Optimizing Data Freshness, Throughput, and Delay in Multi-Server Information-Update Systems

Ahmed M. Bedewy, Yin Sun, and Ness B. Shroff

Dept. of ECE, The Ohio State University, Columbus, OH.

***Abstract*—In this work, we consider an information-update system where a source sends update packets to a remote monitor via multiple network servers. An important metric of data freshness at the monitor is the *age-of-information*, or simply age, which is defined as the time elapsed since the freshest packet at the monitor was generated. Recent studies on information-update systems have shown that the age-of-information can be reduced by intelligently dropping stale packets. However, packet dropping may not be appropriate in many systems (such as news and social networks), where users are interested in not just the latest updates, but also past news. Therefore, all packets may need to be successfully delivered. In this paper, we study how to optimize the age-of-information without throughput loss. We show that the preemptive Last-Come First-Served (LCFS) policy simultaneous optimizes the age, throughput, and delay performance in infinite buffer systems, and hence is appropriate for practical information-update systems. We also show age-optimality regardless of the buffer size. Numerical results are provided to validate our theoretical results.**

I. Introduction

With the ever prevalence of mobile devices and applications, the demand has increased on real-time information updates, such as news, weather reports, email notifications, stock quotes, social updates, mobile ads, etc. Also, in network-based monitoring and control systems, timely status updates are crucial. These include, but are not limited to, sensor networks used in temperature or other physical phenomenon, and autonomous vehicle systems.

A common objective in these applications is to keep the monitor updated with the latest information. To identify the timeliness of the update, a metric called the *age-of-information*, or simply *age*, was defined in [1]–[4]. At time t, if U(t) is defined as the time when the freshest update at the monitor was generated at the source, the status update age is $\Delta(t) = t - U(t)$. Hence, the age is the time elapsed since the freshest packet was generated.

Most works try to analytically characterize the time-average age of different information-update policies under Poisson arrival process, and find a policy with a smaller time-average age [4]–[10]. In [4]–[6], the update generation rate was optimized to improve data freshness in First-Come First-Served (FCFS) information-update systems. To improve the age, these studies reduce the update generation rate from the maximum achievable rate, which in turn degrades system throughput. In [7], [8], it was found that the age can be improved by discarding old packets waiting in the queue if a new sample arrives, which can greatly reduce the impact of queueing delay on data freshness. However, discarding packets may not be appropriate for applications where the users are interested in not just the latest updates, but also past news, where all packets may need to be successfully delivered. In [9], [10], the time-average age was characterized for several information-update policies under Poisson arrival process. Applications of information updates in channel information feedback and sensor networks were considered in [11], [12].

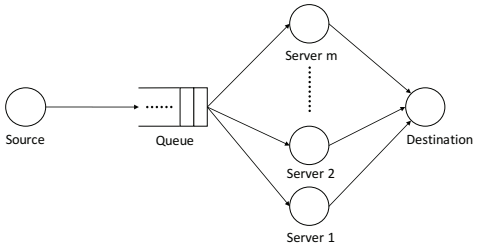

Figure 1: System model.

One important problem is how to achieve the optimal age performance in information-update systems. Recently, the joint control of the generation and transmission of update packets was studied in [12]–[14]. An information update policy was developed in [14], which was proven to minimize the time-average age and time-average age penalty among all causally feasible policies. A counter-intuitive phenomenon was revealed in this study: While a *zero-wait* or *work-conserving* policy, that submits a fresh update once the server becomes idle, achieves the maximum throughput and the minimum average delay, surprisingly, this zero-wait policy does not always optimize the age. It was shown that in many scenarios the age-optimal policy is to insert waiting times between updates, which is, however, not throughput-optimal. Therefore, how to jointly optimize data freshness, throughput, and delay performance in information-update systems remains an open problem.

In this paper, we consider an information-update system with multiple servers, which is illustrated in Fig. 1. We aim to answer the following questions: How to minimize the age-of-information without throughput loss? How to establish age-optimality in a general policy space and under arbitrary arrival process? Is it possible to simultaneously optimize multiple performance metrics, such as age, throughput, and delay? To that end, the following are the key contributions of this paper:

- We prove that, if the packet service times are *i.i.d.* exponentially distributed, then for an arbitrary arrival process and any buffer size, the preemptive Last-Come First-Served (LCFS) policy achieves an age that is stochastically smaller than any causally feasible policies (Theorem 1). This implies that the preemptive LCFS policy minimizes many data freshness metrics, including time-average age, average peak age, and time-average age penalty (Corollary 5). The intuition behind this age-optimality result is that new update packets with the freshest information are served as early as possible in the preemptive LCFS policy.
- In addition, we show that if the queue has an infinite

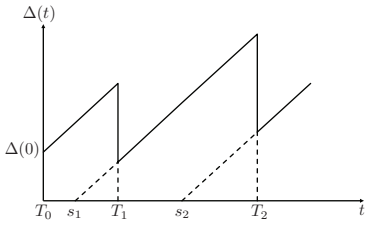

Figure 2: Evolution of the age-of-information $\Delta(t)$.

buffer size, then the preemptive LCFS policy is also throughput-optimal and delay-optimal among all causally feasible policies (Theorem 6). Numerical results are provided to validate our theoretical results.

The closest study to our work is [9], which analyzed the time-average age of preemptive and non-preemptive LCFS policies for a single-server system with Poisson arrival process and a buffer size of one packet. In our paper, we prove that the preemptive LCFS policy is age-optimal for multi-server systems with an arbitrary arrival process and any buffer size. Hence, our study complements that of [9]. To the best of our knowledge, this is the first study which simultaneously optimizes data freshness, throughput, and delay in information-update systems.

II. System Model

We consider an information-update system depicted in Fig. 1, where a source sends an infinite sequence of update packets to a destination through m identical servers. Each server could be a wireless channel, a TCP connection, etc. The update packets are generated at time $s_1, s_2, \ldots$, where $s_1 = 0 \leq s_2 \leq \ldots$ are deterministic and arbitrarily given. After generation, each update packet is stored in a queue, waiting to be assigned to one of the servers. Let B denote the buffer size of the queue which can be infinite, finite, or even zero. If B is finite, the queue buffer may overflow under bursty traffic and some packets are dropped, which would incur a throughput loss. The packet service times are exponentially distributed with rate μ, which are *i.i.d.* across time and servers. Let T_i denote the i-th packet delivery time instant.

The system starts to operate at time $t = 0$. Define $U(t)$ as the time when the freshest update at the monitor at time t was generated at the source, where we set $U(t) = 0$ at $t = 0$. The *age-of-information*, or simply the *age*, is defined as

$$\Delta(t) = t - U(t). \tag{1}$$

From this definition, we can deduce that the age increases linearly with t but is reset to a smaller value with each update delivery that contains newer information, as shown in Fig. 2. Define A_k as the k-th peak value of $\Delta(t)$ since time $t = 0$.

Following [14], we define an age penalty function $g(\Delta(t))$ to represent the level of "dissatisfaction" for data staleness or the "need" for new information update, where $g(\cdot)$ is a general non-decreasing function. A practical example of the age penalty function can be found in [11].

A. Scheduling Policy

A scheduling policy, denoted by π, assigns update packets to the servers over time. A scheduling policy can be either preemptive or non-preemptive.

Definition 1. Service Preemption: In a *preemptive* policy, a server can switch to send an incoming packet with a higher priority at any time; the preempted packets will be stored back to the queue if there is enough buffer space and sent at a later time when the servers are available again. In contrast, in a *non-preemptive* policy, a server must complete delivering the current packet before starting to send another packet.

Definition 2. Work Conservation: A policy is said to be *work-conserving*, if no server is idle when there are packets waiting in the queue.

We use Π to denote the set of all causal policies. Let $\Pi_{wc} \subseteq \Pi$ be the set of work-conserving feasible policies, and $\Pi_{nwc} \subseteq \Pi$ be the set of non-work-conserving feasible policies.

B. Age Optimality

We will need the following definitions: Let $\vec{x} = (x_1, x_2, \ldots, x_n)$ and $\vec{y} = (y_1, y_2, \ldots, y_n)$ be two vectors in $\mathbb{R}^n$, then we denote $\vec{x} \leq \vec{y}$ if $x_i \leq y_i$ for $i = 1, 2, \ldots, n$.

Definition 3. Stochastic Ordering: [15] Let X and Y be two random variables. Then, X is said to be stochastically smaller than Y (denoted as $X \leq_{\text{st}} Y$), if

$$P\{X > x\} \leq P\{Y > x\}, \quad \forall x \in \mathbb{R}.$$

Definition 4. Multivariate Stochastic Ordering: [15] A set $U \subseteq \mathbb{R}^n$ is called upper if $\vec{y} \in U$ whenever $\vec{y} \geq \vec{x}$ and $\vec{x} \in U$. Let $\vec{X}$ and $\vec{Y}$ be two random vectors. Then, $\vec{X}$ is said to be stochastically smaller than $\vec{Y}$ (denoted as $\vec{X} \leq_{\text{st}} \vec{Y}$), if

$$P\{\vec{X} \in U\} \leq P\{\vec{Y} \in U\}, \quad \text{for all upper sets} \quad U \subseteq \mathbb{R}^n.$$

Definition 5. Stochastic Ordering of Stochastic Processes: [15] Let $\{X(t), t \in [0, \infty)\}$ and $\{Y(t), t \in [0, \infty)\}$ be two stochastic processes. Then, $\{X(t), t \in [0, \infty)\}$ is said to be stochastically smaller than $\{Y(t), t \in [0, \infty)\}$ (denoted by $\{X(t), t \in [0, \infty)\} \leq_{\text{st}} \{Y(t), t \in [0, \infty)\}$), if, for all choices of an integer n and $t_1 < t_2 < \ldots < t_n$ in $[0, \infty)$, it holds that

$$(X(t_1), X(t_2), \ldots, X(t_n)) \leq_{\text{st}} (Y(t_1), Y(t_2), \ldots, Y(t_n)), \tag{2}$$

where the multivariate stochastic ordering in (2) was defined in Definition 4.

Definition 6. Age Optimality: A policy $P \in \Pi$ is said to be *age-optimal*, if for all $\pi \in \Pi$

$$\{\Delta_p(t), t \in [0, \infty)\} \leq_{\text{st}} \{\Delta_\pi(t), t \in [0, \infty)\}. \tag{3}$$

As we will see in Corollary 5, this definition of age optimality is stronger than many definitions proposed in prior studies.

III. Optimality Analysis

In this section, we study the preemptive LCFS policy depicted in Algorithm 1, where α_i is the i-th smallest time stamp of the packets under service. We will show that the preemptive LCFS policy is age-optimal, which is stated in the following theorem.

Theorem 1. If the packet service times are exponentially distributed and *i.i.d.* across time and servers, then for any given arrival time sequence $s_1, s_2, \ldots$ and buffer size B, the preemptive LCFS policy is age-optimal.

Algorithm 1: Preemptive Last Come First Served policy.

$\alpha_i := 0,\ i = 1, \dots m$;
while *the system is ON* **do**
  if *a new packet with time stamp s arrives* **then**
    if *all servers are busy* **then**
      The new packet is assigned to a server by preempting the packet with time stamp α_1;
      $\alpha_i := \alpha_{i+1},\ i = 1, \dots m-1$;
      $\alpha_m := s$;
      if *the buffer is not full* **then**
        The preempted packet is stored back to the queue;
      else // the buffer is full
        The preempted packet is dropped;
      end
    else // At least one of the servers is idle
      Assign the new packet to one idle server;
    end
  end
  if *a packet is delivered* **then**
    if *the buffer is not empty* **then**
      Pick any packet in the buffer and assign it to the idle server;
    end
  end
end

We need to define the system state of any policy π:

Definition 7. At any time t, the system state of policy π is specified by $V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$, where $U_\pi(t)$ is the largest time stamp of the packets that have already been delivered to the destination. Define $\alpha_{i,\pi}(t)$ as the i-th smallest time stamp of the packets being processed by the servers. Without loss of generality, if k servers are sending stale packets such that $\alpha_{1,\pi}(t) \le \ldots \le \alpha_{k,\pi}(t) \le U_\pi(t)$ or k servers are idle, then we set $\alpha_{1,\pi}(t) = \ldots = \alpha_{k,\pi}(t) = U_\pi(t)$. Hence,

$$U_\pi(t) \le \alpha_{1,\pi}(t) \le \ldots \le \alpha_{m,\pi}(t). \tag{4}$$

Let $\{V_\pi(t), t \in [0,\infty)\}$ be the state process of policy π, which is assumed to be right-continuous.

For notational simplicity, let policy P denote the preemptive LCFS policy. Then, in policy P, $\alpha_{1,P}(t), \alpha_{2,P}(t), \dots, \alpha_{m,P}(t)$ are the time stamps of m freshest packets among all packets generated during $[0,t]$.

The key step in the proof of Theorem 1 is the following lemma, where we compare policy P with any work-conserving policy $\pi \in \Pi_{wc}$.

Lemma 2. For any given arrival time sequence $s_1, s_2, \dots$ and buffer size B, suppose that $V_P(0^-) = V_\pi(0^-)$ for all work conserving policies $\pi \in \Pi_{wc}$, then

$$\{V_P(t), t \in [0,\infty)\} \ge_{\text{st}} \{V_\pi(t), t \in [0,\infty)\}. \tag{5}$$

We use coupling and forward induction to prove Lemma 2. For any work-conserving policy π, suppose that stochastic processes $\hat{V}_P(t)$ and $\hat{V}_\pi(t)$ have the same stochastic laws as $V_P(t)$ and $V_\pi(t)$. The state processes $\hat{V}_P(t)$ and $\hat{V}_\pi(t)$ are coupled in the following manner: Consider the packet delivery events during the evolutions of $\hat{V}_P(t)$ and $\hat{V}_\pi(t)$. If the packet with time stamp $\hat{\alpha}_{i,P}(t)$ is completed at time t as $\hat{V}_P(t)$ evolves, then the packet with time stamp $\hat{\alpha}_{i,\pi}(t)$ is completed at time t as $\hat{V}_P(t)$ evolves. Such a coupling is valid since the service time is exponentially distributed and thus memoryless. According to Theorem 6.B.30 in [15], if we can show

$$P\left[\hat{V}_P(t) \ge \hat{V}_\pi(t), t \in [0,\infty)\right] = 1, \tag{6}$$

then (5) is proven. Next, we use the following two lemmas to prove (6):

Lemma 3. Suppose that under policy P, $\{\hat{U}'_P, \hat{\alpha}'_{1,P}, \dots, \hat{\alpha}'_{m,P}\}$ is obtained by delivering a packet with time stamp $\hat{\alpha}_{l,P}$ to the destination in the system whose state is $\{\hat{U}_P, \hat{\alpha}_{1,P}, \dots, \hat{\alpha}_{m,P}\}$. Further, suppose that under policy π, $\{\hat{U}'_\pi, \hat{\alpha}'_{1,\pi}, \dots, \hat{\alpha}'_{m,\pi}\}$ is obtained by delivering a packet with time stamp $\hat{\alpha}_{l,\pi}$ to the destination in the system whose state is $\{\hat{U}_\pi, \hat{\alpha}_{1,\pi}, \dots, \hat{\alpha}_{m,\pi}\}$. If

$$\hat{U}_P \ge \hat{U}_\pi, \hat{\alpha}_{i,P} \ge \hat{\alpha}_{i,\pi}, \quad \forall i = 1, \dots, m,$$

then,

$$\hat{U}'_P \ge \hat{U}'_\pi, \hat{\alpha}'_{i,P} \ge \hat{\alpha}'_{i,\pi}, \quad \forall i = 1, \dots, m. \tag{7}$$

Proof. See Appendix A. □

Lemma 4. Suppose that under policy P, $\{\hat{U}'_P, \hat{\alpha}'_{1,P}, \dots, \hat{\alpha}'_{m,P}\}$ is obtained by adding a fresh packet with time stamp s to the system whose state is $\{\hat{U}_P, \hat{\alpha}_{1,P}, \dots, \hat{\alpha}_{m,P}\}$. Further, suppose that under policy π, $\{\hat{U}'_\pi, \hat{\alpha}'_{1,\pi}, \dots, \hat{\alpha}'_{m,\pi}\}$ is obtained by adding a fresh packet with time stamp s to the system whose state is $\{\hat{U}_\pi, \hat{\alpha}_{1,\pi}, \dots, \hat{\alpha}_{m,\pi}\}$. If

$$\hat{U}_P \ge \hat{U}_\pi, \hat{\alpha}_{i,P} \ge \hat{\alpha}_{i,\pi}, \quad \forall i = 1, \dots, m,$$

then

$$\hat{U}'_P \ge \hat{U}'_\pi, \hat{\alpha}'_{i,P} \ge \hat{\alpha}'_{i,\pi}, \quad \forall i = 1, \dots, m. \tag{8}$$

Proof. See Appendix B. □

Proof of Lemma 2. For any sample path, we have that $\hat{U}_P(0^-) = \hat{U}_\pi(0^-)$ and $\hat{\alpha}_{i,P}(0^-) = \hat{\alpha}_{i,\pi}(0^-)$ for $i = 1, \dots, m$. This, together with Lemma 3 and 4, implies that

$$\hat{U}_P(t) \ge \hat{U}_\pi(t), \hat{\alpha}_{i,P}(t) \ge \hat{\alpha}_{i,\pi}(t)$$

holds for all $t \in [0,\infty)$ and $i = 1, \dots, m$. Hence, (6) follows.

Because $\{\hat{V}_P(t), t \in [0,\infty)\}$ and $\{V_P(t), t \in [0,\infty)\}$ are of the same distribution, $\{\hat{V}_\pi(t), t \in [0,\infty)\}$ and $\{V_\pi(t), t \in [0,\infty)\}$ are of the same distribution, by Theorem 6.B.30 in [15] we get (5). This completes the proof. □

Proof of Theorem 1. As a result of Lemma 2, we have

$$\{U_P(t), t \in [0,\infty)\} \ge_{\text{st}} \{U_\pi(t), t \in [0,\infty)\},\ \forall \pi \in \Pi_{wc},$$

which implies

$$\{\Delta_P(t), t \in [0,\infty)\} \le_{\text{st}} \{\Delta_\pi(t), t \in [0,\infty)\},\ \forall \pi \in \Pi_{wc}.$$

Finally, since the service times are *i.i.d.* across time and servers, service idling only increases the waiting time of the packet in the system. Therefore, the age under non-work-conserving policies will be greater. As a result, we have

$$\{\Delta_P(t), t \in [0,\infty)\} \le_{\text{st}} \{\Delta_\pi(t), t \in [0,\infty)\}, \forall \pi \in \Pi.$$

This completes the proof. □

As a result of Theorem 1, we can deduce the following corollary:

Corollary 5. If the packet service times are *i.i.d.* exponentially distributed across time and servers, then for all arrival time sequence $s_1, s_2, \ldots$, buffer size B, $\pi \in \Pi$, and non-decreasing function g,

$$\lim_{T\to\infty} \frac{1}{T}\int_0^T \Delta_{\text{prmp-LCFS}}(t)dt \leq_{\text{st}} \lim_{T\to\infty} \frac{1}{T}\int_0^T \Delta_{\pi}(t)dt,$$

$$\lim_{K\to\infty} \frac{1}{K}\sum_{k=1}^{K} A_{k,\text{prmp-LCFS}} \leq_{\text{st}} \lim_{K\to\infty} \frac{1}{K}\sum_{k=1}^{K} A_{k,\pi},$$

$$\lim_{T\to\infty} \frac{1}{T}\int_0^T g(\Delta_{\text{prmp-LCFS}}(t))dt \leq_{\text{st}} \lim_{T\to\infty} \frac{1}{T}\int_0^T g(\Delta_{\pi}(t))dt,$$

where the limits are assumed to exist.[1]

Hence, the preemptive LCFS policy minimizes time-average age, average peak age, and time-average age penalty for any non-decreasing penalty function g. Finally, the delay and throughput optimality of the preemptive LCFS policy is stated as follows:

Theorem 6. If the packet service times are *i.i.d.* exponentially distributed across time and servers, then for any arrival time sequence $s_1, s_2, \ldots$ and infinite buffer size $B = \infty$, the preemptive LCFS policy is throughput-optimal and mean-delay-optimal among all policies in Π.

In particular, any work-conserving policy is throughput optimal and mean-delay-optimal. The proof of this theorem is omitted due to space limitation.

IV. Numerical Results

We present some numerical results to illustrate the age and throughput performance of different policies. The packet service times are exponentially distributed with mean $1/\mu = 1$, which is *i.i.d.* across time and servers. The inter-arrival times are *i.i.d.* Erlang-2 distribution with mean $1/\lambda$. The number of servers is m. Hence, the traffic intensity is $\rho = \lambda/m\mu$. The queue buffer size is denoted as B, which is a non-negative integer.

Figure 3 illustrates the time-average age versus ρ for an information-update system with $m = 1$ server. One can observe that the preemptive LCFS policy achieves a better age performance than the FCFS policy analyzed in [4], and the non-preemptive LCFS policy with buffer size $B = 1$ [9] which was also named "M/M/1/2*" in [7]. Note that in these prior studies, the time-average age was characterized only for the special case of Poisson arrival process.

Figure 4 plots the time-average age versus ρ for an information-update system with $m = 2$ servers. We found that the age performance of each policy is better than that in Fig. 3, because of the diversity provided by two servers. In addition, the preemptive LCFS policy achieves the best age performance among all plotted policies. It is important to emphasize that the age performance of the preemptive LCFS policy remains the same for any buffer size $B \geq 0$. However, the age performance of the non-preemptive LCFS policy and FCFS policy varies

[1]Please refer to Fig. 2 and Section II for the peak age $A_{k,\pi}$.

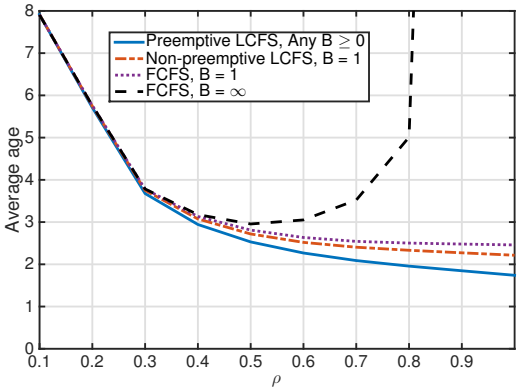

Figure 3: Average age versus traffic intensity ρ for an update system with $m = 1$ server and buffer size B.

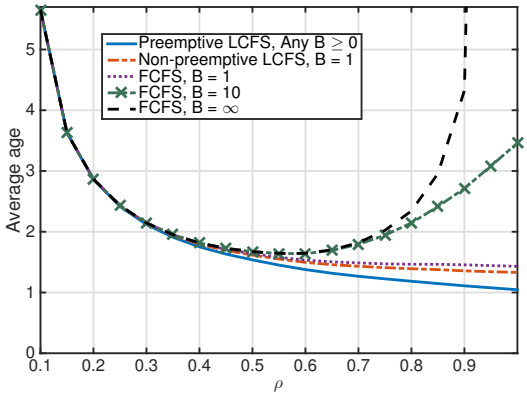

Figure 4: Average age versus traffic intensity ρ for an update system with $m = 2$ servers and buffer size B.

with the buffer size B when there are multiple servers. These numerical results validate Theorem 1.

Figure 5 depicts the throughput versus buffer size B for an information-update system with $m = 2$ servers. From the figure, we can deduce that the preemptive LCFS policy with an infinite buffer achieves the maximum throughput of 2, and it is better than the other policies. This is because other policies have a finite buffer which leads to packet dropping and throughput loss. This result is in accordance with Theorem 6 which shows that the preemptive LCFS policy can simultaneously maintain age and throughput optimality. The delay performance is omitted because any work-conserving policy is mean-delay-optimal.

V. Conclusions

In this paper, we considered an information-update system, in which a source sends update packets to a destination through multiple network servers. It was showed that, if the packet service times are *i.i.d.* exponentially distributed, then for any given arrival process and buffer size, the preemptive LCFS policy simultaneously optimizes the data freshness, throughput, and delay performance among all causally feasible policies. We plan to extend these results to more general system settings with general service time distribution.

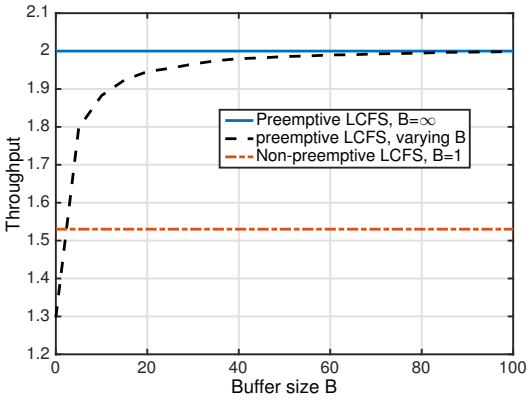

Figure 5: Throughput versus buffer size B for an update system with $m = 2$ servers.

Appendix A
Proof of Lemma 3

Since a packet with time stamp $\hat{\alpha}_{l,P}$ is delivered in policy P and a packet with time stamp $\hat{\alpha}_{l,\pi}$ is delivered in policy π, we have

$$\hat{U}'_P = \hat{\alpha}_{l,P} \geq \hat{\alpha}_{l,\pi} = \hat{U}'_\pi. \tag{9}$$

In policy P, if the queue is empty, then one server will be idle after the packet delivery. Otherwise, if there exist packets in the queue, one of these packets will be assigned to the available server. Because policy P follows a preemptive LCFS principle, the time stamp of this assigned packet must be smaller than $\hat{\alpha}_{l,P}$, and hence the packet is stale. In both cases, we will have

$$\begin{aligned} &\hat{\alpha}'_{i,P} = \hat{U}'_P, \quad i = 1, \ldots, l, \\ &\hat{\alpha}'_{i,P} = \hat{\alpha}_{i,P}, \quad i = l+1, \ldots, m. \end{aligned} \tag{10}$$

We consider two cases for policy π.

Case 1: The queue is empty or the packet which will be served is outdated. In this case, we have

$$\begin{aligned} &\hat{\alpha}'_{i,P} = \hat{U}'_P \geq \hat{U}'_\pi = \hat{\alpha}'_{i,\pi}, \qquad \text{for} \quad i = 1, \ldots, l, \\ &\hat{\alpha}'_{i,P} = \hat{\alpha}_{i,P} \geq \hat{\alpha}_{i,\pi} = \hat{\alpha}'_{i,\pi}, \quad \text{for} \quad i = l+1, \ldots, m. \end{aligned} \tag{11}$$

Case 2: The packet which will be served is not outdated. Assume this packet has a time stamp s. Since policy P is a preemptive LCFS policy, $\hat{\alpha}'_{1,p}, \hat{\alpha}'_{2,P}, \ldots, \hat{\alpha}'_{m,P}$ are the freshest packets among all packets which have arrived. As a result we have

$$\begin{aligned} &\hat{\alpha}'_{i,P} \geq \hat{\alpha}'_{i,\pi}, \quad i = 1, \ldots, m-1, \\ &\hat{\alpha}'_{m,P} \geq s = \hat{\alpha}'_{m,\pi}. \end{aligned} \tag{12}$$

Hence, (7) holds, which complete the proof.

Appendix B
Proof of Lemma 4

Since policy P is a preemptive policy, this packet will be immediately assigned to a server. If one server is idle, then the new packet will be assigned to an idle server; otherwise, if all servers are busy, then the new packet will preempt the packet with minimum time stamp $\alpha_{1,P}$. The preempted packet will be stored back to the queue if the queue is not full, and will be discarded if the queue is full. In both cases, the system state will not be affected. Since $s \geq \hat{\alpha}_{i,P}$ for $i = 1, \ldots, m$, we have

$$\begin{aligned} &\hat{\alpha}'_{i,P} = \hat{\alpha}_{i+1,P} \quad i = 1, \ldots, m-1, \\ &\hat{\alpha}'_{m,P} = s. \end{aligned} \tag{13}$$

For policy π, we consider two cases:

Case 1: One of the servers is idle, then the new packet will be assigned to an idle server. Since $s \geq \hat{\alpha}_{i,\pi}$ for all $i = 1, \ldots, m$, after the assignment we will have

$$\begin{aligned} &\hat{\alpha}'_{m,P} = s = \hat{\alpha}'_{m,\pi}, \\ &\hat{\alpha}'_{i,P} = \hat{\alpha}_{i+1,P} \geq \hat{\alpha}_{i+1,\pi} = \hat{\alpha}'_{i,\pi}, \quad i = 1, \ldots, m-1, \end{aligned} \tag{14}$$

Case 2: All servers are busy. If policy π is non-preemptive policy, then the new packet will be stored in the queue if the queue is not full. Otherwise, the new packet will either be discarded or replace one of the existed packets in the queue. As a result, $\hat{\alpha}_{i,\pi} = \hat{\alpha}'_{i,\pi}$ for all i. Using (13), we get

$$\begin{aligned} &\hat{\alpha}'_{i,P} = \hat{\alpha}_{i+1,P} \geq \hat{\alpha}_{i+1,\pi} \geq \hat{\alpha}_{i,\pi} = \hat{\alpha}'_{i,\pi}, \quad i = 1, \ldots, m-1, \\ &\hat{\alpha}'_{m,P} = s > \hat{\alpha}_{m,\pi} = \hat{\alpha}'_{m,\pi}. \end{aligned} \tag{15}$$

On the other hand, if policy π is a preemptive policy, then this fresh packet preempts one packet under service. Suppose that the preempted packet has a time stamp $\hat{\alpha}_{l,\pi}$, then we have

$$\begin{aligned} &\hat{\alpha}'_{i,P} = \hat{\alpha}_{i+1,P} \geq \hat{\alpha}_{i+1,\pi} \geq \hat{\alpha}_{i,\pi} = \hat{\alpha}'_{i,\pi}, \quad i = 1, \ldots, l-1, \\ &\hat{\alpha}'_{i,P} = \hat{\alpha}_{i+1,P} \geq \hat{\alpha}_{i+1,\pi} = \hat{\alpha}'_{i,\pi}, \qquad i = l, \ldots, m-1, \\ &\hat{\alpha}'_{m,P} = s = \hat{\alpha}'_{m,\pi}. \end{aligned}$$

Since there is no packet delivery, we have

$$\hat{U}'_P = \hat{U}_P \geq \hat{U}_\pi = \hat{U}'_\pi. \tag{16}$$

Hence, (8) holds, which complete the proof.

References

[1] B. Adelberg, H. Garcia-Molina, and B. Kao, "Applying update streams in a soft real-time database system," in *ACM SIGMOD Record*. ACM, 1995, vol. 24, pp. 245–256.

[2] J. Cho and H. Garcia-Molina, "Synchronizing a database to improve freshness," in *Acm Sigmod Record*. ACM, 2000, vol. 29, pp. 117–128.

[3] L. Golab, T. Johnson, and V. Shkapenyuk, "Scheduling updates in a real-time stream warehouse," in *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. IEEE, 2009, pp. 1207–1210.

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

[5] R. Yates and S. Kaul, "Real-time status updating: Multiple sources," in *Proc. IEEE Int. Symp. Inform. Theory*, July 2012.

[6] L. Huang and E. Modiano, "Optimizing age-of-information in a multi-class queueing system," in *Proc. IEEE Int. Symp. Inform. Theory*, 2015.

[7] M. Costa, M. Codreanu, and A. Ephremides, "Age of information with packet management," in *Proc. IEEE Int. Symp. Inform. Theory*, June 2014, pp. 1583–1587.

[8] N. Pappas, J. Gunnarsson, L. Kratz, M. Kountouris, and V. Angelakis, "Age of information of multiple sources with queue management," in *Proc. IEEE ICC*, June 2015, pp. 5935–5940.

[9] S. Kaul, R. Yates, and M. Gruteser, "Status updates through queues," in *Conf. on Info. Sciences and Systems*, Mar. 2012.

[10] C. Kam, S. Kompella, and A. Ephremides, "Effect of message transmission diversity on status age," in *Proc. IEEE Int. Symp. Inform. Theory*, June 2014, pp. 2411–2415.

[11] M. Costa, S. Valentin, and A. Ephremides, "On the age of channel information for a finite-state markov model," in *Proc. IEEE ICC*, June 2015, pp. 4101–4106.

[12] T. Bacinoglu, E. T. Ceran, and E. Uysal-Biyikoglu, "Age of information under energy replenishment constraints," in *Proc. Info. Theory and Appl. Workshop*, Feb. 2015.

[13] R. Yates, "Lazy is timely: Status updates by an energy harvesting source," in *Proc. IEEE Int. Symp. Inform. Theory*, 2015.

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

[15] M. Shaked and J. G. Shanthikumar, *Stochastic orders*, Springer Science & Business Media, 2007.