneous case to reduce the state space and derive AoI. In particular, we need to keep track of the age of updates at the physical servers, as well as their ordering, resulting in \( n! \) number of states. However, we can still use fake updates and fake preemption, such that the server is always busy even after its update is delivered or is outdated. If we consider \( n \) servers, we will have \( n! \) states, \((2n)n!\) transitions and \((n+1)n!\) equations and unknowns. Writing down the \((n+1)n!\) equations from Lemma 1 in a matrix form, we obtain \( TV = \pi \), for the coefficient matrix \( T \) and the steady state probabilities \( \pi \). First, we find steady-state probabilities. Then, we prove that we can break down matrix \( T \) into submatrices which have the same general form as some matrix \( T_0 \). Afterward, by doing some column and row operations on matrix \( T_0 \), we show that we are able to solve all the equations and eventually find the average AoI.

In the following subsections, we present the notations, the main theorem, and examples with 2 and 3 servers.
B. Notations and Definitions

A permutation of the set \(\{1, 2, \ldots, n\}\) is denoted by a lower-case letter or a tuple of length \(n\), e.g., \(q = (q_1, q_2, \ldots, q_n)\). Additionally, set by default \(q_0 = 0\). A permutation \(q\) is said to be \(j\)-increasing if the last \(j\) positions are increasing:

\[q_{n-j+1} < q_{n-j+2} < \cdots < q_n.\]

If a permutation is \(2\)-increasing, it is said to be odd. Otherwise, it is even. Define a permutation \(h_i(\cdot)\) that takes the \(i\)-th element of \(q\) and place it at the first position:

\[h_i(q) = (q_i, q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n),\]

for \(i \in [n]\). Let its inverse be \(h^{-1}_i(\cdot)\). Define the set

\[H^{-1}_q = \{h^{-1}_i(q) : i = 1, 2, \ldots, n\}.\]

Define a function \(g_{j,k}(\cdot)\) that takes the \(k\)-th element of \(q\) and place it at the \((n-j)\)-th position:

\[g_{j,k}(q) = (q_1, \ldots, q_{n-j-1}, q_k, q_{n-j}, \ldots, q_{k-1}, q_{k+1}, \ldots, q_n),\]

for \(k = n-j, \ldots, n\).

Denote by \(g_{j,k}^{-1}(\cdot)\) the inverse permutation.

Given the set of linear equations:

\[Av = b,\]

the matrix \(A\) is called the coefficient matrix, \(v\) the variable vector, and \(b\) the constant vector. The matrices and vectors will be indexed by permutations and/or integers. Let \(m, n\) be the row index and the column index for a matrix \(A\), then \(A(m, n)\) is the \((m, n)\)-th entry. Let \(M, N\) be sets of rows and columns indices for a matrix \(A\), then \(A(M, N)\) is the corresponding sub-matrix of \(A\) with rows \(M\) and columns \(N\). Moreover, \(A(\cdot, N)\) is the sub-matrix of \(A\) with columns \(N\), and \(A(M, \cdot)\) is the sub-matrix with rows \(M\). For a vector \(v\), its \(n\)-th entry is denoted by \(v_n\), and its sub-vector indexed by \(N\) is denoted by \(v(N)\).

C. Main Result

In this subsection, we derive the algorithm to compute the AoI of the heterogeneous network. To simplify the presentation, the proofs for the results are provided in the appendix.

First, let us describe the SHS. The continuous state \(x = (x_0, x_1, \ldots, x_n)\) represents the ages of the monitor, Server 1, \ldots, and Server \(n\). The set of discrete states \(Q\) is the set of all permutations of the set \(\{1, 2, \ldots, n\}\). There are \(n!\) states in total. State \(q = (q_1, q_2, \ldots, q_n) \in Q\) represents the ordering of the age among all the servers, meaning \(x_{q_1} \leq x_{q_2} \leq \cdots \leq x_{q_n}\).

The incoming transitions of state \(q = (q_1, q_2, \ldots, q_n)\) are listed in Figure 8. Here for an incoming state \(p\), \(v_{p,q}A_l\) corresponds to the last term in Equation (4). For ease of exposition, the entries in vector \(x\) are reordered as \((x_0, x_1, \ldots, x_{q_n})\). By abuse of notation, in Figure 8, the reordered vector is still called \(x\). Similarly, \(x', v_p\) and \(p\) are also reordered.

For transition \(l, 1 \leq l \leq n\), state \(p = (q_2, \ldots, q_{l-1}, q_l, q_{l+1}, \ldots, q_n)\) is an incoming state of state \(q\), corresponding to an update arrival at server \(q_l\) with rate \(\lambda_{q_l}\). The \(q_1\)-th entry in \(x'\) becomes 0. Accordingly, the \(q_1\)-th entry in \(v_p\) becomes 0. The set of incoming states of \(q\) for such transitions can be represented as \(H^{-1}_q\).

For transition \(l, n+1 \leq l \leq 2n\), set \(l = n - l\). An update is delivered to the monitor from Server \(q_l\) with rate \(\mu_{q_l}\), and \(q\) is an incoming state to itself. In this case, we preempt any update in the servers that has a larger information age and put a fake update in them which is the update from Server \(q_l\). In other words, we preempt updates in servers \((q_{l+1}, \ldots, q_n)\), and replace them with the update from Server \(q_l\). Therefore, the new vector \(x'\) becomes \((x_{q_1}, x_{q_1}, \ldots, x_{q_{n-1}}, x_{q_n}, x_{q_n})\). Similarly, the corresponding vector \(v_p\) changes to \((v_{p,q_1}, v_{p,q_1}, \ldots, v_{p,q_{n-1}}, v_{p,q_n}, \ldots, v_{p,q_n})\).

Now, we write down Equation (4), as in Lemma 1 for each state \(q \in Q\). Notice that each update arrival or update delivery results in an outgoing state for state \(q\). Hence, on the left-hand side of Equation (4), \(v_q\) is multiplied by the sum of rates of outgoing transitions, which for every \(q \in Q\) is equal to \(\sum_{j=1}^{n} \lambda_{q_j} + \sum_{j=1}^{n} \mu_{q_j}\). Also, \(b_q = [1, \ldots, 1]\) due to the fake update, and \(p_q\) is the stationary distribution to be computed by Lemma 2. The last term on the right-hand side of (4) can be expressed according to Figure 8. Therefore, for \(q \in Q\),

\[(v_{q,0}, v_{q,q_1}, v_{q,q_2}, \ldots, v_{q,q_n})(\sum_{j=1}^{n} \lambda_{q_j} + \sum_{j=1}^{n} \mu_{q_j}) = \pi_q + \lambda_{q_1} (\sum_{p \in H^{-1}_q} v_{p,0}, 0, v_{p,q_2}, \ldots, v_{p,q_n})) + \sum_{i=1}^{n} \mu_{q_i} (v_{q,q_1}, v_{q,q_1}, \ldots, v_{q,q_{i-1}}, v_{q,q_i}, \ldots, v_{q,q_n}).\]

The following lemma gives the steady-state probability, which only depends on arrival rates \(\lambda_i\) and the order of the update’s age in a state.

Lemma 2: For a given state \(q = (q_1, q_2, \ldots, q_n)\) in which \(q \in Q\), the steady state probability \((\pi_q)\) is:

\[\pi_q = \frac{\lambda_{q_1}}{\sum_{j=1}^{n} \lambda_{q_j}} \cdot \frac{\lambda_{q_2}}{\sum_{j=2}^{n} \lambda_{q_j}} \cdot \frac{\lambda_{q_3}}{\sum_{j=3}^{n} \lambda_{q_j}} \cdots \frac{\lambda_{q_{n-1}}}{\sum_{j=n-1}^{n} \lambda_{q_j}}.\]

In the next theorem, we represent the equations of (51) in matrix form as \(Tv = \pi\) for some coefficient matrix \(T\) and some constant vector \(\pi\). In total, there are \((n+1)\) equations since there \(n!\) states and each \(v_q = (v_{q,0}, v_{q,q_1}, v_{q,q_2}, \ldots, v_{q,q_n})\) has \(n+1\) entries. We represent the row and column indices of matrix \(T\) using 2 tuples of \((q, i)\) and \((p, k)\) where \(p\) are any 2 arbitrary permutations and \(i, k\) are numbers in \(\{0, 1, \ldots, n\}\). In particular, \(v_{q,p}\) corresponds to column index \((p, k)\) in the coefficient matrix, and the \(i\)-th equation (out of \(n+1\)) in equation (51) corresponds to row \((q, i)\). Accordingly, vectors \(v, \pi, T\) are indexed by \((p, k)\).

Lemma 3: The \((n+1)!\) transition equations in (51) can be written as

\[Tv = \pi.\]

Here, the constant vector \(\pi\) has entry \(\pi_p\) in row \((p, k)\), for all \(0 \leq k \leq n\), and any permutation \(p\). The coefficient matrix \(T\) is as (53), as shown at the bottom of the next page. Here \(k = (i)_j\) means that \(k = i - 1 \) if \(i \leq j\), and \(k = i \) if \(i > j\).

Next, we show that solving the original set of equations simplifies solving smaller sets of equations separately. In Algorithm 2, we break down the \((n+1)\) equations into smaller sets to solve all variables \(v_{q,i}\) with fixed \(i\) and fixed \(q_i, q_{i+1}, \ldots, q_n\) (Line 5). Namely, we solve \((i-1)!\) variables at
Algorithm 2 AoI Calculation of $n$-Server Heterogeneous Network

1: for $i = 1, 2, \ldots, n$
2:   for distinct $c_i, \ldots, c_n \in [n]$
3:     $N \leftarrow \{(q, i) : (q_i, \ldots, q_n) = (c_i, \ldots, c_n)\}$
4:     $N' \leftarrow \{(q, i) : (q_i, \ldots, q_n) \neq (c_i, \ldots, c_n)\}$
5:     Solve $T(N, N)\nu(N) = c(N)$ (will use Algorithm 3)
6:     $c(N) \leftarrow c(N) + T(N, N)\nu(N)$
7:     $\nu(N)$
8: end for
9: $N \leftarrow \{(q, 0) : \text{all permutations } q\}$
10: $\nu(N) \leftarrow \{\nu_{q, 0} : \text{all permutations } q\}$
11: $\text{AoI} \leftarrow \sum \nu_{q, 0}$ where $T(N, N)\nu(N) = c(N)$ (will use Algorithm 3)

The breakdown is justified in Lemma 4. We show that the $(i-1)!$ equations in Line 5 and the $n!$ equation in Line 11 have coefficient matrices in the same form, denoted as $T_0$. The equations defined by $T_0$ will be solved by Algorithm 3 explained later.

**Lemma 4:** Define $T_0$ parameterized by $i$ to be the following $i! \times i!$ matrix,

$$T_0(q, p) = \begin{cases} \sum_{j=2}^{n} \lambda_{q_j} + \sum_{j=1}^{n} \mu_{q_j}, & \text{if } q = p, \\ -\lambda_{q_1}, & \text{if } q = h_j(p), j = 2, \ldots, i, \\ 0, & \text{o.w.} \end{cases}$$

Moreover, we define $T_0$ parameterized by $i = 0, 1$ to be the scalar

$$T_0 = \mu_1.$$ (55)

Solving the set of equations in Lemma 3 is equivalent to solving the equations corresponding to $T_0$, parameterized by $0, 1, 2, \ldots, n$, shown in Lines 5 and 11 of Algorithm 2. Thus far, the entire system of equations can be solved once we solve equations defined by $T_0$. In Algorithm 3, we provide a recursive method for solving equations defined by $T_0$, which breaks down $T_0$ into matrices in the same form as $T_0$, but with smaller parameters. Thus, the AoI can be expressed by $\sum \nu_{q, 0}$ (Algorithm 2 Line 11) and computed from Algorithm 3 Line 32 according to Lemma 3. Moreover, Lemma 6 shows the correctness of Algorithm 3 and the non-negativity of the solution.

**Lemma 5:** The result of Algorithm 3 Line 32 is

$$\sum_{\text{all permutations } q} \nu_q^{(0)} = \sum_{i=1}^{n} \nu_q^{(0)} c_q^{(0)}.$$ (54)
Algorithm 3 Single-Source $n$-Server Heterogeneous Network

1: function HETERO_SOLVER($n, T_0, c^{(0)}$)
2: ▷ Solve the equation $T_0 v^{(0)} = c^{(0)}$.
3: ▷ Output: $\{v_q^{(0)} : q \text{ all } q\}$, and $v^{(n-1)}_{(1,2,\ldots,n)} = \sum_{\text{all permutations } q} v_q^{(0)}$.
4: ▷ Base cases:
5: if $n = 0$ or $1$ then
6: $v_1 \leftarrow c^{(0)}_{T_0}$
7: end if
8: ▷ Forward path:
9: for $j = 1, 2, \ldots, n-1$ do
10: ▷ Column operation:
11: $T_j' \leftarrow T_j - 1$
12: for each $(j + 1)$-increasing $p$ do
13: $T_j'(\cdot, g_j(k(p))) \leftarrow T_j - 1(\cdot, g_j(k(p))) - T_{j-1}(\cdot, \cdot)$, for $k = n - j + 1, \ldots, n$
14: end for
15: ▷ Row operation:
16: $T_{j}'' \leftarrow \sum_{k=n-j}^{n} T_{j}(k, g_j(k,q))$
17: for each $(j + 1)$-increasing $q$ do
18: $T_j''(q, :) \leftarrow \sum_{k=n-j}^{n} (T_j(k, g_j(k,q), :))$
19: $c_j^{(j)} \leftarrow \sum_{k=n-j}^{n} c_j^{(j-1)}$
20: end for
21: ▷ Pick specific rows and columns:
22: ▷ Variables $v^{(j)}$, $v^{(j-1)}$ satisfy $T_j v^{(j)} = c^{(j)}$, $R_j v^{(j)}(Q) = c^{(j)}(Q)$, $S_j v^{(j)}(Q)$
23: $Q \leftarrow \{q : q \text{ is } (j + 1)-\text{increasing}\}$
24: $\bar{Q} \leftarrow \{q : q \text{ is } j-\text{increasing but not } (j + 1)-\text{increasing}\}$
25: $T_j \leftarrow T_j''(Q, Q)$
26: $R_j \leftarrow T_j''(Q, Q)$
27: $S_j \leftarrow T_j''(Q, Q)$
28: end for
29: ▷ Now $T_{n-1}, c^{(n-1)}$ are both scalars
30: $v^{(n-1)}_{(1,2,\ldots,n)} \leftarrow c^{(n-1)}_{T_n}$
31: ▷ Backward path:
32: for $j = n - 1, n - 2, \ldots, 1$ do
33: ▷ For distinct $c_{n-j}, \ldots, c_n \in [n]$ such that $c_{n-j+1} < \cdots < c_n < c_{n-j+1}$
34: $N \leftarrow \{q : \{q, n_j, \ldots, g_n(k)\} = c_{n-j}, \ldots, c_n\}$
35: $v^{(j-1)}(N) \leftarrow \text{HeteroSolver}(n - j - 1, R_j(N, N), c^{(j-1)}(N) - S_j(N, :) ) v^{(j)}(N)$
36: end for
37: $v_p^{(j-1)} \leftarrow v_p^{(j)} - \sum_{k=n-j+1}^{n} v_{g_j(k,p)}^{(j-1)}$, for $j + 1$-increasing $p$
38: end for
39: end function

Lemma 6: Consider the following linear equation:

$$T_0 v^{(0)} = c^{(0)},$$

where $T_0$ as defined in (54) is parameterized by $n \geq 0$.

- Correctness: Algorithm 3 finds its solution.
- Non-negativity: The solution is non-negative if the entries of $c^{(0)}$ are non-negative and $\mu_1, \ldots, \mu_n, \lambda_1, \ldots, \lambda_n > 0$.

In summary, we can calculate the AoI by Algorithm 2, of which the equations in Lines 5 and 11 are solved by Algorithm 3.

**Theorem 4:** The AoI of a heterogeneous network with one source and $n$ servers is

$$AoI = \sum_{q} v_{q,0},$$

such that $v = \{v_{q,i} : q \text{ is a permutation of length } n, 0 \leq i \leq n\}$ satisfy

$$Tv = \pi,$$

which is solved by Algorithms 2 and 3.

**Lemma 7:** The computational complexity of Algorithm 2 is $O((n+1)!)^3$ for solving the $(n+1)!$ equations in (57).

Remark 1: If we were to naively invert the matrix $T$, the complexity using Gauss–Jordan elimination would be $O((n+1)!)$, and using the state of the art algorithm for inverting a large matrix (Optimized CW-like algorithms) would be $O((n+1)!^2) [48]$. Even though the complexity issue still exists due to the massive number of equations to be solved ($(n+1)!$), achieving $O((n+1)!)$ time complexity is a great improvement compared to any other algorithm.

D. Cases With 2 and 3 Servers

Before proving that Algorithms 2 and 3 solve the $(n+1)!$ equations, we show how they execute when $n = 2$ and 3. From these two examples, we demonstrate the intuition of finding the average AoI, and our proof for the general case follows similar steps. In particular, the lemmas in Section IV-C can be generalized from these two examples.

**Example 2:** In the case of $n = 2$, we have only 2 states: $(1, 2)$ and $(2, 1)$. State $(2, 1)$ is defined as the state that Server 1 contains a fresher update compared to Server 2, and State $(2, 1)$ as the state that Server 2 has the fresher update. Upon arrival of an update at each server or receipt of an update at the monitor, we observe some self-transitions and intra-state transitions. Transition rates and mappings are illustrated in Table III.

Steady state probabilities are found knowing that $\pi_{(1,2)} + \pi_{(2,1)} = 1$ and $\pi_{(1,2)} \lambda_2 = \pi_{(2,1)} \lambda_1$. Therefore, we have $\pi_{(1,2)} / \pi_{(2,1)} = \frac{\lambda_1}{\lambda_1 + \lambda_2} \lambda_2 + \lambda_2$.

$$\lambda_1 + \lambda_2 + \mu_1 + \mu_2 v_{(1,2)} = b_1 \pi_{(1,2)} \lambda_1 v_{(1,0), 0, v_{12}} + \mu_1 (v_{11}, v_{11}, v_{11}, v_{11}, v_{11}, v_{11}, v_{11}, v_{12}),$$

$$\lambda_1 + \lambda_2 + \mu_1 + \mu_2 v_{(2,1)} = b_2 \pi_{(2,1)} \lambda_2 v_{(1,0), 0, v_{11}, v_{11}} + \mu_2 (v_{22}, v_{22}, v_{22}, v_{22}, v_{22}, v_{22}),$$

where $v_{(1,2)} = (v_{(1,2), 0}, v_{(1,2), 1}, v_{(1,2), 2}, v_{(1,2), 3}, v_{(1,2), 4}, v_{(1,2), 5})$, and $b_1 = b_2 = (1, 1, 1)$. Therefore, we have six equations and six unknowns here. By writing down the equations from equations (58) and (59) in matrix form, $Tv = \pi$ will be as (60), as shown at the bottom of the next page.

We can see that matrix $T$, here, matches the general form in Lemma 3. Now, we show these equations have non-negative solutions, and use Lemma 1 to find the AoI. First, we look
at the rows/columns \((1, 2), 1\) and \((2, 1), 2\) notice that they form a diagonal matrix of size 2 by 2. Therefore we can solve and remove the variables \(v_{1(2),1}, v_{2(1),2}\). They correspond to variables \(v_{q,q}\) in Lemma 4. They are also non-negative since the 2 diagonal entries and the entries of vector \(\pi\) are positive. Second, consider rows/columns \((1, 2), 2\) and \((2, 1), 1\) corresponding to variables \(v_{q,q}\) in Lemma 4, again we obtain a 2 by 2 diagonal matrix. Hence, we are able to find the variables \(v_{1(2),2}, v_{2(1),1}\). After removing 4 variables, we are left with matrix \(T_0\), which is in the same form as Equation (54):

\[
\begin{pmatrix}
1 & 2 \\
2 & 1
\end{pmatrix}
\begin{pmatrix}
0 & 0 \\
0 & 0
\end{pmatrix}
\begin{pmatrix}
(1, 2) \\
(2, 1)
\end{pmatrix}
\begin{pmatrix}
\lambda_2 + \sum_{i=1}^{2} \mu_i & -\lambda_1 \\
-\lambda_2 & \lambda_1 + \sum_{i=1}^{2} \mu_i
\end{pmatrix}
\begin{pmatrix}
0 \\
0
\end{pmatrix}
= \begin{pmatrix}
0 \\
0
\end{pmatrix}
\begin{pmatrix}
1 & 2 \\
2 & 1
\end{pmatrix}
\begin{pmatrix}
\lambda_2 + \sum_{i=1}^{2} \mu_i & -\lambda_1 \\
-\lambda_2 & \lambda_1 + \sum_{i=1}^{2} \mu_i
\end{pmatrix}
\begin{pmatrix}
0 & 0 \\
0 & 0
\end{pmatrix}
\begin{pmatrix}
0 \\
0
\end{pmatrix}
. \tag{61}
\]

We can solve the matrix \(T_0\) only after one iteration of Algorithm 3. The corresponding variables are denoted as \(v^{(0)} = (v_{1(2),1}, v_{2(1),2})^\top \). By definition in Section IV-B, the permutation \((1, 2)\) is odd \((2)\)-increasing), and \((2, 1)\) is even. In the forward path of Algorithm 3, we perform column operation in Line 15, meaning subtracting the odd column from the even one, and then the row operation in Line 20, meaning adding the even row to the odd row. After these 2 operations, the matrix \(T_0\) becomes \(T_1\):

\[
\begin{pmatrix}
1 & 2 \\
2 & 1
\end{pmatrix}
\begin{pmatrix}
0 & 0 \\
0 & 0
\end{pmatrix}
\begin{pmatrix}
\sum_{i=1}^{2} \mu_i \\
\sum_{i=1}^{2} \mu_i + \sum_{i=1}^{2} \lambda_i
\end{pmatrix}
\begin{pmatrix}
0 \\
0
\end{pmatrix}
= \begin{pmatrix}
0 \\
0
\end{pmatrix}
\begin{pmatrix}
\sum_{i=1}^{2} \mu_i \\
\sum_{i=1}^{2} \mu_i + \sum_{i=1}^{2} \lambda_i
\end{pmatrix}
. \tag{62}
\]

After the column operation, the second variable \(v_{2(1),0}\) remains unchanged, and the first variable becomes \(v_{1(2),0} = v_{1(2),2} + v_{2(1),2} = v_{1(2),2} + v_{2(1),0}\). From (62) we can solve the first equation with the first variable \(v_{1(2),0}\), whose coefficient matrix (Line 27) is \(T_1 = \sum_{i=1}^{3} \mu_i\). The remaining coefficient matrix (Line 28) for the second variable \(v_{2(1),0}\) is \(R_1 = \sum_{i=1}^{2} \mu_i + \sum_{i=1}^{2} \lambda_i\).

In the backward path of Algorithm 3, we find the first variable \(v_{1(2),0}\), and then the second variable \(v_{2(1),0}\). Now we can find the variable \(v_{1(2),0} = v_{1(2),2} - v_{2(1),0}\). One can see from (62) that \(v_{2(1),0}\) is non-negative, and we will show in Lemma 6 that \(v_{1(2),0}\) is also non-negative. So the solution to (60) is non-negative, and by Lemma 1 the AoI equals \(v_{1(2),1}\).

**Example 3:** Consider the case with \(n = 3\) servers.

By writing down the equations in (51), we have \(T\nu = \pi\), where the constant vector \(\pi\) is

\[
\begin{pmatrix}
\pi_{1(2),3}, \pi_{1(2),2}, \pi_{1(2),3}, \pi_{1(2),1}, \pi_{1(2),3}, \pi_{1(2),2}, \pi_{1(2),1}, \\
\pi_{1(3),2}, \pi_{1(3),1}, \pi_{1(3),2}, \pi_{1(3),1}
\end{pmatrix}
\]

and the coefficient matrix \(T\) is in Figure 9. We can see that it matches our result in Lemma 3 as expected. Non-zero elements of the first 4 rows indexed by the permutation \((1, 2, 3)\) are in columns indexed by permutations \((1, 2, 3), (2, 1, 3), (2, 3, 1)\), which are the incoming states of \((1, 2, 3)\). Non-zero elements of the first 4 columns are in rows indexed by \((1, 2, 3), (2, 1, 3), (3, 1, 2)\), which are the outgoing states of state \((1, 2, 3)\).

We illustrate how to solve \(T\nu = \pi\). Variables \(v_{q,q}\) correspond to a diagonal sub-matrix of \(T\) with size \(n!\times n!\) (the $S$ entries), and we can find their values. After removing these variables, to find \(v_{q,q}\), we solve the ones that the last 2 entries of their permutation are the same. For instance if \((q_1, q_2) = (2, 3)\) we see that we only need to solve the single variable \(v_{1(2,3),2}\) corresponding to the 3rd row/column. Therefore, we can solve all these variables individually. For solving \(v_{q,q}\), we solve the ones that the last entry of their permutation is the same. For instance if \(q_3 = 3\), we need to

\[
\begin{pmatrix}
(1, 2) & 0 \\
(1, 2) & 1 \\
(1, 2) & 2 \\
(2, 1) & 0 \\
(2, 1) & 1 \\
(2, 1) & 2
\end{pmatrix}
\begin{pmatrix}
\lambda_2 + \sum_{i=1}^{2} \mu_i & -\mu_1 & -\mu_2 & -\lambda_1 \\
0 & \lambda_1 + \lambda_2 & 0 & 0 \\
0 & -\mu_1 & \lambda_2 + \mu_1 & 0 \\
0 & 0 & 0 & \lambda_1 + \lambda_2 \\
0 & -\lambda_2 & 0 & 0 \\
0 & 0 & 0 & -\mu_2
\end{pmatrix}
\begin{pmatrix}
v_{1(2),0} \\
v_{1(2),1} \\
v_{2(1),0} \\
v_{2(1),1} \\
v_{2(1),2} \\
v_{2(1),1}
\end{pmatrix}
= \begin{pmatrix}
\pi_{1(2),0} \\
\pi_{1(2),1} \\
\pi_{1(2),2} \\
\pi_{2(1),0} \\
\pi_{2(1),1} \\
\pi_{2(1),2}
\end{pmatrix}
. \tag{60}
\]
After the second run of the forward path, we perform row and column operations on $T_0$, and obtain $T''_2$ as (65), as shown at the bottom of the page. Therefore, the sub-matrices $T_0$ and $R_1$ are as follows, respectively, (66) and (67), as shown at the bottom of the next page.

Since we performed column operations on each iteration of Algorithm 3, the variables change accordingly. After the first run of the algorithm, the new variables corresponding to $T_1$ are as follows:

$$v^{(1)}_{(1,3,2)} = v^{(1)}_{(1,3,3)} = v^{(1)}_{(2,1,3)}, \quad v^{(1)}_{(2,3,1)} = v^{(1)}_{(2,3,2)}, \quad v^{(1)}_{(3,1,2)} = v^{(1)}_{(3,1,3)}, \quad v^{(1)}_{(3,2,1)} = v^{(1)}_{(3,2,3)}.$$

The remaining variables corresponding to $R_1$ are unchanged:

$$v^{(1)}_{(1,3,2)} = v^{(1)}_{(2,1,3)}, \quad v^{(1)}_{(2,3,1)} = v^{(1)}_{(2,3,2)}, \quad v^{(1)}_{(3,2,1)} = v^{(1)}_{(3,2,3)}.$$

Hence, $T_2 = \sum_{i=1}^{3} \mu_i$ and $R_2$ is the $2 \times 2$ diagonal matrix with diagonal entries $\sum_{i=1}^{3} \mu_i$. Correspondingly, the new variable corresponding to $T_2$ is

$$v^{(2)}_{(1,2,3)} = v^{(2)}_{(1,3,2)} = v^{(2)}_{(2,1,3)} = v^{(2)}_{(2,3,1)}, \quad v^{(2)}_{(3,1,2)} = v^{(2)}_{(3,2,1)}.$$

and the other two variables corresponding to $R_2$ are not changed:

$$v^{(2)}_{(1,3,2)} = v^{(2)}_{(2,1,3)} = v^{(2)}_{(2,3,1)}, \quad v^{(2)}_{(3,1,2)} = v^{(2)}_{(3,2,1)}.$$

Fig. 9. Coefficient matrix $T$ for $n = 3$. For row $(q, i)$, symbol * is $\sum_{j=2}^{3} \lambda_{qj} + \sum_{j=1}^{i} \mu_{ij}$, $*$ is $\sum_{j=2}^{3} \lambda_{qj} + \mu_{q1}$, and # is $\sum_{j=2}^{3} \lambda_{qj} + \sum_{j=1}^{i} \mu_{ij}$.

solve variables $v^{(1,2,3)}_{3}$ and $v^{(2,1,3)}_{3}$ together. The resulting coefficients for these variables are as follows:

$$\begin{bmatrix}
\lambda_2 + \lambda_3 + \mu_1 + \mu_2 & -\lambda_1 \\
-\lambda_2 & \lambda_1 + \lambda_3 + \mu_1 + \mu_2
\end{bmatrix} = 2$$
}

and we can see that we solved this in Equation (61) of Example 2 with a change of variable. At the end, we need to solve variables $v_{3,q_1}$ or in another word $T_0$ which is (64), as shown at the bottom of the page. In the first run of the forward path in Algorithm 3, we perform row and column operations on $T_0$, and obtain $T''_1$ as (63), as shown at the bottom of the page. Therefore, the sub-matrices $T_1$ and $R_2$ are as follows, respectively, (66) and (67), as shown at the bottom of the next page.

\begin{align*}
(1, 2, 3) & \quad (1, 3, 2) & \quad (2, 1, 3) & \quad (2, 3, 1) & \quad (3, 1, 2) & \quad (3, 2, 1) \\
(1, 3, 2) & \quad \lambda_2 + \lambda_3 + \sum_{i=1}^{2} \mu_i & \quad 0 & \quad -\lambda_1 & \quad -\lambda_1 & \quad 0 & \quad 0 \\
(2, 1, 3) & \quad 0 & \quad \lambda_2 + \lambda_3 + \sum_{i=1}^{2} \mu_i & \quad 0 & \quad 0 & \quad -\lambda_1 & \quad -\lambda_1 \\
(2, 3, 1) & \quad -\lambda_2 & \quad -\lambda_2 & \quad \lambda_1 + \lambda_3 + \sum_{i=1}^{2} \mu_i & \quad 0 & \quad 0 & \quad 0 \\
(3, 1, 2) & \quad 0 & \quad 0 & \quad 0 & \quad \lambda_1 + \lambda_3 + \sum_{i=1}^{2} \mu_i & \quad -\lambda_2 & \quad -\lambda_2 \\
(3, 2, 1) & \quad 0 & \quad 0 & \quad -\lambda_3 & \quad -\lambda_3 & \quad 0 & \quad \lambda_1 + \lambda_3 + \sum_{i=1}^{2} \mu_i
\end{align*}

(64)

\begin{align*}
\begin{bmatrix}
\lambda_2 + \lambda_3 + \sum_{i=1}^{2} \mu_i & \quad 0 & \quad -\lambda_1 & \quad 0 & \quad -\lambda_1 & \quad 0 \\
0 & \quad \lambda_2 + \lambda_3 + \sum_{i=1}^{2} \mu_i & \quad 0 & \quad 0 & \quad 0 & \quad -\lambda_1 \\
-\lambda_2 & \quad 0 & \quad \lambda_1 + \lambda_3 + \sum_{i=1}^{2} \mu_i & \quad 0 & \quad 0 & \quad -\lambda_2 \\
0 & \quad 0 & \quad \lambda_2 + \lambda_3 + \sum_{i=1}^{2} \mu_i & \quad 0 & \quad 0 & \quad -\lambda_2 \\
-\lambda_3 & \quad 0 & \quad -\lambda_3 & \quad 0 & \quad \lambda_1 + \lambda_3 + \sum_{i=1}^{2} \mu_i & \quad 0 \\
0 & \quad 0 & \quad -\lambda_3 & \quad 0 & \quad 0 & \quad \lambda_1 + \lambda_3 + \sum_{i=1}^{2} \mu_i
\end{bmatrix}
\end{align*}

(65)
In the backward path, we solve the variables. From $T_2$, we can find the variable $v^{(2)}_{(1,2,3)}$ and then after removing it from the matrix $T_2^{(2)}$, we can find $v^{(2)}_{(1,2,3)}$ and $v^{(2)}_{(1,3,2)}$. Hence, we can solve the variables in (68). After removing these variables, the variables in (69) can be solved since $R_1$ is diagonal. Finally, $v_{q,0}$ can be solved for any $q$ using (68) and (69). The non-negativity of the solution is shown in Lemma 6, and by Lemma 1 the AoI can be computed as $v^{(2)}_{(1,2,3)}$.

In the following, we derive average AoI explicitly in the case of $n = 2$ in Example 2.

**Theorem 5:** Consider one source and $n = 2$ heterogeneous servers. The AoI is given by:

$$\Delta = \frac{1}{\mu_1 + \mu_2} + \frac{1}{\lambda_1 + \lambda_2} + \frac{1}{\mu_1 + \mu_2} \left( \frac{\mu_1 \lambda_2}{\lambda_1 + \mu_2} + \frac{\mu_2 \lambda_1}{\lambda_2 + \mu_1} \right).$$

**Proof:** Following the solution in Example 2, we find the 6 variables corresponding to $v_{1,2} = (v^{(2)}_{(1,2,0)}, v^{(2)}_{(1,2,1)}, v^{(2)}_{(1,2,2)})$ and $v_{2,1} = (v^{(2)}_{(2,1,0)}, v^{(2)}_{(2,1,1)}, v^{(2)}_{(2,1,2)})$ as:

$$v^{(2)}_{(1,2,1)} = \frac{v_{1,2}}{\lambda_1 + \lambda_2} \text{ and } v^{(2)}_{(2,1,2)} = \frac{v_{2,1}}{\lambda_1 + \lambda_2}.$$  

Also from Lemma 2 we know that, $\pi = [\pi^{(1,2)}, \pi^{(2,1)}] = [\frac{\lambda_1}{\lambda_1 + \lambda_2}, \frac{\lambda_2}{\lambda_1 + \lambda_2}]$. Following the steps in Example 2, the average AoI is $v_{(1,2),0} + v_{(2,1),0}$ which simplifies to:

$$AoI = \frac{1}{\mu_1 + \mu_2} + \frac{\mu_1 (v^{(2)}_{(1,2),1} + v^{(2)}_{(1,2),2}) + \mu_2 (v^{(2)}_{(1,2),2} + v^{(2)}_{(2,1),2})}{\mu_1 + \mu_2}$$

$$= \frac{1}{\mu_1 + \mu_2} + \frac{1}{\lambda_1 + \lambda_2} + \frac{1}{\mu_1 + \mu_2} \left( \frac{\mu_1 \lambda_2}{\lambda_1 + \mu_2} + \frac{\mu_2 \lambda_1}{\lambda_2 + \mu_1} \right).$$

The proof is completed.

Next, for $n = 2$ servers, we find the optimal arrival rates, $\lambda_1^*$, $\lambda_2^*$, representing the optimal update scheduling, given fixed service rates $\mu_1, \mu_2$ and sum arrival rate $\lambda \triangleq \lambda_1 + \lambda_2$.

The optimal $\lambda_1^*$ is illustrated in Figure 10.

**Theorem 6:** For one source and $n = 2$ heterogeneous servers, given $\mu_1, \mu_2$ and fixed $\lambda_1 + \lambda_2 = \lambda$, the optimal $\lambda_1^*$ satisfies

- if $\mu_1 < \mu_2$ and $\mu_2^2 - \frac{\mu_1 (\lambda + \mu_1) (\lambda + \mu_2)}{\mu_2} < 0$, (73), as shown at the bottom of the next page;
- if $\mu_1 < \mu_2$ and $\mu_2^2 - \frac{\mu_1 (\lambda + \mu_1) (\lambda + \mu_2)}{\mu_2} \geq 0$:
  $$\lambda_1^* = 0, \lambda_2^* = \lambda.$$  

**Fig. 10.** Optimal value of $\lambda_1$ as a function of $\mu_1$. $\lambda_1 + \lambda_2 = \lambda, \mu_1 + \mu_2 = 100.$

- if $\mu_1 > \mu_2$ and $\mu_2^2 \geq \frac{\mu_1 (\lambda + \mu_1) (\lambda + \mu_2)}{\mu_1}$:
  $$\lambda_1^* = \lambda, \lambda_2^* = 0.$$  

- if $\mu_1 > \mu_2$ and $\mu_2^2 < \frac{\mu_1 (\lambda + \mu_1) (\lambda + \mu_2)}{\mu_1}$, (76), as shown at the bottom of the next page.

**Proof:** In order to find the optimal values of $\lambda_1$ and $\lambda_2$ for given values of $\mu_1, \mu_2, \lambda$ where $\lambda_1 + \lambda_2 = \lambda$, we set the derivative of the following equation with respect to $\lambda_1, \lambda_2$ and $a$ to zero.

$$AoI = \frac{1}{\mu_1 + \mu_2} + \frac{\mu_1 (v^{(2)}_{(1,2,1)} + v^{(2)}_{(1,2,2)}) + \mu_2 (v^{(2)}_{(1,2,2)} + v^{(2)}_{(2,1,2)})}{\mu_1 + \mu_2}$$

$$= -a (\lambda_1 + \lambda_2 - \lambda),$$

$$\frac{\partial AoI}{\partial \lambda_1} = -1 \frac{\mu_2 (2 \lambda_1 + \lambda_2 + \mu_2)}{(\lambda_1 + \lambda_2)^2} \left( \lambda_1 + \lambda_2 \right) (\lambda_1 + \mu_2) - a = 0,$$

$$\frac{\partial AoI}{\partial \lambda_2} = -1 \frac{\mu_1 (2 \lambda_2 + \lambda_1 + \mu_1)}{(\lambda_1 + \lambda_2)^2} \left( \lambda_1 + \lambda_2 \right) (\lambda_1 + \mu_2) - a = 0.$$  

Also, we know that $\lambda_1 + \lambda_2 = \lambda$. With some algebraic simplification we reach to this 2nd order polynomial equation for finding the optimal value of $\lambda_1$ and consequently $\lambda_2$.

$$\lambda_1 (c - 2 \lambda_1) + 2 \lambda_1 (\mu_2 + c (\lambda + \mu_1)) + \mu_2^2 - c (\lambda + \mu_1)^2, \quad (82)$$

where $c = \frac{\mu_1 (\lambda + \mu_2)}{\mu_2 (\lambda + \mu_1)}$.

When $c = 1$, it is equivalent to $\mu_1 = \mu_2$, and the equation (82) becomes a first order polynomial which results in $\lambda_1 = \lambda_2 = \frac{\lambda}{2}$. This polynomial has 2 real roots because of its
positive discriminant, and therefore solving the equation (82) gives us 2 possible candidates for our optimization problem. When \( \mu_1 < \mu_2 \), then \( c < 1 \). Knowing the fact that for 2 roots of (82) we have,

\[
\begin{align*}
  r_1 + r_2 &= \frac{\mu_2 + \mu_1(\lambda + \mu_2)}{\mu_2}, \\
  r_1r_2 &= \frac{\mu_1^2 - \mu_1(\lambda + \mu_1)(\lambda + \mu_2)}{1 - c},
\end{align*}
\]

we conclude when \( \mu_1 < \mu_2 \) and \( \mu_2^2 - \mu_1(\lambda + \mu_1)(\lambda + \mu_2) \geq 0 \), the 2 roots are negative and therefore in this regime our optimal values become \( \lambda_1 = 0, \lambda_2 = \lambda \). When \( \mu_1 < \mu_2 \) and \( \mu_2^2 - \mu_1(\lambda + \mu_1)(\lambda + \mu_2) \geq 0 \), the positive root is the optimal rate which is equal to:

\[
\begin{align*}
  \lambda_1 &= \frac{-(\mu_2 + c(\lambda + \mu_1)) + \sqrt{\mu_1(\lambda + \mu_2)(2 + \frac{\mu_1}{\lambda + \mu_1} + \frac{\lambda + \mu_1}{\mu_2})}}{1 - \frac{\mu_1(\lambda + \mu_2)}{\mu_2(\lambda + \mu_1)}},
\end{align*}
\]

Similarly, by writing the 2-nd order polynomial for \( \lambda_2 \), we reach to the conclusion that when \( \mu_1 > \mu_2 \), \( \mu_2^2 \geq \frac{\mu_2(\lambda + \mu_1)(\lambda + \mu_2)}{\mu_1} \) the optimal rates are \( \lambda_1 = \lambda, \lambda_2 = 0 \). In the regime that \( \mu_1 > \mu_2 \) and \( \mu_2 < \frac{\mu_2(\lambda + \mu_1)(\lambda + \mu_2)}{\mu_1} \), the positive root is the optimal rate.

\[
\begin{align*}
  \lambda_2 &= \frac{-(\mu_1 + (\lambda + \mu_1)) + \sqrt{\mu_2(\lambda + \mu_2)(2 + \frac{\mu_1}{\lambda + \mu_2} + \frac{\lambda + \mu_2}{\mu_1})}}{1 - \frac{\mu_2(\lambda + \mu_1)}{\mu_1(\lambda + \mu_2)}},
\end{align*}
\]

The proof is completed. \( \square \)

**E. Numerical Results**

In the following section, we conduct numerical evaluations for heterogeneous networks. In the evaluations, we assume the service rates \( \mu_i \)'s are fixed system parameters determined by the capability of the servers, and the arrival rates \( \lambda_i \)'s are adjustable sampling rates at the information source. We view the arrival rate as an indicator for the load of the associated server. Each choice of \( \lambda_i \) represents an update scheduling strategy. In several evaluations, the optimal update scheduling are chosen such that the lowest permissible AoI is achieved. The purpose of these evaluations is to demonstrate the relationship (1) between the optimal update scheduling strategy and the given service rates, (2) between the AoI and the level of heterogeneity of the network, and (3) between the AoI and the different update scheduling strategies.

When \( \mu_1 = \mu_2 \), the optimal rates that minimize AoI in Theorem 6 are \( \lambda_1^* = \lambda_2^* = \frac{\lambda}{2} \). As Figure 10 illustrates, for \( \mu_1 = \mu_2 = 50 \), optimal rates are \( \lambda_1^* = \frac{\lambda}{2} \). In the regimes where one of the service rates is much greater than the other one, AoI minimizes when all the updates are sent to the server with the greater service rate.

Also, when \( n = 3 \) and the sum of the service rates is \( \mu_1 + \mu_2 + \mu_3 = 100 \), we notice a saturation region in Figure 11 similar to Figure 10. Here, Server 1 and Server 2 have the same arrival rate \( (\lambda_1 = \lambda_2) \) and service rate \( (\mu_1 = \mu_2) \). It shows that when the service rate for one of the servers is much greater compared to the other 2 servers, it is optimal in terms of minimizing average AoI to allocate most of the arrival rate to that server. Furthermore, when the service rates are equal \( (\mu_1 = \mu_2 = \mu_3 = 100) \), the optimal update arrival rates are also equal \( \lambda_1^* = \lambda_2^* = \lambda_3^* = \frac{\lambda}{3} \). This observation matches the results for \( n = 2 \) in Figure 10.

Furthermore, we investigate the relationship between the age of information and the level of heterogeneity of the service rates \( \mu_i \)'s. To that end, we consider \( n = 3, 4 \) and define a parameter \( \alpha \) as the ratio of the service rates, specifically \( \alpha = \frac{\mu_i}{\mu_j} \), for \( 1 \leq i \leq n - 1 \). This means that the service rate ratio of any two adjacent servers is kept constant. Due to symmetry, we only consider the case where \( \alpha > 1 \). A larger \( \alpha \) implies a higher disparity in service rates among the servers. The AoI analysis is presented under the constraints \( \sum_{i=1}^{n} \lambda_i = 30 \) and \( \sum_{i=1}^{n} \mu_i = 15 \). We observe similar behavior when the total service rate and the total arrival rate vary. As mentioned, the optimal update scheduling selects the arrival rates \( \lambda_i \)'s in order to achieve the lowest permissible AoI. The solid lines in Figure 12 and Figure 14 demonstrate that as \( \alpha \) increases, the best permissible AoI (under optimal update scheduling) decreases. This result indicates that higher disparity in service rates improves the information freshness.

When \( n = 3 \), Figure 13 illustrates the optimal arrival rates \( (\lambda_1, \lambda_2, \text{and } \lambda_3) \) for each \( \alpha, \lambda_1 \) and \( \lambda_2 \) decrease as \( \alpha \) increases, while \( \lambda_3 \) increases significantly, showing that

\[
\lambda_1^* = \frac{-(\mu_2 + c(\lambda + \mu_1)) + \sqrt{\mu_1(\lambda + \mu_2)(2 + \frac{\mu_2}{\lambda + \mu_1} + \frac{\lambda + \mu_1}{\mu_2})}}{1 - \frac{\mu_2(\lambda + \mu_1)}{\mu_1(\lambda + \mu_2)}},
\]

\[
\lambda_1^* = \frac{-(\mu_1 + (\lambda + \mu_1)) + \sqrt{\mu_2(\lambda + \mu_2)(2 + \frac{\mu_1}{\lambda + \mu_2} + \frac{\lambda + \mu_2}{\mu_1})}}{1 - \frac{\mu_2(\lambda + \mu_1)}{\mu_1(\lambda + \mu_2)}},
\]
the system allocates more load to the server with the highest service rate as \( \alpha \) grows. In fact, when \( \alpha \) is greater than 2.47, all updates should be scheduled to the two servers with the highest service rates. When \( \alpha \) exceeds 7.98, the system’s load is all concentrated in the most powerful server.

For \( n = 4 \), Figure 15 presents the optimal arrival rates for each \( \alpha \), showing distinct and insightful trends. As \( \alpha \) increases, we observe that both \( \lambda_1 \) and \( \lambda_2 \) decrease, indicating a reduction in the arrival rates for the first two servers as the disparity in service rates grows. In contrast, \( \lambda_3 \) initially increases slightly before decreasing steadily. Also, \( \lambda_4 \) increases substantially with \( \alpha \), suggesting that the system allocates progressively more load to the server with the highest service rate as \( \alpha \) grows. Similar to the case of three servers, only a subset of the servers need to serve the updates as \( \alpha \) reaches certain threshold.

For the above heterogeneous networks, we also investigate the AoI variation due to different update scheduling, where three scheduling strategies are compared: optimal \( \lambda \) allocation, equal \( \lambda \) allocation, and proportional \( \lambda \) allocation with \( \alpha = \frac{\lambda_{i+1}}{\lambda_i} \) for \( 1 \leq i \leq n - 1 \). From Figures 12 and 14, we observe that the optimal \( \lambda \) allocation consistently achieves the lowest AoI across all values of \( \alpha \) for both \( n = 3 \) and \( n = 4 \). The equal \( \lambda \) allocation results in the highest AoI, and we observe that the age increases as \( \alpha \) grows, indicating that the equal allocation strategy does not effectively respond to changes in service rates. The proportional \( \lambda \) allocation offers a middle ground, performing better than the equal allocation but not as well as the optimal strategy. These observations highlight the importance of update scheduling or \( \lambda \) allocation strategies in heterogeneous systems to minimize the AoI and enhance the overall system performance.

V. CONCLUSION

In this paper, we studied the age of information in the presence of multiple independent servers monitoring several information sources. We derived the AoI for the LCFS queue model using SHS analysis for the homogeneous network with a single source. We also provided an algorithm for deriving the AoI when we have \( m \) sources and \( n \) servers in a homogeneous network. For the heterogeneous network, an algorithm is presented that can solve the exponential number of equations for this case and find the AoI. Future directions include deriving an explicit formula of AoI in heterogeneous sensing networks where the update arrival rates and/or service rates are different for any number of sources and servers. Also, investigating arrival and service time distributions other than exponential distributions can further enrich the system model.

REFERENCES

[1] E. Kasaeyan Naeini, S. Shahhosseini, A. Subramanian, T. Yin, A. M. Rahmani, and N. Dutt, “An edge-assisted and smart system for real-time pain monitoring,” in Proc. IEEE/ACM Int. Conf. Connected Health: Appl., Syst. Eng. Technol. (CHASE), Sep. 2019, pp. 47–52.
[2] R. Trimananda, S. A. H. Aqajari, J. Chuang, B. Demsky, G. H. Xu, and S. Lu, “Understanding and automatically detecting conflicting interactions between smart home IoT applications,” in Proc. 28th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng., 2020, pp. 1215–1227.
[3] R. Du, C. Chen, B. Yang, N. Lu, X. Guan, and X. Shen, “Effective urban traffic monitoring by vehicular sensor networks,” IEEE Trans. Veh. Technol., vol. 64, no. 1, pp. 273–286, Jan. 2015.
[4] S. Kaul, R. Yates, and M. Gruteser, “Real-time status: How often should one update?” in Proc. IEEE INFOCOM, Mar. 2012, pp. 2731–2735.
A. Arafa, K. Banawan, K. G. Seddik, and H. Vincent Poor, “Timely estimation using coded quantized samples,” in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Jun. 2020, pp. 1812–1817.

A. Ferdowsi, M. A. Abd-Elmajid, W. Saad, and H. S. Dhillon, “Neural combinatorial deep reinforcement learning for age-optimal joint trajectory and scheduling design in UAV-assisted networks,” IEEE J. Sel. Areas Commun., vol. 39, no. 5, pp. 1250–1265, May 2021.

B. Yin, S. Zhang, and Y. Cheng, “Application-oriented scheduling for optimizing the age of correlated information: A deep-reinforcement-learning-based approach,” IEEE Internet Things J., vol. 7, no. 9, pp. 8748–8759, Sep. 2020.

O. Ayan, H. M. Gürsu, S. Hırche, and W. Kellerer, “Aol-based finite horizon scheduling for heterogeneous networked control systems,” in Proc. IEEE Global Commun. Conf., Dec. 2020, pp. 1–7.

B. Sombabu and S. Moharir, “Age-of-information based scheduling for multi-channel systems,” IEEE Trans. Wireless Commun., vol. 19, no. 7, pp. 4439–4448, Jul. 2020.

Z. Bao, Y. Dong, Z. Chen, P. Fan, and K. B. Letaief, “Age-optimal service and decision processes in Internet of Things,” IEEE Internet Things J., vol. 8, no. 4, pp. 2826–2841, Feb. 2021.

S. Feng and J. Yang, “Precoding and scheduling for AoI minimization in MIMO broadcast channels,” IEEE Trans. Inf. Theory, vol. 68, no. 8, pp. 5185–5202, Aug. 2022.

M. A. Abd-Elmajid and H. S. Dhillon, “Average peak age-of-information minimization in UAV-assisted IoT networks,” IEEE Trans. Veh. Technol., vol. 68, no. 2, pp. 2003–2008, Feb. 2019.

S. F. Abedin, Md. S. Munir, N. H. Tran, Z. Han, and C. S. Hong, “Data freshness and energy-efficient UAV navigation optimization: A deep reinforcement learning approach,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 9, pp. 5994–6006, Sep. 2021.

F. Wu, H. Zhang, J. Wu, Z. Han, H. V. Poor, and L. Song, “UAV-to-Device underlay communications: Age of information minimization by multi-agent deep reinforcement learning,” IEEE Trans. Commun., vol. 69, no. 7, pp. 4461–4475, Jul. 2021.

B. Zhou and W. Saad, “On the age of information in Internet of Things systems with correlated devices,” in Proc. GLOBECOM - IEEE Global Commun. Conf., Dec. 2020, pp. 1–6.

S. Farazi, A. G. Klein, and D. R. Brown, “Age of information with unreliable transmissions in multi-source multi-hop status update systems,” in Proc. 53rd Asilomar Conf. Signals, Syst., Comput., Nov. 2019, pp. 2017–2021.

A. M. Bedewy, Y. Sun, and N. B. Shroff, “The age of information in multi-hop networks,” IEEE/ACM Trans. Netw., vol. 27, no. 3, pp. 1248–1257, Jun. 2019.

B. Soret, S. Ravikanti, and P. Popovski, “Latency and timeliness in multi-hop satellite networks,” in Proc. IEEE Int. Conf. Commun. (ICC), Aug. 2020, pp. 1–6.

R. Talak, S. Karaman, and E. Modiano, “Minimizing age-of-information in multi-hop wireless networks,” in Proc. 55th Allerton Conf. Commun., Control, Comput. (Allerton), Oct. 2017, pp. 486–493.

M. A. Abd-Elmajid and H. S. Dhillon, “Joint distribution of ages of information in networks,” IEEE Trans. Inf. Theory, vol. 69, no. 9, pp. 5701–5722, Sep. 2023, doi: 10.1109/TIT.2023.3282578.

T.-W. Kuo, “Competitive analyses of online minimum age of information transmission scheduling,” in Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS), May 2022, pp. 1–8.

A. Maatouk, M. Assaad, and A. Ephremides, “Age-aware stochastic hybrid systems: Stability, solutions, and applications,” IEEE J. Sel. Areas Inf. Theory, vol. 4, pp. 762–783, 2023.

N. Akar and S. Ulukus, “Age of information in a single-source generate-at-will dual-server status update system,” 2024, arXiv:2404.01229.

M. A. Abd-Elmajid and H. S. Dhillon, “Closed-form characterization of the MGF of AoI in energy harvesting status update systems,” IEEE Trans. Inf. Theory, vol. 68, no. 6, pp. 3896–3919, Jun. 2022.

J. P. Hespanha, “Modelling and analysis of stochastic hybrid systems,” in Proc. Conf. Decision Control, pp. 502–535, Sep. 2006.

M. A. Davie and A. J. Stothers, “Improved bound for complexity of matrix multiplication,” Proc. Roy. Soc. Edinburgh: Sect. A Math., vol. 143, no. 2, pp. 351–369, Apr. 2013.