Towards Assigning Priorities in Queues Using Age of Information

Jin Xu∗ Natarajan Gautam†

Abstract

We consider a priority queueing system where a single processor serves \( k \) classes of packets that are generated randomly following Poisson processes. Our objective is to compute the expected Peak Age of Information (PAoI) under various scenarios. In particular, we consider two situations where the buffer sizes are one and infinite, and in the infinite buffer size case we consider First Come First Serve (FCFS) and Last Come First Serve (LCFS) as service disciplines. We derive PAoI exactly for the exponential service time case and bounds (which are excellent approximations) for the general service time case. Using those we suggest optimal ordering of priorities for the various scenarios. We perform extensive numerical studies to validate our results and develop insights.

Keywords— Age of Information, Priority Queues, Performance Analysis

1 Introduction

In the recent years the notion of Age of Information (AoI) has garnered attention from several researchers. The main applications that have been cited include sensor networks, wireless networks and autonomous vehicle systems \[12\], as in all those cases it is important to know the freshness of information. Our research has been motivated by an application in smart manufacturing of the future where edge devices, sensors in particular, with limited processing capabilities, would monitor the health of various tools, condition of components and quality of work pieces in machines. This sensed information would be used to make real-time decisions such as tool changes, re-calibration and rework, thereby improving overall cost and quality of the manufactured products. Hence it is crucial to consider the freshness of information to make these decisions, for some type of which AoI is an ideal choice.

AoI is a metric defined and used by researchers such as Kaul et al \[12\] to describe the freshness of data. We consider a system where a data source (sensor or resource) from time to time sends updates or files (in this paper we call each update or file a “packet”) to the processor (also called server). The time when the data source generates a packet can also be regarded as the arrival time (also called release time) of the packet into the system. The server processes packets in a non-preemptive way. Unprocessed packets are queued due to the limited processing capacity of the server. AoI at an arbitrary time point \( t \) is defined as the length of period between time \( t \) and the most recent release time among all the packets that have been processed. Mathematically, we define the AoI at time \( t \) as \( \triangle(t) = t - \max\{r_1 : C_1 \leq t\} \), where \( r_1 \) is the release time of the \( 1^{\text{st}} \) packet and \( C_1 \) is the time when it is processed by the server (also called completion time). While the time-average AoI could be a metric to measure data freshness, many researchers consider Peak Age of Information (PAoI) as a more useful and tractable metric \[9, 25\]. We let the \( n^{\text{th}} \) peak value of \( \triangle(t) \) be \( A_n \),

∗Email: jinxu@tamu.edu, Industrial and Systems Engineering, Texas A&M University
†Corresponding author, Email: gautam@tamu.edu, Industrial and Systems Engineering, Texas A&M University
which is a random variable and it is shown in Figure 1. The expectation of this peak value, i.e., $E[A_n]$, is then defined as PAoI for this data source. Next we extend this notion to multiple sources and formulate our model.

![Figure 1: Age of Information for a Single Queue](image)

It has been well documented and accepted that monitoring and sensing according to a Poisson process is effective [17]. In that light we consider multiple data sources (sensors) that monitor according to a Poisson process with potentially different rates due to the difficulty in sensing (recall our motivation example of a manufacturing setting). Also, not all streams of packets have the same priorities, and we consider a setting where there are $k$ data sources prioritized from 1 (highest) to $k$ (lowest). There is a single processor (server) that “serves” the $k$ packet streams based on a static priority mechanism. The system model is provided in Figure 2. We consider two settings in this paper: one in which there is buffer for each data source that can hold only one packet at a time, and the buffer only holds the packet that arrives most recently; another where each buffer can hold infinitely many packets. The first scenarios is ideal if the objective is only to perform real-time decisions, while the latter is useful if we also wish to not drop any packets and keep entire data streams for offline diagnostics. Our objective is to obtain the PAoI for each class of sensed information under each scenario assuming a general distribution of service time for packets.

This system is modeled as a multi-class multiple parallel queueing system with static priorities. Such a system for computing PAoI has not been studied in the literature. A summary of the literature is provided in Section 2. Then, in Section 3 we provide the PAoI analysis for M/G/1/1+1* type queues, where arrivals are Poisson and service time for packets are iid and generally distributed. The notation 1 + 1* means besides the processing area at the server, each data source has a buffer with size one. The asterisk means that the packet waiting in the buffer is replaced by the newest arrival, the same as the notation used in [7]. In Section 4 we provide the PAoI analysis for queues with infinite buffer size, under both First Come First Serve (FCFS) and Last Come First Serve (LCFS) disciplines within each queue. We perform numerical studies in Section 5 and make concluding remarks as well as discuss the future work in Section 6.
2 Related Work

The idea of data age, freshness and timeliness for data warehouses are introduced and discussed in [23, 5]. In recent years, data freshness has drawn much more attention because of the development of Internet of Things, fog computing and edge data storage [2, 24]. Kaul et al [12] firstly provided average AoI for M/M/1, M/D/1 and D/M/1 type queues. Costa et al [7] obtained analytical results of average AoI and PAoI under FCFS for M/M/1/1, M/M/1/2 (which allows drop of new arrivals) as well as M/M/1/2* (which allows update for the waiting packet) queues. The performance of LCFS policy for the single queue case where service times are gamma distributed was provided by Najm and Nasser [18]. Some recent works have considered AoI for single server with multiple queues. Huang and Modiano [9] provided the PAoI for multi-class M/G/1 and M/G/1/1 queues where all packets flow into a combined queue. Najm and Telatar [19] considered the M/G/1/1 system with multiple sources updating while allowing preemption. The multi-class queues with FCFS and LCFS across queues are discussed in Yates and Kaul [25]. A detailed review for the current literature for AoI is also provided in [25].

However, we notice that if FCFS or LCFS across queues is adopted in the multi-queue system, queues with high arrival rates will be served more frequently. It is not always the case that the queues with higher arrival rates are more important. Queues with low traffic intensities may also be important, and their packets may need to be processed as soon as they enter the queues. Besides, spending too much time processing a certain data source is a waste of service power. We thus want to consider a service policy which gives certain queues higher priorities. Such a multiple-queue system with queue priorities has been studied for a long time, however they are for different metrics such as queue lengths and waiting time distributions [10, 1]. AoI and PAoI are metrics introduced in recent years, and their performance under queue priorities are not well understood. Recently, Kaul and Yates [13] modeled the AoI of M/M/1 priority queues as a hybrid system by assuming the waiting room (buffer size) for all queues is either null or one. However, their model is restrictive since there is at most one buffer for all queues. It is also believed that the model with each queue having an independent buffer is more efficient but complicated [13]. Maatouk et al [15] discussed the model where each queue has an individual buffer, and provided the closed form of AoI using a hybrid system analysis. However, it is assumed in [15] that arrival and service rates for all queues are exponential and identical. In our work, we for the first time provide the exact PAoI for the system where queues are prioritized and each queue has its independent waiting room (buffer), arrival rate and service rate. We provide a new approach of calculating age-related metrics by focusing on the buffer state, and derive the PAoI for M/M/1 and M/G/1
queues with priorities, for both the cases where each queue has one and infinite sized buffers. Also, for the infinite buffer size case, we derive the PAoI for both FCFS and LCFS service disciplines within a queue. We seek to find a priority order that would result in low average PAoI across queues, which is more efficient than the simple FCFS or LCFS across queues that is introduced in [25]. We also seek to understand the effect of arrival rates and service times on the PAoI for our systems.

3 Queues with Buffer Size One

In this section we mainly discuss a system in which the buffer size for each queue is one, and the arrival process for each queue $i$ is a Poisson process with rate $\lambda_i$. The service time (processing time) $P_i$ for packets from queue $i$ is iid with cdf $F_i(x)$ and mean $\frac{1}{\mu_i}$. A new arrival will replace the packet waiting in the queue (if there is one) since the newest packet contains the most recent information of the source. Note that this model is different from the M/G/1/1 model introduced in [9] [19]. In their model, there is no buffer for each queue, so whenever a packet arrives and sees the server being busy, the packet is either rejected or preempts the packet in service. Further, when the server becomes available, it has to wait until the next packet arrives. In our model, the buffer allows the server to serve packets whenever the server becomes available, which is potentially more efficient by not waiting for the next packet. Moreover, only keeping the most recent packet in the buffer can potentially reduce the server’s load, and also keep the most recent information from each source.

The difficulty in analyzing such a system with waiting room for one packet in each queue is that packets entering the system are only a subset of packets generated by the data source, due to some getting rejected. Focusing on how each packet goes through the system often makes modeling more complicated [13]. Instead, in our model we introduce a new way of modeling such systems, which is to incorporate the buffer state. Note that we can also use this idea to derive PAoI for other systems with buffer size more than one, as we will see in Section 4. In this section we only consider the model with buffer size of one for each queue, we now show how this buffer size of one helps us characterize PAoI. We depict a sample path of the buffer state for queue $i$ in Figure 3 with notations described subsequently. From Figure 3 we can see that buffer state of queue $i$ is either 0 or 1. When the buffer state is 1 (the buffer is full), we say the buffer is busy. At time $r_{i1}$ packet 1 arrives. It waits until time $S_{i1}$ when the server becomes available to serve it by removing the packet from the buffer and placing it in the processing area. Right after time $S_{i1}$ buffer $i$ becomes empty until packet 2 arrives at time $r_{i2}$. Packet 2 stays in the buffer for while, then gets replaced by packet 3 at time $r_{i3}$. Packet 3 is then replaced by packet 4 at time $r_{i4}$. At time $S_{i4}$ the server becomes available and starts serving packet 4, and the buffer gets empty again. The service of packet 4 is completed at time $C_{i4}$, and the peak age of information upon the completion of packet 4 is given as $C_{i4} - r_{i1}$, which is equal to

$$C_{i4} - r_{i1} = (C_{i4} - S_{i4}) + (S_{i4} - r_{i2}) + (r_{i2} - S_{i1}) + (S_{i1} - r_{i1}).$$

The term $(C_{i4} - S_{i4})$ of Equation (1) is the processing time of packet 4, and $(S_{i4} - r_{i2})$ is the time period during which the buffer has one packet. The third term $(r_{i2} - S_{i1})$ is the time period during which the buffer stays empty, and the last term $(S_{i1} - r_{i1})$ is the gap between start of service and the release time of the most recent arrival. Recall that the processing time of packets from the same source is iid, so the expected value of $(C_{i4} - S_{i4})$ is $E[P_i] = \frac{1}{\mu_i}$. The buffer is empty during time $(r_{i2} - S_{i1})$, and we know that there is no arrival in $(r_{i1}, S_{i1})$. Using the memoryless property of exponential inter-arrival times, the expected time
of buffer staying empty is the expected inter-arrival time, $E[I_i] = \frac{1}{\lambda_i}$. Therefore we can write the PAoI for source $i$ as

$$E[A_i] = E[P_i] + E[W_i] + E[I_i] + E[G_i], \quad (2)$$

where $E[G_i]$ is the expected gap from the release time of the most recent arrival to the time when buffer becomes empty, and $E[W_i]$ is the expected length of time period when the buffer is continuously occupied (busy). Note that Equation (2) holds true for every queue $i$. For M/G/1 type queues, we have already stated that $E[P_i] = \frac{1}{\mu_i}$ and $E[I_i] = \frac{1}{\lambda_i}$. The difficult part remains in calculating $E[W_i]$ and $E[G_i]$. From Figure 3 we observe that if we reject new arrivals (instead of new arrivals replacing ones in the buffer) when the buffer is full, $W_i$ is not changed. If we reject the most recent arrival (instead of the system that we are analyzing) when the buffer is full, then $W_i$ is the waiting time for the packet that enters the buffer, which equals to $W_i$ if we keep the most recent arrival. Using this property, if we let $p_i$ be the probability that buffer $i$ is full, then from Little’s Law we know the average queue length is $p_i = \lambda_i(1 - p_i)E[W_i]$. So we have

$$E[W_i] = \frac{p_i}{\lambda_i(1 - p_i)}. \quad (3)$$

From Equation (3), $E[W_i]$ can be obtained once we know $p_i$. We shall discuss how to find $p_i$ later in this section. Now we continue with the system where new arrivals replace the existing ones in queue. We first characterize $G_i$, which depends on $W_i$, as we will see in Lemma 1.

**Lemma 1.** $E[G_i | W_i = t] = \frac{1}{\lambda_i}(1 - e^{-\lambda_i t})$.

**Proof.** Suppose there are $N(t) = m$ packets arriving during $W_i$, then $G_i$ is the time gap from the release time of the $m^{th}$ packet $R_m$ to time $t$. From Campbell’s Theorem (P173, Theorem 5.14 in [15]) we have
\[ P(G_i < x | N(t) = m, W_i = t) \]
\[ = P(t - R_m < x | N(t) = m, W_i = t) \]
\[ = P(R_m > t - x | N(t) = m, W_i = t) \]
\[ = \int_{t-x}^{t} \frac{m}{t} \left( \frac{u}{t} \right)^{m-1} du \]
\[ = \left( \frac{u}{t} \right)^{m} |_{t-x}^{t} \]
\[ = 1 - \left( \frac{t-x}{t} \right)^{m}. \]

Thus by integrating \( P(G_i > x | N(t) = m, W_i = t) \) for \( x \) from 0 to \( t \), we have

\[ E[G_i | N(t) = m, W_i = t] = \frac{t}{m+1}. \]

Then, unconditioning using \( P(N(t) = m) = e^{-\lambda_i t} \left( \frac{\lambda_i t}{m!} \right)^m \), we get

\[ E[G_i | W_i = t] = \sum_{m=0}^{\infty} \frac{t}{m+1} e^{-\lambda_i t} \left( \frac{\lambda_i t}{m!} \right)^m \]
\[ = \sum_{m=0}^{\infty} e^{-\lambda_i t} \left( \frac{\lambda_i t}{m+1} \right)^{m+1} \frac{1}{m!} \lambda_i \]
\[ = \frac{e^{-\lambda_i t}}{\lambda_i} (e^{\lambda_i t} - 1). \]

Lemma [1] shows that one needs to know the distribution of \( W_i \) or its Laplace–Stieltjes transform (LST) to get \( E[G_i] \). The exact LST of \( W_i \) can be obtained when service times are exponentially distributed, as we will see in Section 3.1. If service times are generally distributed, we provide the bounds for PAoI based on result of Lemma [1], which we will see in Section 3.2.

### 3.1 Exact Analysis for M/M/1/1+Σ 1* Type Queues

In this subsection we consider a special case where the processing time \( P_i \) is \( \text{exp}(\mu_i) \) for all \( i \) and discuss how to calculate \( E[W_i] \) and \( E[G_i] \). Knowing the LST of \( W_i \) can help us obtain both \( E[W_i] \) and \( E[G_i] \), so in this subsection we focus on calculating LST of \( W_i \). Since \( W_i \) is not affected by which packet we reject when the buffer is full, in this subsection, we assume that we reject the most recent arrivals. We adopt the method used to characterize the busy period in [6] to characterize the LST of \( W_i \), i.e., \( E[e^{-sW_i}] \). Let \( B_i(t) \) be the number of priority \( i \) packets in buffer \( i \) at time \( t \), \( B_i(t) \in \{0, 1\} \). Let \( J(t) \in \{0, 1, ..., k\} \) be the packet that is in service at time \( t \), where \( J(t) = 0 \) means the server is idling. The vector \( S(t) = (J(t), B_1(t), ..., B_k(t)) \) thus indicates the state of the system at time \( t \). Obtaining the stationary state seen by packets that enter the system (which are not all the arrivals) is crucial for our analysis, so in the following we introduce an approach to find its stationary probability. From PASTA [13] we know that the time average performance of the system is the same as that seen by Poisson arrivals. If a packet from class \( i \) sees \( B_i(t) = 0 \), it then enters the buffer if the server is busy, or enters the server directly if the server is idling. Thus the state that
we have source 2 enters, then different scenarios observed by the packets that enter buffer 2. If the server is idling when a packet from

the remaining service time observed by a packet entering queue time for packets from queue

another busy period will start from time

This packet is of length

Before characterizing \( E[e^{-sW_2}] \) for buffer 2, we first introduce the busy period of the server. Let \( T_1 \) be the time period that the server is continuously busy processing packets from buffer 1, and \( \eta_1(s) = E[e^{-sT_1}] \). The busy period \( T_1 \) always starts from processing a packet from buffer 1. Suppose the processing time of this packet is of length \( P_1 = l \) and if there is more than one priority 1 packet arriving during \([0, l] \), then another busy period will start from time \( l \) and the new busy period is identically distributed as \( T_1 \). Thus we have \( E[e^{-s(l+T_1)}] P_1 = l, B_1(l) = 1 = e^{-sl} \eta_1(s) \).

If there is no arrival then the busy period would be \( l \) only, then by unconditioning on \( B_1(l) \) we have

\[
E[e^{-s(l+T_1)}] P_1 = l = e^{-sl} \eta_1(s)(1 - e^{-\lambda_1 l}) + e^{-sl} e^{-\lambda_1 l}.
\]

Unconditioning on \( P_1 = l \) we have

Thus the LST of \( T_1 \) is given by \( \eta_1(s) = \frac{\psi_1(s + \lambda_1)}{1 - \psi_1(s) + \psi_1(s + \lambda_2)} = \frac{\mu_1(s + \mu_1)}{s^2 + 2\mu_1 s + s^2 + \mu_1^2} \) and the derivative of \( \eta_1(s) \) at \( s = 0 \) is given by \( \eta_1'(s) |_{s=0} = \frac{-\lambda_1 - \mu_1}{\mu_1^2} \).

Now we characterize the LST of \( W_2 \) by the fact that \( E[e^{-sW_2}] = E[e^{-s(U_2+T_1)}] \) and conditioning on different scenarios observed by the packets that enter buffer 2. If the server is idling when a packet from source 2 enters, then

\[
E[e^{-s(U_2+T_1)}] B_1(0) = 0, J(0) = 0, B_2(0) = 0 = 1.
\]

If the server is busy processing a packet from buffer \( j \) for \( j \in \{1, \ldots, k\} \), and buffer 1 is not empty, then we have

\[
E[e^{-s(U_2+T_1)}] B_1(0) = 1, J(0) = j, B_2(0) = 0
\]

If the server is busy processing a packet from buffer \( j \) for \( j \in \{1, \ldots, k\} \), and buffer 1 is empty, then we have

\[
E[e^{-s(U_2+T_1)}] B_1(0) = 0, J(0) = j, U_2 = u, B_1(u) = 1, B_2(0) = 0 = e^{-su} E[e^{-sT_1}] = e^{-s\mu_1} \eta_1(s), \quad \text{and}
\]

\[
E[e^{-s(U_2+T_1)}] B_1(0) = 0, J(0) = j, U_2 = u, B_1(u) = 0, B_2(0) = 0 = e^{-su}.
\]

an entering packet from source \( i \) observes is always \( B_i(t) = 0 \). We let \( \psi_j(s) = \frac{\mu_i}{\mu_i + s} \) be the LST of service time for packets from queue \( j \). Because the service time is exponential, \( \psi_j(s) \) is also the LST of remaining service time of the packet observed by an entering packet, if the packet in service is from queue \( j \). Let \( U_i \) be the remaining service time observed by a packet entering queue \( i \). If we assume that the system starts from time 0, then we have for queue 1 that

\[
E[e^{-sW_1}|B_1(0) = 0] = E[e^{-sU_1}|B_1(0) = 0]
\]

\[
= P(J(0) = 0|B_1(0) = 0) + \sum_{j=1}^k \psi_j(s) P(J(0) = j|B_1(0) = 0).
\]
By unconditioning on $B_1(u)$ we have

$$E[e^{-s(U_2+T_1)}|B_1(0) = 0, J(0) = j, U_2 = u, B_2(0) = 0]$$

$$= e^{-su}\eta_1(s)(1 - e^{-\lambda_1 u}) + e^{-su}e^{-\lambda_1 u}.$$ 

By unconditioning on $U_2 = u$ we have

$$E[e^{-s(U_2+T_1)}|B_1(0) = 0, J(0) = j, B_2(0) = 0]$$

$$= \psi_j(s)\eta_1(s) - \psi_j(s + \lambda_1)\eta_1(s) + \psi_j(s + \lambda_1).$$

So far we have characterized the LST of $W_2$ conditioning on different scenarios. We only need probabilities of $P(B_1(0) = 0, 1, J(0) = j | B_2(0) = 0)$ to obtain $E[e^{-sW_2}]$, which we will discuss at the end of this subsection. Before doing that, we now consider how to obtain LST of $W_3$ by conditioning on different scenarios. For simplicity of analysis we here assume $\lambda_1 = \lambda_2$ and $\mu_1 = \mu_2$. The argument for distinct $\lambda_1$ and $\lambda_2$ or $\mu_1$ and $\mu_2$ are similar, however notationally cumbersome. We let $T_{12}$ be the busy time during which the server continuously serves packets from queue 1 and queue 2 and let $B_{12}(t) = B_1(t) + B_2(t)$. We now characterize the LST of $T_{12}$ by letting $\eta_{12,0}(s) = E[e^{-sT_{12}} | B_{12}(0) = 0]$ and $\eta_{12,1}(s) = E[e^{-sT_{12}} | B_{12}(0) = 1]$.

Since the busy time $T_{12}$ always starts with processing either a packet from source 1 or 2, we suppose the busy period starts with processing a packet with processing time $P_1 = l$. We then have

$$E[e^{-s(l+T_{12})}|B_{12}(0) = 0, P_1 = l, B_{12}(l) = 0, B_3(0) = 0] = e^{-sl},$$

$$E[e^{-s(l+T_{12})}|B_{12}(0) = 0, P_1 = l, B_{12}(l) = 1, B_3(0) = 0] = e^{-sl}\eta_{12,0}(s),$$

$$E[e^{-s(l+T_{12})}|B_{12}(0) = 0, P_1 = l, B_{12}(l) = 2, B_3(0) = 0] = e^{-sl}\eta_{12,1}(s),$$

$$E[e^{-s(l+T_{12})}|B_{12}(0) = 1, P_1 = l, B_{12}(l) = 1, B_3(0) = 0] = e^{-sl}\eta_{12,0}(s),$$

$$E[e^{-s(l+T_{12})}|B_{12}(0) = 1, P_1 = l, B_{12}(l) = 2, B_3(0) = 0] = e^{-sl}\eta_{12,1}(s).$$

Note that $B_{12}(0) = 2$ has probability 0 since the busy period $T_{12}$ always starts with processing a packet from either buffer 1 or 2. Unconditioning on $B_{12}(l)$, we have

$$E[e^{-s(l+T_{12})}|B_{12}(0) = 0, P_1 = l, B_3(0) = 0]$$

$$= e^{-sl}e^{-2\lambda_1 l} + 2(1 - e^{-\lambda_1 l})e^{-\lambda_1 l}e^{-sl}\eta_{12,0}(s)$$

$$+ e^{-sl}(1 - e^{-\lambda_1 l})^2\eta_{12,1}(s),$$

and

$$E[e^{-s(l+T_{12})}|B_{12}(0) = 1, P_1 = l, B_3(0) = 0]$$

$$= e^{-sl}e^{-\lambda_1 l}\eta_{12,0}(s) + e^{-sl}(1 - e^{-\lambda_1 l})\eta_{12,1}(s).$$

Unconditioning on $P_1 = l$, we have

$$\eta_{12,0}(s) = \psi_1(s + 2\lambda_1) + 2[\psi_1(s + \lambda_1) - \psi_1(s + 2\lambda_1)]\eta_{12,0}(s)$$

$$+ [\psi_1(s) - 2\psi_1(s + \lambda_1) + \psi_1(s + 2\lambda_1)]\eta_{12,1}(s).$$
and

\[ \eta_{12,1}(s) = \psi_1(s + \lambda_1)\eta_{12,0}(s) + [\psi_1(s) - \psi_1(s + \lambda_1)]\eta_{12,1}(s). \]

By solving the two equations above for \( \eta_{12,0}(s) \) and \( \eta_{12,1}(s) \), we have

\[ \eta_{12,0}(s) = \frac{\psi_1(s + 2\lambda_1)}{1 - 2[\psi_1(s + \lambda_1) - \psi_1(s + 2\lambda_1)] - \frac{\psi_1(s + \lambda_1)}{1 - \psi_1(s + \lambda_1)}[\psi_1(s) - 2\psi_1(s + \lambda_1) + \psi_1(s + 2\lambda_1)]}, \]

and

\[ \eta_{12,1}(s) = \frac{\eta_{12,0}(s)\psi_1(s + \lambda_1)}{1 - \psi_1(s + \lambda_1)}. \]

Recall that \( U_3 \) is the remaining service time observed by a packet that enters buffer 3, we then have the LST of busy period of buffer 3 as conditioned on various scenarios:

\[ E[e^{-s(U_3 + T_{12})}|B_{12}(0) = 0, J(0) = 0, B_3(0) = 0] = 1, \]

\[ E[e^{-s(U_3 + T_{12})}|B_{12}(0) = 0, J(0) = j, B_3(0) = 0] = \psi_j(s + 2\lambda_1)\eta_{12,0}(s) + [\psi_j(s) - \psi_j(s + \lambda_1)]\eta_{12,1}(s), \]

\[ E[e^{-s(U_3 + T_{12})}|B_{12}(0) = 1, J(0) = j, B_3(0) = 0] = \psi_j(s + \lambda_1)\eta_{12,0}(s) + [\psi_j(s) - \psi_j(s + \lambda_1)]\eta_{12,1}(s), \]

and

\[ E[e^{-s(U_3 + T_{12})}|B_{12}(0) = 2, J(0) = j, B_3(0) = 0] = \psi_j(s)\eta_{12,1}(s). \]

Thus we can characterize the LST of \( W_3 \) once we know the stationary probability of each scenario. For queues with lower priorities, the analysis requires more argument, but they are all similar (albeit cumbersome notationally). To get the stationary probability of each scenario, we model \( S(t) = (J(t), B_1(t), B_2(t), ..., B_k(t)) \) as a CTMC and obtain the stationary probabilities. Here we only show the example for the case of \( k = 2 \), for \( k > 2 \) the analysis is similar. The rate matrix \( Q \) of the two-queue case is:
arrivals still follow Poisson processes, Lemma 1 holds. We can write the PAoI of queue entering packet, thus the analysis in Subsection 3.1 does not hold for M/G/1 type queues. However, since the system is an M/G/1/1 system with each queue having a unique buffer size. Our system thus becomes a special case of the model in Takenaka [21], and in our model each queue has a buffer with size one. Takenaka [21] considers a multi-queue M/G/1 system with each queue having a unique buffer size. Our system thus becomes a special case of the model in Takenaka [21], and in our model each queue has a buffer with size one. Takenaka [21] introduces the relationship of \( p_i \) with the stationary state that is seen by departures, for the system in which service times for packets from different queues are identically distributed with \( F_i(x) = F(x) \) and \( \mu_i = \mu \) for all \( i \). Thus one can get the stationary distribution of states by solving an

### 3.2 Bounds and Approximation for M/G/1/1+\( \sum \) 1 Type Queues

Here we generalize the analysis in Subsection 3.1 so that the service times are generally distributed, hence the system is an M/G/1/1+\( \sum \) 1 Type system. Without the assumption that the service time is exponentially distributed, the remaining service time observed at an arbitrary time is no longer what is observed by an entering packet, thus the analysis in Subsection 3.1 does not hold for M/G/1 type queues. However, since arrivals still follow Poisson processes, Lemma 1 holds. We can write the PAoI of queue \( i \) as

\[
E[A_i] = E[P_i] + E[W_i] + E[I_i] + E[G_i] \\
= \frac{1}{\mu_i} + \frac{p_i}{\lambda_i(1 - p_i)} + \frac{2}{\lambda_i} \frac{1}{E[e^{-\lambda_i W_i}]} \\
\leq \frac{1}{\mu_i} + \frac{p_i}{\lambda_i(1 - p_i)} + \frac{2}{\lambda_i} \frac{1}{e^{-1}} \\
(4)
\]

The last inequality of (4) follows from the Jensen’s inequality by knowing that \( e^{-\lambda_i x} \) is a convex function. Notice that Equation (4) gives an upper bound of PAoI in terms of rejection probability \( p_i \). Takenaka [21] considered a multi-queue M/G/1 system with each queue having a unique buffer size. Our system thus becomes a special case of the model in Takenaka [21], and in our model each queue has a buffer with size one. Takenaka [21] introduces the relationship of \( p_i \) with the stationary state that is seen by departures, for the system in which service times for packets from different queues are identically distributed with \( F_i(x) = F(x) \) and \( \mu_i = \mu \) for all \( i \). Thus one can get the stationary distribution of states by solving an

\[
\begin{bmatrix}
(0.0.0) & (1.0.0) & (2.0.0) & (1.1.0) & (2.1.0) & (2.0.1) & (1.1.1) & (2.1.1)
\end{bmatrix}
\begin{bmatrix}
(0.0.0) & -\lambda_1 - \lambda_2 \\
(1.0.0) & \mu_1 & -\lambda_1 - \lambda_2 - \mu_1 & 0 & \lambda_1 & 0 & 0 & 0 \\
(2.0.0) & 0 & 0 & -\lambda_1 - \lambda_2 - \mu_2 & 0 & 0 & \lambda_1 & 0 & 0 \\
(1.1.0) & 0 & \mu_1 & 0 & -\lambda_1 - \mu_1 & 0 & \lambda_1 & 0 & 0 \\
(2.1.0) & 0 & 0 & \mu_2 & 0 & 0 & 0 & 0 & -\lambda_1 - \mu_2 & 0 & \lambda_1 \\
(2.0.1) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
(1.1.1) & 0 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & -\mu_1 & 0 \\
(2.1.1) & 0 & 0 & 0 & 0 & \mu_2 & 0 & 0 & 0 & 0 & -\mu_2 \\
\end{bmatrix}
\]

The stationary distribution \( \hat{\pi} \) (which is a vector) is given by solving \( \hat{\pi}Q = 0 \) and \( \hat{\pi}1 = 1 \), and we have

\[
p_1 = \hat{\pi}(1, 1, 0) + \hat{\pi}(2, 1, 0) + \hat{\pi}(1, 1, 1) + \hat{\pi}(2, 1, 1),
\]

\[
p_2 = \hat{\pi}(1, 0, 1) + \hat{\pi}(2, 0, 1) + \hat{\pi}(1, 1, 1) + \hat{\pi}(2, 1, 1),
\]

\[
P(J(0) = 1|B_1(0) = 0) = \frac{\hat{\pi}(1, 0, 0) + \hat{\pi}(1, 0, 1)}{1 - p_1},
\]

and

\[
P(B_1(0) = 0, J(0) = 1|B_2(0) = 0) = \frac{\hat{\pi}(1, 0, 0)}{1 - p_2}.
\]
embedded Markov chain. It is important to note that the result in [21] only works for identically distributed service times. For heterogeneous service times with \( k > 2 \), the results are difficult to obtain [20, 21]. So till the end of this subsection, we assume that service times for packets across queues are identically distributed. To use the result in [21] to get \( p_i \)'s, we first introduce some notations here. Let \( \psi(s) \) be the LST of service time. Let \( S_k \) be our original system which has \( k \) queues. Say \( S_l \) is the subsystem of \( S_k \) which contains only queue 1 to queue \( l \), and packets from queue \( l + 1 \) to \( k \) do not arrive in system \( S_l \). Let \( \pi_i(B_1, B_2, \ldots, B_l) \) be the stationary distribution in which the system \( S_l \) has \( B_i \in \{0, 1\} \) number of packets in queue \( i \) immediately after the departure of a packet. Now we re-write a theorem from [21] for our model.

**Theorem 2.** (Theorem 3 of [21]) The rejection probability of queue \( i \) if the buffer size of each queue is one, is given by

\[
p_i = 1 - \frac{\pi_{i+1}(0, \ldots, 0) - \pi_i(0, \ldots, 0)}{\pi_i(0, \ldots, 0)} = \frac{\psi(1) - \psi(0)}{\psi(1)} \text{ for all } i \in \{1, 2, \ldots, k\}, \text{ where } \pi_0(0, \ldots, 0) = 1.
\]

To obtain the rejection probability, we only need to find the stationary distribution that is seen by departures. For that, we model the system state seen by departures as an embedded Markov chain. We only introduce the case for \( k = 2 \) here. For \( k > 2 \) the analysis is similar but not presented here for notational and space restrictions. Since the departure can see at most one packet waiting at each buffer, the transition matrix for \( k = 2 \) is given as follows:

\[
\tilde{P}_2 = \begin{bmatrix}
(0,0) & (0,1) & (1,0) & (1,1) \\
(0,0) & a_0 & a_1 & a_2 & a_3 \\
(0,1) & a_0 & a_1 & a_2 & a_3 \\
(1,0) & a_0 & a_1 & a_2 & a_3 \\
(1,1) & 0 & b_0 & 0 & 1 - b_0
\end{bmatrix},
\]

where \( a_0 = \int_0^\infty e^{-\lambda_1 x} dF(x) \), \( a_1 = \int_0^\infty e^{-\lambda_1 x}(1 - e^{-\lambda_2 x}) dF(x) \), \( a_2 = \int_0^\infty (1 - e^{-\lambda_1 x}) e^{\lambda_2 x} dF(x) \), \( a_3 = \int_0^\infty (1 - e^{-\lambda_1 x})(1 - e^{-\lambda_2 x}) dF(x) \), and \( b_0 = \int_0^\infty e^{-\lambda_1 x} dF(x) \). The stationary distribution \( \pi_2(0, 0) \) can thus be obtained by solving the linear system \( \pi_2 \tilde{P}_2 = \pi_2 \) with \( \pi_2(1, 1) = 1 \), where \( \pi_2 = (\pi_2(0, 0), \pi_2(0, 1), \pi_2(1, 0), \pi_2(1, 1)) \).

Notice from Theorem 2 that we also need \( \pi_1(0) \) to get \( p_i \)'s. To obtain \( \pi_1(0) \) we solve the subsystem \( S_l \) with \( \pi_1 \tilde{P}_1 = \pi_1 \) and \( \pi_1(1, 1) = 1 \), where the transition matrix \( \tilde{P}_1 \) of the embedded Markov chain is given by

\[
\tilde{P}_1 = \begin{bmatrix}
(0) & (1) \\
(0) & b_0 & 1 - b_0 \\
(1) & 0 & 1 - b_0
\end{bmatrix}.
\]

By solving the embedded Markov chains we have \( \pi_1(0) = \psi(\lambda_1) \) and \( \pi_2(0, 0) = \frac{\psi(\lambda_1 + \lambda_2) \psi(\lambda_1)}{\psi(\lambda_1) + \psi(\lambda_1 + \lambda_2)} \). Then using Theorem 2 we have \( p_1 = 1 - \frac{1 - \psi(\lambda_1)}{\lambda_1 + \frac{1}{\lambda_1} + \frac{1}{\lambda_1} + \psi(\lambda_1) - \pi_2(0, 0)} \) and \( p_2 = 1 - \frac{\psi(\lambda_1 + \lambda_2) \psi(\lambda_1)}{\psi(\lambda_1) + \psi(\lambda_1 + \lambda_2) - \pi_2(0, 0)} \).

For systems with large \( k \), solving all the embedded Markov chains could be tedious. Fast approximations for \( p_i \)'s are provided in [22].

**Corollary 3.** The PAoI for a single M/G/1/2* queue is upper bounded by

\[
\frac{1}{\mu} + \frac{1}{\lambda_1 + \psi(\lambda_1)} - \frac{\psi(\lambda_1) - 1}{\lambda_1^2} \quad \text{ and } \quad \frac{2}{\lambda_1} - \frac{1}{\lambda_1} e^{\frac{\lambda_1}{\mu} - \psi(\lambda_1) + 1},
\]

where \( \psi(s) \) is the LST of service time.

**Proof.** It follows directly from Theorem 2 that when \( k = 1 \), the rejection probability is \( 1 - \frac{1}{\lambda_1 + \psi(\lambda_1)} \). \( \square \)

So far we characterized the rejection probability for Equation (4), which we can use to obtain the bounds of PAoI for each queue. In fact, when the variance of \( W_i \) is not large, the upper bounds that we provide in Equation (4) are decent approximations of PAoI for queues. We will show it numerically in Section 5.
It is found by Costa et al [7] that for M/M/1/1, M/M/1/2 and M/M/1/2* queues, increasing the arrival rate can reduce the PAoI continuously. However it is not the case in our model with multiple queues. We find that by increasing the arrival rate of a certain queue, its PAoI will be decreased, however PAoI for queues with lower priorities will be increased drastically. We will show the detail numerically in Section 5. Besides, we have the following theorem discussing the scenario when the arrival rate of a certain queue becomes large. We still keep the assumption that the service times are homogeneous with cdf $F(x)$.

**Theorem 4.** For $1 \leq i \leq k$, if $\lambda_i \to \infty$, then $E[A_j] \to \infty$ for $j > i$, and $E[A_j]$ will be bounded for $j \leq i$.

**Proof.** We first show that as $\lambda_i \to \infty$, then $\pi_j(0,...,0) \to 0$ for $j > i$. To show this, we know that in the subsystem $S_j$, the first element of the transition matrix for the embedded Markov chain is given by

$$a_0 = \int_0^\infty e^{-(\sum_{l=1}^j \lambda_l)x}dF(x) = \int_0^\infty (\sum_{l=1}^j \lambda_l)e^{-(\sum_{l=1}^j \lambda_l)x}F(x)dx.$$  

Since $F(x) \leq 1$ for any $x \in [0,\infty)$, by dominated convergence theorem, we have $\lim_{\lambda_i \to \infty} \int_0^\infty e^{-(\sum_{l=1}^j \lambda_l)x}dF(x)dx = 0$. Thus $a_0 \to 0$ and by result from [21] that $\pi_j(0,...,0) = \sum_{l=1}^j a_0\pi_j(0,...,0) + a_0\pi_j(0,...,0)$, we have $\pi_j(0,...,0) \to 0$ for $j > i$. From Theorem 2 we have $p_j = 1 - \frac{\lambda_j}{\pi_j(0,...,0)} - \frac{\lambda_j}{\pi_k(0,...,0)} = \frac{\lambda_j(1 - \frac{1}{\pi_j(0,...,0)})}{\pi_j(0,...,0)}$. From the fact that $\pi_j(0,...,0)$ for $j < i$ will not be affected by $\lambda_i$ and $\pi_j(0,...,0) < \pi_j(0,...,0)$ if $j \neq 0$ (see Theorem 1 of [21]), we then have $p_j \leq 1 - \frac{\lambda_j}{\pi_j(0,...,0)} < 1$ as $\lambda_i \to \infty$. Thus $E[A_j]$ is bounded by Equation (4).

For $j < i$, we have

$$p_i = \frac{\lambda_i(1 - p_i)}{\lambda_i(1 - p_i)} = \frac{1 - \pi_{i-1}(0,...,0) - \pi_i(0,...,0)}{\pi_{i-1}(0,...,0) + \pi_i(0,...,0)}.$$  

As $\lambda_i \to \infty$, $\frac{\lambda_i}{\pi_{i-1}(0,...,0)} \leq \frac{1}{\mu \pi_{i-1}(0,...,0)}$. By Equation (4) we prove the theorem. 

Theorem 4 shows that if we increase the arrival rate for a queue, the PAoI of queues with lower priorities will be greatly increased, while PAoI of queues with higher priorities will be bounded. It implies that if we have queues with traffic intensity significantly greater than the others, it is better to give high priorities to those queues with low traffic intensities to guarantee that all queues have a relative low PAoI.
4 Infinite Buffer Size

On one hand, keeping the most recent packet of each queue can help reduce the system traffic and guarantee a low PAoI for effective real-time decisions. On the other hand, for some applications, dropping packets is not an option when the entire data stream must be obtained for performing offline diagnostics. In such a scenario, processing all the generated packets is necessary, and for that, buffer size of each queue needs to be large enough. In this section, we discuss a model in which buffer size of each queue is infinite. This model has been discussed in [9, 25], however they do not consider queues with priorities. In this section, since there could be multiple packets waiting in each queue, it is necessary to ascertain the order of service within a queue. We consider FCFS and LCFS service discipline separately when the server serves packets from the same queue. Still, the server serves packets from high priority queues when the server becomes available.

Throughout this section, we assume that 
\[ \sum_{j=1}^{k} \frac{\lambda_j}{\mu_j} < 1 \]
so that the system is stable.

4.1 M/G/1 Type Queues with FCFS

We first discuss the model in which each queue is served according to FCFS discipline. From the definition of PAoI we know that when processing is complete for the \( j \)th packet from queue \( i \), the random variable corresponding to PAoI is equal to 
\[ A_{ij} = C_{ij} - r_{ij} - (r_{ij} - r_{i(j-1)}) \]
where \( C_{ij} \) is the sojourn time of packet \( j \) and \( r_{ij} - r_{i(j-1)} \) is the inter-arrival time between packet \( j \) and \( j-1 \). The PAoI for queue \( i \) can be written as 
\[ E[A_i] = E[W_i] + E[I_i] \]
where 
\[ E[W_i] \]
is the expected waiting time in queue and 
\[ E[I_i] = \frac{1}{\lambda_i} \]
is the expected inter-arrival time. From [8] we have the exact expression of 
\[ E[W_i] \]
for M/G/1 type queues with priority, thus the PAoI of queue \( i \) is given by:
\[ E[A_i] = E[W_i] + E[I_i] + E[P_i] \]
\[ = \frac{1}{2} \sum_{j=1}^{k} \lambda_j E[P_j^2] \frac{1}{(1 - \sum_{j=1}^{k} \lambda_j)(1 - \sum_{j=1}^{i-1} \lambda_j)} + \frac{1}{\lambda_i} + \frac{1}{\mu_i} \]  
(5)

Interestingly, from the expression of \( E[W_i] \) we find that the packets from higher priority queues always have shorter expected waiting times in queue compared with those from low priority queues. However, Equation (5) shows that higher priority queues do not always have shorter PAoI because \( \frac{1}{\lambda_i} \) and \( \frac{1}{\mu_i} \) also contribute to PAoI. Another interesting point from Equation (5) is that by increasing arrival rate \( \lambda_i \) we can reduce the PAoI for queue \( i \) but greatly enlarge the PAoI for queues with priority lower than \( i \). We will also show this result numerically in Section 5.

Bedewy et al. [3] considered the scheduling policy to minimize the average PAoI across queues, i.e., \( \frac{1}{k} \sum_{i=1}^{k} E[A_i] \). If we also consider the same objective and ask the design question of how to minimize the average PAoI across queues by assigning queue priorities, the answer is assigning high priorities to queues with low \( \rho_i = \frac{\lambda_i}{\mu_i} \), as we see in Theorem 5.

**Theorem 5.** If the queue priorities satisfy \( \rho_1 \leq \rho_2 \leq \ldots \leq \rho_k \), then the average PAoI across queues given by this priority order is the smallest among all the priority orders.
Proof. Since

\[
\frac{1}{k} \sum_{i=1}^{k} E[A_i] = \frac{1}{k} \sum_{i=1}^{k} \left[ \frac{1}{2} \sum_{j=1}^{k} \lambda_j E[P_j^2] \left(1 - \sum_{j=1}^{k} \rho_j \right)^{-1} \right] \left(1 - \sum_{j=1}^{k} \rho_j \right)^{-1}, \quad (6)
\]

changing priority orders only affects the denominator of the first term in Equation (6). So minimizing the average PAoI across queues is equivalent to minimizing \( \sum_{i=1}^{k} \frac{1}{(1 - \sum_{j=1}^{k} \rho_j)} \right)^{-1} \), if (\( \rho_1, \rho_2, ..., \rho_k \)) is the optimal priority order with \( \rho_i \geq \rho_{i+m} \), by switching the order of \( \rho_i \) and \( \rho_{i+m} \) we have a new priority order \( (\rho_1^*, \rho_2^*, ..., \rho_k^*) \) with \( \rho_i^* = \rho_{i+m} \), \( \rho_{i+m}^* = \rho_i \) and \( \rho_j = \rho_j \) for \( j \in \{1, ..., k\} \setminus \{i, i+m\} \). Then we have \( \sum_{i=1}^{j} \rho_i = \sum_{i=1}^{j} \rho_i^* \) for \( j < i \), \( \sum_{i=1}^{j} \rho_i \geq \sum_{i=1}^{j} \rho_i^* \) for \( i \leq j < i + m \) and \( \sum_{i=1}^{j} \rho_i = \sum_{i=1}^{j} \rho_i^* \) for \( j \geq i + m \). Thus we have

\[
\begin{align*}
\frac{1}{k} \sum_{i=1}^{k} \left( \frac{1}{1 - \sum_{j=1}^{i} \rho_j} \right) & = \sum_{i=1}^{k+m} \left[ \frac{1}{(1 - \sum_{j=1}^{i} \rho_j)} \left(1 - \sum_{j=1}^{i-1} \rho_j \right) \right] - \sum_{i=1}^{k} \left[ \frac{1}{(1 - \sum_{j=1}^{i} \rho_j^*)} \left(1 - \sum_{j=1}^{i-1} \rho_j^* \right) \right] \\
& \geq 0,
\end{align*}
\]

which contradicts to the assumption that \( (\rho_1, \rho_2, ..., \rho_k) \) is the optimal priority order. Therefore we prove the theorem. \( \square \)

From Theorem 5 we see that for M/G/1 type queues with FCFS discipline, it is always better to give higher queue priorities (if we have the option) to queues with smaller traffic intensities. In fact, this observation is also true for M/G/1/1+\( \sum \) \( 1^* \) queues that we discussed in Section 3. The intuitive reason for this is if we do the opposite, i.e., allowing high traffic queues to have high priority, the server would be busy serving high traffic intensity queues and barely have chance to serve low priority queues. Packets from low priority queues therefore would suffer a large waiting time. We will show this numerically in Section 5.

### 4.2 M/M/1 Type Queues with LCFS

It is shown in [11] that LCFS is the optimal service discipline of a G/G/1 system with the objective of minimizing PAoI. So if we fix the queue priorities, LCFS is also the optimal service discipline for each queue among all service disciplines when dropping packets is not allowed. In this subsection, we derive the PAoI for priority queues with LCFS within each queue. The server still chooses the highest priority queue when it becomes available, and from each queue it serves the last arrived packet first. There is no preemption during service. To derive the exact expression of PAoI, we use the method in Section 3 and focus on buffer state. Different from the model discussed in Section 3, here in each queue the buffer size is infinite. We now introduce a new service scheme here which has the same PAoI as LCFS. We first separate the queue into two virtual parts: initial buffer and main queue. The initial buffer can hold only one packet. Whenever a new arrival occurs, we send this new arrival into the initial buffer if it is empty. If there is a packet waiting in the initial buffer when a new arrival occurs, we replace it with the newly arrived packet and transfer the old one to the main queue. When the server serves a queue, it serves the packet from initial buffer first if it is not empty, then serves packets from main queue in an arbitrary order with the understanding that service times are iid. However, if an arrival occurs when the server is busy, this arrival goes to the initial buffer and waits
for the next available service. A demonstrative graph of the idea of initial buffer and main queue is shown in Figure 4, in which the server is processing a packet from initial buffer 1 and initial buffer 1 is empty.

This service scheme has the same PAoI as LCFS since the most recent arrival is always stored in the initial buffer. The benefit of modeling in this way is that we can characterize the PAoI of each queue by focusing on the initial buffer. The state of the initial buffer is either 0 or 1, and each period length of state 0 (when the buffer is empty), is equal to the inter-arrival time $I_i$ between packets. The time period of state 1 (when the buffer is full) is denoted as $W_i$, which we call the busy period of the initial buffer. Using the analysis in Section 3.1, the PAoI for queue $i$ is given as $E[A_i] = E[P_i] + E[W_i] + E[I_i] + E[G_i]$, where $E[P_i] = \frac{1}{\mu_i}$ is the expected service time, $E[W_i]$ is the expected length of period when initial buffer is full, $E[I_i] = \frac{1}{\lambda_i}$ is the expected inter-arrival time, and $E[G_i] = E[\frac{1}{\lambda_i} (1 - e^{-\lambda_i W_i})]$ is the expected gap from the arrival time of the packet in the initial buffer to the time when the buffer becomes empty, which is given in Lemma 1. For the remainder of subsection, we assume that the service times are exponential, so our model now becomes an M/M/1 type queueing system. We will use the analysis in Section 3.1 to derive the exact results for M/M/1 type queues with LCFS in this subsection.

Now we characterize $E[e^{-sW_i}]$. We modify and use the earlier defined notation by letting the vector $S(t) = (J(t), B_1(t), ..., B_k(t))$ indicate the state of the system, where $B_i(t) = (B_{iq}(t), B_{im}(t))$ is the number of packets in queue $i$ with the number of packets in initial buffer $B_{iq}(t)$ and the number of packets in main queue $B_{im}(t)$. Let $U_i$ be the remaining service time observed by a packet from source $i$ which enters the initial buffer, we then have for queue 1

$$E[e^{-sW_1}|B_{1q}(0) = 0] = E[e^{-sU_1}|B_{1q}(0) = 0] = P(J(0) = 0|B_{1q}(0) = 0) + \sum_{j=1}^k \psi_j(s)P(J(0) = j|B_{1q}(0) = 0).$$

We now consider queue 2. Let $T_1$ be the time period that the server is busy processing packets from queue 1. We have $E[e^{-sW_2}] = E[e^{-s(U_2+T_1)}]$. Similar to Section 3.1 by conditioning on different scenarios, we have

$$E[e^{-s(U_2+T_1)}|J(0) = 0, B_1(0) = 0, B_{2q}(0) = 0] = 1,$$

$$E[e^{-s(U_2+T_1)}|J(0) = j, B_1(0) = n, U_2 = u, B_1(u) = n + m, B_{2q}(0) = 0] = E[e^{-su}(s)\eta_{1+m}^n(s)],$$

$$E[e^{-s(U_2+T_1)}|J(0) = j, B_1(0) = 0, U_2 = u, B_1(u) = n, B_{2q}(0) = 0] = e^{-su}E[e^{-sT_1}] = e^{-su}\eta_0^u(s),$$

and
\[ E[e^{-s(U_2 + T_1)} | J(0) = j, B_1(0) = 0, U_2 = u, B_1(u) = 0, B_2(0) = 0] = e^{-su}. \]

We also know the busy period that the server continuously serves packets from queue 1 is given by

\[ \eta_1(s) = E[e^{-sT_1}] = \psi_1(s + \lambda_1 - \lambda_1 \eta_1(s)) \] (7)

from [6].

We can again use a CTMC to find the stationary distribution of the system, however the dimension of the state space is infinite, hence solving for the exact distribution is difficult even for \( k = 2 \). To find its approximation we can truncate the state space into a finite one. The state space for this two-queue case is given as \( S(t) = (J(t), B_1 q(t), B_1 m(t), B_2 q(t), B_2 m(t)) \) and we can use a similar approach as what we used in Section 3.1 to solve the finite dimension matrix.

### 4.3 Bounds and Approximations for M/G/1 Type Queues with LCFS

From Section 4.2 we find that even for M/M/1 type queues with LCFS, the exact solution of PAoI is difficult to obtain. For M/G/1 type queues, the analysis is expected to be more complicated. However, in many scenarios bounds and approximations of PAoI are useful. In this subsection, we provide bounds of PAoI for M/G/1 type queues with priority across queues and LCFS within each queue.

Similar to what we did in Section 3.2, since in the system of M/G/1 type queues with LCFS, there is one initial buffer in each queue, we can thus give the upper bound of queue \( i \)'s PAoI using Jensen’s inequality as

\[
E[A_i] = E[P_i] + E[W_i] + E[I_i] + E[G_i] \\
\leq \frac{1}{\mu_i} + \frac{p_i}{\lambda_i (1 - p_i)} + \frac{2}{\lambda_i} - \frac{1}{\lambda_i} e^{-\frac{\mu_i}{\lambda_i} p_i},
\] (8)

for all \( i \in \{1, ..., k\} \). It is crucial to note that here \( p_i \) is the probability that the initial buffer is full. Now we introduce the method of finding \( p_i \)'s by providing Lemma 6 for the case of \( k = 1 \) first.

**Lemma 6.** For the M/G/1 system with \( k = 1 \), the probability that the initial buffer is full is given by \( p_1 = \frac{\lambda_1}{\mu_1} + 1 + \psi_1(\lambda_1) \), where \( \psi_1(u) \) is the LST of the service time.

**Proof.** From Figure 5 we find that the busy period of the initial buffer always occurs when the server is serving (busy), and ends when the service is complete. From Figure 5 we see that \( P_1 = S_1 - S_1 \), thus the period that the initial buffer being full, i.e., \( \tilde{W} \), is the waiting time of the first packet that arrives during the processing time \( P_1 \) of a certain packet. From the property of Poisson arrivals and Campbell’s Theorem we have

\[
E[\tilde{W}|P_1 = u] = \sum_{m=1}^{\infty} \frac{m}{m + 1} \frac{ue^{-\lambda_1 u} (\lambda_1 u)^m}{m!} = u - \frac{1}{\lambda_1} + \frac{1}{\lambda_1} e^{-\lambda_1 u}.
\]

By unconditioning on \( P_1 = u \) we have \( E[\tilde{W}] = \int_0^{\infty} (u - \frac{1}{\lambda_1} + \frac{1}{\lambda_1} e^{-\lambda_1 u}) dF_1(u) = \frac{1}{\mu_1} - \frac{1}{\lambda_1} + \frac{\psi_1(\lambda_1)}{\lambda_1} \). However, it is important to note that this \( E[\tilde{W}] \) is the expected busy time for initial buffer during the processing time of a packet. To obtain \( p_i \), we need the following argument. Suppose \( n(t) \) packets have been served during
(0, t]. Thus the amount of time that the initial buffer being full during (0, t] is $n(t)E[\hat{W}]$. If the queue is stable, we have \( n(t) \) converging $\lambda_1 t$ as $t \to \infty$. Therefore

$$\lim_{t \to \infty} \frac{n(t)E[\hat{W}]}{t} = \lambda_1 E[\hat{W}],$$

which is the stationary probability that the initial buffer is full. Note that $\frac{\lambda_1}{\mu_1} - 1 + \psi_1(\lambda_1)$ is a legitimate probability as it always lies within $[0, 1]$. To show this, first from the fact that $\psi_1(\lambda_1) = \int_0^\infty e^{-\lambda x}dF(x) \leq 1$, we have $\frac{\lambda_1}{\mu_1} - 1 + \psi_1(\lambda_1) \leq \frac{\lambda_1}{\mu_1} < 1$ from stability assumption. Since $\psi_1(\lambda_1) = \int_0^\infty e^{-\lambda x}dF(x) \geq \int_0^\infty (1 - \lambda_1 x)dF_1(x) = 1 - \frac{\lambda_1}{\mu_1}$, we have $\frac{\lambda_1}{\mu_1} - 1 + \psi_1(\lambda_1) \geq 0$. Thus $\frac{\lambda_1}{\mu_1} - 1 + \psi_1(\lambda_1)$ is a legitimate probability.

**Corollary 7.** For an M/G/1 queue with LCFS for $k = 1$, the PAoI is upper bounded by $\frac{1}{\mu_1} + \frac{p_1}{\lambda_1(1 - p_1)} + \frac{2}{\lambda_1} - \frac{1}{\lambda_1} - \frac{1}{\lambda_1} \psi_1(\lambda_1)$ with $p_1 = \frac{\lambda_1}{\mu_1} - 1 + \psi_1(\lambda_1)$.

Now we discuss the case when $k \geq 2$. Notice that for each packet that is in service, if there is a new arrival from queue 1 occurring during this service time, then the busy period for initial buffer 1 is from the arrival time of this new packet to the completion time of the packet being processed. From Lemma 7, the busy period for initial buffer 1 if a type $i$ packet is being processed when the busy period starts, is given by $\frac{1}{\mu_i} - \frac{1}{\lambda_i} + \frac{1}{\lambda_i} \psi_i(\lambda_i)$, and we have $p_1 = \sum_{i=1}^k \lambda_i(\frac{1}{\mu_i} - \frac{1}{\lambda_i} + \frac{1}{\lambda_i} \psi_i(\lambda_i))$.

To get the rejection probability $p_i$ for queue $i \geq 2$, we use the idea introduced by Kella and Yechiali [14]. We merge the queues with priority higher than $i$ as one class and the other queues as another class by letting $\lambda_{ai} = \sum_{j=1}^{i-1} \lambda_j$, $\lambda_{bi} = \sum_{j=1}^k \lambda_j$, $\rho_{ai} = \sum_{j=1}^{i-1} \rho_j$ and $\rho_{bi} = \sum_{j=1}^k \rho_j$. We also let $F_{ai}(x) = \sum_{j=1}^{i-1} \frac{\lambda_j}{\lambda_i} F_j(x)$ be the service time distribution for packets from queue $j < i$, with mean $E[F_{ai}]$ and $F_{bi}(x) = \sum_{j=i}^k \frac{\lambda_j}{\lambda_i} F_j(x)$ be the service time distribution for packets from queue $j \geq i$, with mean $E[F_{bi}]$. Notice that the busy period of initial buffer $i$ only ends when there is no packet from queue $j < i$. We now classify the busy periods of server (the time period during which the server is continuously serving packets) into two types, and hence we can characterize the busy period of initial buffer $i$. One type of busy period $V_{ai}$ of the server starts with

![Figure 5: Initial Buffer and Total Queue Length (Including Initial Buffer and Main Queue) for LCFS](image)
processing a packet with priority higher than $i$, and ends when there is no packet of priority higher than $i$ left in the system. The other type of busy period $V_{bi}$ start with processing a packet with priority equal to or lower than $i$, and also ends when there is no packet of priority higher than $i$ left in the system. Similar to what we did in Section 3.1 and 4.2, by conditioning on service time of the first packet in a busy period, the LST of $V_{ai}$ and $V_{bi}$, denoted as $\tilde{V}_{ai}(s)$ and $\tilde{V}_{bi}(s)$, are given as $\tilde{V}_{ai}(s) = \psi_{ai}(s + \lambda_{ai} - \lambda_{ai}\tilde{V}_{ai}(s))$ and $\tilde{V}_{bi}(s) = \psi_{bi}(s + \lambda_{ai} - \lambda_{ai}\tilde{V}_{ai}(s))$, where $\psi_{ai}(s)$ is the LST of $F_{ai}(x)$ and $\psi_{bi}(s)$ is the LST of $F_{bi}(x)$.

By taking the derivative of $\tilde{V}_{ai}(s)$ and $\tilde{V}_{bi}(s)$ at $s = 0$, the expected length of server’s busy periods can be given as

$$E[V_{ai}] = \frac{E[P_{ai}]}{1 - \rho_{ai}}$$

and

$$E[V_{bi}] = \frac{E[P_{bi}]}{1 - \rho_{ai}},$$

Note that busy period $V_{bi}$ always starts with one packet from queue $j \geq i$, thus we know

$$P:\text{(system in } V_{bi}) = \lambda_{bi} \frac{E[P_{bi}]}{1 - \rho_{ai}}.$$  

Since when the server is busy, it is either in busy period $V_{ai}$ or $V_{bi}$, we have

$$P\text{(system in } V_{ai}) = \sum_{j=1}^{k} \rho_{j} - \lambda_{bi} \frac{E[P_{bi}]}{1 - \rho_{ai}} = \tilde{\lambda}_{ai} \frac{E[P_{ai}]}{1 - \rho_{ai}},$$

where $\tilde{\lambda}_{ai} = \sum_{j=1}^{k} \rho_{j} - \lambda_{bi} \frac{E[P_{bi}]}{1 - \rho_{ai}}$ is the “arrival rate” of busy period $V_{ai}$. From Lemma 6, we know that during busy period $V_{ai}$, the time period of initial buffer being busy is given as $E[\hat{W}_{ai}] = \int_{0}^{\infty} (u - \frac{1}{\lambda_{ai}} + \frac{1}{\lambda_{ai}} e^{-\lambda_{ai}u})dF_{ai}(u)$. Similarly, we have $E[\hat{W}_{bi}] = \int_{0}^{\infty} (u - \frac{1}{\lambda_{bi}} + \frac{1}{\lambda_{bi}} e^{-\lambda_{bi}u})dF_{bi}(u)$. From the same argument in Lemma 6, we have

$$p_{i} = \tilde{\lambda}_{ai} E[\hat{W}_{ai}] + \lambda_{bi} E[\hat{W}_{bi}].$$

Now we get the rejection probabilities that can be used to obtain upper bounds of PAoI for LCFS queues by Equation (8). It is important to note that these bounds can serve as reasonable approximations of PAoI for queues with LCFS, which we will show numerically in Section 5.

### 5 Numerical Study

In this section we will firstly use a numerical study to verify the exact solutions for M/M/1+\sum 1* that we provided in Section 3.1 and then test the bounds which we provided in Section 3.2 and Section 4.3. Besides, we will develop our insights based on the numerical studies.

We begin our discussion by comparing simulation results with exact solutions for M/M/1+\sum 1* system with $k = 2$. The comparison is done by changing one parameter from $\lambda_{1}$, $\lambda_{2}$, $\mu_{1}$ and $\mu_{2}$ while keeping the others fixed. The results are shown in Figure 6. From plots in Figure 6 we can see that the simulation
results match the exact solutions that we provide in Section 3.1, thus verifying our results. Figure 6(a) shows that when we increase the arrival rate for the priority 1 queue, its PAoI is drastically decreased, while the PAoI for queue 2 increasing linearly. Figure 6(b) shows that if we increase the arrival rate of queue 2, its PAoI will decrease dramatically, while PAoI of queue 1 increases slowly. Figure 6(c) and (d) show that when service rate increases, PAoI for both queues are decreased. Interestingly, we find that when queue 1 has a low service rate, PAoI for both queues will be large, while PAoI of queue 1 is not significantly affected by the service rate change of queue 2. It also implies that the average PAoI across all queues, i.e., \( \frac{1}{k} \sum_{i=1}^{k} E[A_i] \), is more sensitive to the arrival rate and service rate of high priority queues. We then test how the average PAoI across queues is affected by parameters, which we show in Figure 7. From Figure 7(a) we see that by increasing the service rate of either queues, the average PAoI across queues will be reduced, and increasing the service rate of queue 1 makes this reduction more significant. Figure 7(b) shows that by increasing the arrival rate of queue 2, the average PAoI across queues is decreased. This is because the PAoI for queue 1 is not sensitive to the arrival rate of queue 2, as we show in Theorem 4. However, when we increase the arrival rate of queue 1, the average PAoI will decrease drastically at the beginning, and increase afterwards. This is because the PAoI of queue 2 increases constantly when we increase \( \lambda_1 \), which we also see from Figure 6(a). Note that although we only discuss the optimization problem of minimizing average PAoI across queues here, since we have the exact solution for PAoI, we could also formulate and solve optimization problems such as minimizing average weighted PAoI (similar to [11]) and minimizing the maximum PAoI (similar to [9]).

Figure 6: M/M/1+∑ 1* Type Queues with Buffer Size One
Next we consider queues with general service times. The bounds for M/G/1+∑^1^\* type queues with buffer size one and \( k = 3 \) are shown in Figure 8, where the bounds are provided by Equation 4. We test the bounds by letting service time follow exponential, uniform and gamma distributions. Note that in Figure 8 we provide the approximations for exponential service too, although we have the exact solution for PAoI when service times are exponential. We find from Figure 8 that Equation 4 serves as a decent approximation for the actual PAoI since the bounds and simulation curves for all queues are close. The
three service distributions in Figure 8 have the same mean but the Gamma case has the lowest variance, so that we can conclude that the approximation can become closer to the exact solution when the variance of service time becomes smaller, as the gap between each approximation curve and simulation curve is smaller in Figure 8(c) than that in Figure 8(a) and (b). It is also interesting that when service time variance is small, the PAoI of lower priority queues are relatively large. The PAoI of queue 3 in Figure 8(c) is larger than it in Figure 8(a) and (b). This is because when service time has large variance, it is likely for an arrival to see a large packet in service, therefore the rejection probability for queue 1 is high. On the contrary, if the service time has a small variance, the server is more likely to be occupied by packets from queue 1 so that the rejection rate for queue 1 is low. From Theorem 4 and our discussion about Figure 6 and 7 we know that PAoI of queues with low priorities are sensitive to the rejection rate of high priority queues (such as queue 1). Therefore the PAoI for queue 2 and 3 are larger when the service time is less variable.

Next we consider queues with infinite buffer size. The bounds for M/G/1 type queues with LCFS are shown in Figure 9. We also test the bounds given in Section 4.3 for exponential, uniform and gamma distributed service times, and we find that the upper bound provided in Section 4.3 also serves as excellent approximations for the exact solution. We also find that in M/G/1 type queues with LCFS, by increasing the arrival rate of queue 1, PAoI of queue 1 is significantly reduced, and PAoI for lower priority queues is increased at the same time. We do not present the numerical test for M/G/1 queues with FCFS here, as its analysis is exact and also straightforward.

Figure 9: Bounds for M/G/1 Type Queues with LCFS
Next we address PAoI by comparing the single buffer size case against infinite buffer size cases under FCFS and LCFS. In fact, since in the M/G/1 system with infinite buffer size, if we keep replacing the packets with new arrivals, then there is at most one packet waiting in each queue therefore the system will act exactly the same as M/G/1+∑1* system. So here we consider the PAoI under M/G/1+∑1* and M/G/1 with FCFS and LCFS altogether. In Figure 10 we plot the PAoI for each queue in the case of k = 2, and in Figure 11 we plot the average PAoI across all queues (1/k ∑i=1 E[Ai]). In both Figure 10 and 11 we use simulation results for LCFS (as its exact solution is difficult to obtain) and exact results for M/G/1 with FCFS and M/G/1+∑1* model. From Figure 10 we see that under FCFS, LCFS and M/G/1+∑1*, PAoI of queue 2 is sensitive to the change of λ1, however PAoI of queue 1 is less sensitive to λ2. This is because the PAoI for queue 2 highly depends on the busy time of queue 1. For FCFS, the PAoI increases greatly when arrival rate becomes large. This is because under FCFS, every packet that arrives the system needs to be processed, and increasing arrival rate enlarges the average queue size, causing packets to wait a longer time. From the average PAoI across queues shown in Figure 11 we can see that increasing the arrival rate for the high priority queue enlarges the PAoI much faster than increasing λ2. It indicates that when designing the priority for queues to minimize average PAoI across queues, the one with the lowest traffic intensity should be allocated with the highest priority. We also proved this result in Section 4 for M/G/1 queues with FCFS.
Also notice from Figure 11 that M/M/1+$\sum 1^*$ result in the lowest PAoI followed by LCFS and then FCFS, which conforms to our expectation.

6 Concluding Remarks and Future Work

In this research we considered a multi-class multi-buffer queueing system where each class of data source generates packets according to a Poisson process and a single processor uses a static priority scheme to serve the packets. We characterized the PAoI for such a system under two situations: (i) when the buffer size for each queue is one; (ii) when the buffer size for each queue is infinite and service disciplines within each queue can be FCFS or LCFS. We obtained exact expressions for PAoI in both case (i) and (ii) when the service times are exponential, and bounds which serve as excellent approximations when service times are general. The methodology leverages upon insights from the stochastic models that result in the probabilities for deriving the PAoI.

Using PAoI results we make a few observations that are useful in determining priorities and sampling rate for release times. We find that for minimizing the average PAoI across queues, it is beneficial to give higher priorities to queues with lower traffic intensities. Besides, we find that the PAoI of queues with low priorities are more sensitive to the packet arrival rate of high priority queues, and increasing the arrival rate for one queue, while reducing the PAoI for this certain data source, would significantly increase the PAoI of queues with lower priorities.

Since in this paper we mainly focus on static queue priorities, in our future work we will consider a system with dynamic priorities. Besides, in smart manufacturing systems where the status of machines changes over time, sampling with a time-varying rate is also possible and it is interesting to consider the PAoI with time-varying arrival rates. Moreover, the variance of PAoI is also useful in measuring the data freshness in real-time systems, and the distribution of PAoI is also of interest. Thus there are numerous opportunities for research in the area of PAoI for multi-priority queues.

References

[1] Ivo Adan, Onno J. Boxma, and Jacobus Adrianus Cornelis Resing. Queueing models with multiple waiting lines. Queueing Systems, 37(1-3):65–98, 2001.

[2] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: dynamic flow scheduling for data center networks. In Nsdi, volume 10, pages 89–92, 2010.

[3] Ahmed M Bedewy, Yin Sun, Sastry Kompella, and Ness B Shroff. Age-optimal sampling and transmission scheduling in multi-source systems. arXiv preprint arXiv:1812.09463, 2018.

[4] Ahmed M Bedewy, Yin Sun, and Ness B Shroff. Age-optimal information updates in multihop networks. In 2017 IEEE International Symposium on Information Theory (ISIT), pages 576–580. IEEE, 2017.

[5] Mokrane Bouzeghoub. A framework for analysis of data freshness. In Proceedings of the 2004 international workshop on Information quality in information systems, pages 59–67. ACM, 2004.

[6] Richard Walter Conway, William L Maxwell, and Louis W Miller. Theory of scheduling. Courier Corporation, 2003.
[7] Maice Costa, Marian Codreanu, and Anthony Ephremides. On the age of information in status update systems with packet management. *IEEE Transactions on Information Theory*, 62(4):1897–1910, 2016.

[8] Natarajan Gautam. *Analysis of queues: methods and applications*. CRC Press, 2012.

[9] Longbo Huang and Eytan Modiano. Optimizing age-of-information in a multi-class queueing system. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 1681–1685. IEEE, 2015.

[10] Narendra Kumar Jaiswal. *Priority queues*. Elsevier, 1968.

[11] Igor Kadota, Elif Uysal-Biyikoglu, Rahul Singh, and Eytan Modiano. Minimizing the age of information in broadcast wireless networks. In *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*, pages 844–851. IEEE, 2016.

[12] Sanjit Kaul, Roy Yates, and Marco Gruteser. Real-time status: How often should one update? In *INFOCOM, 2012 Proceedings IEEE*, pages 2731–2735. IEEE, 2012.

[13] Sanjit K Kaul and Roy D Yates. Age of information: Updates with priority. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 2644–2648. IEEE, 2018.

[14] Offer Kella and Uri Yechiali. Priorities in M/G/1 queue with server vacations. *Naval Research Logistics (NRL)*, 35(1):23–34, 1988.

[15] Vidyadhar G Kulkarni. *Modeling and analysis of stochastic systems*. Chapman and Hall/CRC, 2016.

[16] Ali Maatouk, Mohamad Assaad, and Anthony Ephremides. Age of information with prioritized streams: When to buffer preempted packets? *arXiv preprint arXiv:1901.05871*, 2019.

[17] Elias Masry. Poisson sampling and spectral estimation of continuous-time processes. *IEEE Transactions on Information Theory*, 24(2):173–183, 1978.

[18] Elie Najm and Rajai Nasser. Age of information: The gamma awakening. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 2574–2578. IEEE, 2016.

[19] Elie Najm and Emre Telatar. Status updates in a multi-stream M/G/1/1 preemptive queue. In *IEEE Infocom 2018-IEEE Conference On Computer Communications Workshops (Infocom Wkshps)*, pages 124–129. IEEE, 2018.

[20] Toyofumi Takenaka. Buffer management schemes for a heterogeneous packet switching system. *Electronics and Communications in Japan (Part I: Communications)*, 67(11):46–54, 1984.

[21] Toyofumi Takenaka. Analysis of a nonpreemptive $\Sigma M_i/G/1(\Sigma N_i)$ system. *Electronics and Communications in Japan (Part I: Communications)*, 72(3):75–84, 1989.

[22] Toyofumi Takenaka, Takeshi Akaike, and Kazumasa Takami. Characteristics and approximation methods of a nonpreemptive $\Sigma M_i/G/1(\Sigma N_i)$ system. *Electronics and Communications in Japan (Part I: Communications)*, 72(3):85–94, 1989.

[23] Dimitri Theodoratos and Mokrane Bouzeghoub. Data currency quality factors in data warehouse design. In *DMDW*, page 15, 1999.
[24] Luis M Vaquero and Luis Rodero-Merino. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, 44(5):27–32, 2014.

[25] Roy D Yates and Sanjit K Kaul. The age of information: Real-time status updating by multiple sources. *IEEE Transactions on Information Theory*, 65(3):1807–1827, 2019.