1 Age-Efficient Scheduling in Communication Networks

Bin Li, Pennsylvania state university, University Park, PA, USA
Bo Ji, Virginia Tech, Blacksburg, VA, USA
Atilla Eryilmaz, Ohio State University, Columbus, OH, USA

Age-of-information (AoI) is used to characterize the freshness of information, and is critical for information monitoring, tracking, and control, which is typically required in many network applications, such as autonomous vehicles, virtual/augmented reality, and Internet-of-Things (IoT). Both inter-arrival times and delays of packets affect AoI performance, and thus traditional delay-efficient algorithms do not necessarily exhibit low AoI performance. This calls for "age-efficient" algorithm design in communication networks, which forms the focus of this chapter. In particular, we first discuss the recent advances in the age-efficient algorithm design for three different types of common network traffic: (i) elastic traffic (cf. Section 1.1): packets are allowed to be delivered without any specific deadline constraints; (ii) inelastic traffic (cf. Section 1.2): packets will be dropped if they are not delivered within a specific deadline; (iii) heterogeneous traffic (cf. Section 1.3): different packets may have different size. To facilitate our discussions, we explicitly consider the discrete-time model and emphasize the difference between age-efficient and delay-efficient algorithm design paradigms. Then, we examine "fresh" scheduling design for remote estimation with the goal of optimally balancing the tradeoff between the estimation accuracy and the communication cost (cf. Section 1.4).

1.1 Age-Efficient Scheduling for Elastic Traffic

In this section, we focus on the age-efficient scheduling design for elastic traffic, where packets are allowed to be delivered without any specific deadline constraints. We first consider a simple Geo/Geo/1 queue to illustrate the difference between the AoI metrics and the traditional delay metric (i.e., average delay). Then, we consider the age-efficient scheduling algorithm design in wireless networks.

1.1.1 Delay and AoI Analysis in the Geo/Geo/1 Queue

We consider an infinite-buffer single server queue. In each time slot, a packet arrives at the queue with probability $\lambda \in (0, 1]$, and a packet at the front of the

queue is served with probability $\mu \in (0, 1]$. In such a queue, both inter-arrival and inter-service times are geometrically distributed, so the queue is called a Geo/Geo/1 queue.

Let $\Delta[t]$ and $H[t]$ be the AoI associated with the receiver of the single server queue and the delay of the Head-of-Line (HoL) packet at the queue in time slot t, respectively, where $\Delta[t]$ is the time since the generated time of the latest packet received by the receiver. Noting that the HoL delay measures the duration from the time that the HoL packet was generated to the current time and that at most one packet can be served in each time slot, we have $\Delta[t+1] = H[t] + 1$ whenever a packet is served in time slot t and $\Delta[t+1] = \Delta[t] + 1$ otherwise, i.e.,

$$\Delta[t+1] = \begin{cases} H[t] + 1, & \text{if a packet is served in time slot } t; \\ \Delta[t] + 1, & \text{otherwise.} \end{cases} \tag{1.1}$$

Fig. 1.1 shows one sample path of the AoI.

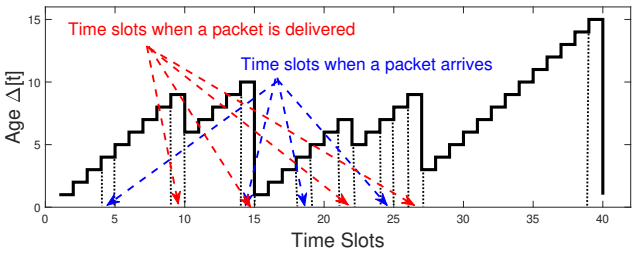


Figure 1.1 The evolution of the AoI.

The number of packets or the queue length in the Geo/Geo/1 queue evolves as a Markov chain, which is shown in Fig. 1.2. Here, $\alpha \triangleq \Pr\{\text{one arrival, no}$

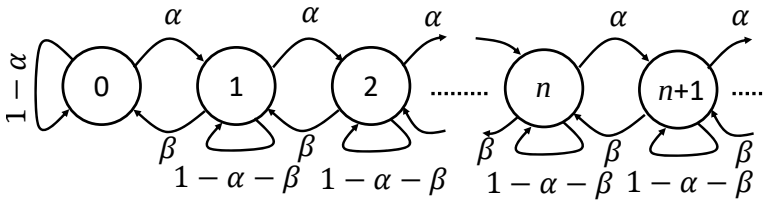


Figure 1.2 The Markov chain of a Geo/Geo/1 queue.

departure$\}$ is the probability that the queue length increases by one and $\beta \triangleq \Pr\{\text{no arrival, one departure}\}$ is the probability that the queue length decreases by one. From this Markov Chain, we can calculate the average delay, average peak AoI, and average AoI, as shown in the following theorem.

THEOREM 1.1 *If $\lambda < \mu$, then average delay $\overline{D}$, average peak age $\overline{\Delta}^{(p)}$, and*

average age $\overline{\Delta}^{(avg)}$ *are expressed as follows:*

$$\overline{D} = \frac{1-\lambda}{\mu-\lambda}, \tag{1.2}$$

$$\overline{\Delta}^{(p)} = \frac{\mu-\lambda^2}{\lambda(\mu-\lambda)}, \tag{1.3}$$

$$\overline{\Delta}^{(avg)} = \frac{\lambda^3+\mu^3-\lambda^3\mu-\lambda^2\mu}{\lambda\mu^2(\mu-\lambda)}. \tag{1.4}$$

Proof Let π_n denote the steady-state distribution of the Markov Chain. According to the local balance equation and the normalizing condition (i.e., $\sum_{n=0}^{\infty}\pi_n = 1$), we can obtain π_n:

$$\pi_n = \gamma^n(1-\gamma), \quad n = 1, 2, 3, \ldots \tag{1.5}$$

where $\gamma \triangleq \alpha/\beta = \lambda(1-\mu)/(\mu(1-\lambda))$. Since $\lambda < \mu$, we have $\gamma < 1$ and thus the mean queue length can be calculated as follows:

$$\mathbb{E}[Q] = \sum_{n=0}^{\infty} n\pi_n = (1-\gamma)\sum_{n=0}^{\infty} n\gamma^n = \frac{\gamma}{1-\gamma} = \frac{\lambda(1-\mu)}{\mu-\lambda}. \tag{1.6}$$

The average delay of an incoming packet consists of the average queueing delay and its average service time, where the average queueing delay is the total average service time of all existing packets in the system. According to the BASTA (Bernoulli arrivals See Time Averages) property (Srikant & Ying 2013, Section 3.4.3), the average delay $\overline{D}$ is expressed as follows:

$$\overline{D} = \mathbb{E}[Q]\mathbb{E}[S] + \mathbb{E}[S] = \frac{\lambda(1-\mu)}{\mu-\lambda}\frac{1}{\mu} + \frac{1}{\mu} = \frac{1-\lambda}{\mu-\lambda}, \tag{1.7}$$

where S is the service time that is identically and independently geometrically distributed with mean $1/\mu$, and we use the fact that all packets experience identical and independent service time distribution.

Since the average peak age is the sum of the average delay and the average inter-arrival time in a First-Come-First-Serve queue, the average peak age can be calculated as follows:

$$\overline{\Delta}^{(p)} = \overline{D} + \mathbb{E}[I_A] = \frac{1-\lambda}{\mu-\lambda} + \frac{1}{\lambda} = \frac{\mu-\lambda^2}{\lambda(\mu-\lambda)}, \tag{1.8}$$

where I_A is the inter-arrival time that is i.i.d. geometrically distributed with mean $1/\lambda$. The calculation of the average age $\overline{\Delta}^{(avg)}$ is much more involved and is available in (Talak, Karaman & Modiano 2019, Theorem 5). □

In the light traffic regime (i.e., $\lambda \downarrow 0$), the average delay is close to $1/\mu$, which is the average service time of a packet and thus is the minimum delay experienced by a packet. In contrast, both average peak age and average age blow up to infinity, since the inter-arrival time is extremely large, and thus, it takes a really long time to receive a packet update. On the other hand, in the heavy traffic

regime (i.e., $\lambda \uparrow \mu$), all these three metrics are dominated by the queueing delay and thus increase to infinity.

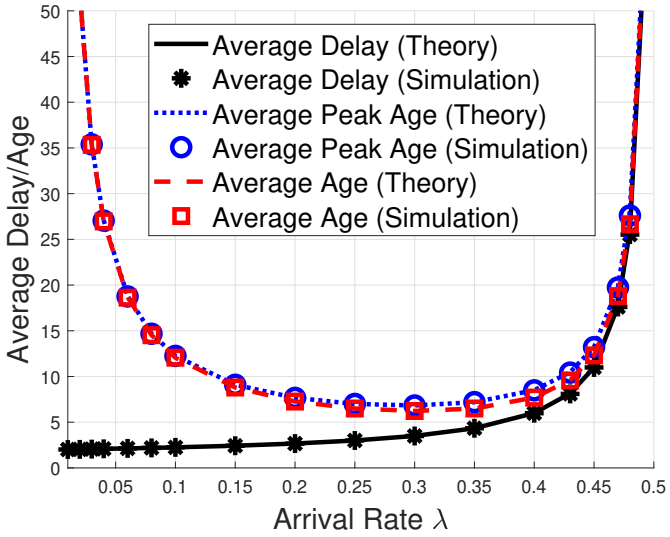


Figure 1.3 Average delay/age in the Geo/Geo/1 queue.

We verify these theoretical results via simulations with $\mu = 0.5$. We plot both theoretical and simulated results with varying arrival rate λ. From Fig. 1.3, we can observe that simulated values exactly match their corresponding theoretical results, which validates the correctness of these theoretical results. In addition, we can observe that in the light traffic regime, the average delay is quite small, while both the average peak age and average age are rather large. In contrast, all three metrics are large in the heavy traffic regime.

From the AoI analysis of the Geo/Geo/1 queue, we learned that the AoI performance is poor in both low-rate and high-rate regimes. This implies that an age-efficient scheduling design should take into account not only the queueing congestion as in the traditional delay-efficient scheduling algorithms but also the arrival rates. We will discuss the age-efficient scheduling design in the next subsection.

1.1.2 Age-Efficient Scheduling Algorithms

In this section, we consider the age-efficient scheduling design for elastic traffic in wireless networks. We first introduce the network model and then develop an age-efficient scheduling algorithm. Finally, we demonstrate the efficiency of the proposed scheduling algorithm via various simulations.

System Model: We consider a wireless network with N links, where each link represents a pair of transmitter and receiver within the transmission range of

each other. Due to the wireless interference, the success or failure of transmission over a link depends on whether one of its interfering links is also active in the same time slot, commonly called the *link-based conflict model*. Let $S_n[t] = 1$ denote that link n is scheduled in time slot t and $S_n[t] = 0$ otherwise. We call $\mathbf{S}[t] \triangleq (S_n[t])_{n=1}^N$ the *feasible schedule* denoting the set of links that can be active simultaneously in time slot t. Let $\mathcal{S}$ be the collection of all feasible schedules. Fig. 1.4 shows a network with 4 wireless links, where the neighboring links (e.g., link 1 and link 2) interfere with each other and thus cannot be scheduled at the same time. In such a network, either one link or two links can be scheduled at the same time, and none of the three links can be simultaneously active. Here, $(0,0,0,0)$, $(1,0,0,0)$, $(1,0,1,0)$, $(0,1,0,1)$ are feasible schedules, while $(1,1,1,0)$ and $(1,1,0,1)$ are not feasible.

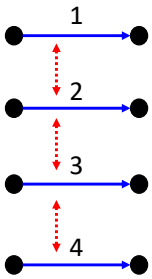


Figure 1.4 A wireless network with 4 links: a line denotes a transmitter-receiver pair and a dash line means that there is an interference between two links.

We use $A_n[t]$ to denote the number of packets arriving at link n in time slot t that is independently distributed over links, and independently and identically distributed (i.i.d.) over time with finite mean $\lambda_n > 0$. We assume that each link n independently experiences i.i.d. ON-OFF channel fading over time with $C_n[t] = 1$ denoting that the channel is available for link n in time slot t. We maintain a queue at each link and assume that packets are served in the order that they arrive, i.e., First-Come-First-Served (FCFS). Let $Q_n[t]$ be the queue length at link n in time slot t, denoting the number of packets waiting for transmission, and thus its evolution can be described as follows:

$$Q_n[t+1] = \max\{Q_n[t] + A_n[t] - C_n[t]S_n[t], 0\}, \tag{1.9}$$

where the maximum operation is due to the fact that the queue length is non-negative.

Let $\Delta_n(t)$ and $H_n(t)$ be the age associated with the receiver of link n and the delay of the Head-of-Line (HoL) packet at link n in time slot t, respectively. Noting that the HoL delay measures the duration from the time that the packet was generated to the current time and that at most one packet can transmit over a link in each time slot, we have $\Delta_n[t+1] = H_n[t] + 1$ whenever a packet is successfully delivered over link n in time slot t and $\Delta_n[t+1] = \Delta_n[t] + 1$

otherwise, i.e.,

$$\Delta_n[t+1] = \begin{cases} \Delta_n[t]+1, & \text{if } S_n[t]C_n[t]=0; \\ H_n[t]+1, & \text{if } S_n[t]C_n[t]=1. \end{cases} \tag{1.10}$$

We call the system *stable* if the average queue lengths of all links are finite, i.e., $\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[Q_n[t]]<\infty, \forall n=1,2,\ldots,N$. The *capacity region* Λ is defined as the maximal set of the achievable link rates in packets per slot that can be supported by the network under stability for a given interference model. For example, in the wireless network shown in Fig. 1.4, the capacity region is $\Lambda=\{(\lambda_n)_{n=1}^4 : \lambda_1+\lambda_2\le 1, \lambda_2+\lambda_3\le 1, \lambda_3+\lambda_4\le 1\}$. We say that a scheduler is *throughput-optimal* if it achieves the network stability for any arrival rate vector $\boldsymbol{\lambda}\triangleq(\lambda_n)_{n=1}^N$ that lies strictly inside the capacity region Λ. We are interested in developing throughput-optimal algorithms that achieve low AoI performance (i.e., low average peak age and average age).

Algorithm Design: As we mentioned in the last section, the age-efficient scheduling design should take both queueing congestion and arrival rate into account. On the one hand, the smaller the arrival rate of a link, the larger the AoI of this link, and thus the scheduling priority should be given to the link with a low arrival rate. On the other hand, the queueing delay dominates the AoI performance in the heavy-traffic regime, and thus the scheduling priority should be given to the congested links (i.e., links with large HoL delays). These two observations naturally motivate the following scheduling algorithm (see also (Joo & Eryilmaz 2018)).

(Scaled HoL-Delay-based MaxWeight (SHD-MW) Algorithm). In each time slot t, given the channel state $(C_n[t])_{n=1}^N$, select a feasible schedule $\mathbf{S}^*[t]\in\mathcal{S}$ such that

$$\mathbf{S}^*[t]\in\arg\max_{\mathbf{S}[t]\in\mathcal{S}}\sum_{n=1}^{N}\frac{1}{\lambda_n}H_n[t]C_n[t]S_n[t]. \tag{1.11}$$

In the SHD-MW Algorithm, we require the knowledge of the arrival rate of each link, which is usually not available. However, we can use the estimate $\widetilde{\lambda}_n[t]=(1-1/t)\widetilde{\lambda}[t-1]+A_n[t]/t$ to replace λ_n. According to the Law of Large Numbers, $\widetilde{\lambda}[t]\to\lambda_n$ as $t\to\infty$.

It has been well-known that the HoL-Delay-based scheduling algorithm possesses not only the throughput optimality (i.e., achieving stability for any arrival rate vector within the capacity region) and but also possess good delay performance (see (McKeown, Mekkittikul, Anantharam & Walrand 1999)). Therefore, the SHD-MW Algorithm is expected to be throughput-optimal and delay-efficient. Due to the fact that the mean peak age of each link is the sum of its average delay and average inter-arrival time when packets are served in FCFS,

the SHD-MW Algorithm exhibits excellent AoI performance, which is demonstrated via simulations.

Simulation Results: We evaluate the performance of the SHD-MW Algorithm and compare it with the following existing policies:

(i) Queue-Length-based MaxWeight (Q-MW) Algorithm (Tassiulas & Ephremides 1990): Select a feasible schedule satisfying

$$\mathbf{S}^{(\text{Q-MW})}[t] \in \arg\max_{\mathbf{S}[t]\in\mathcal{S}} \sum_{n=1}^{N} Q_n[t]C_n[t]S_n[t].$$

(ii) HoL-Delay-based MaxWeight (HD-MW) Algorithm (McKeown et al. 1999): Select a feasible schedule satisfying

$$\mathbf{S}^{(\text{HD-MW})}[t] \in \arg\max_{\mathbf{S}[t]\in\mathcal{S}} \sum_{n=1}^{N} H_n[t]C_n[t]S_n[t].$$

(iii) Age-based MaxWeight (A-MW) Algorithm (Sun, Kadota, Talak & Modiano 2019): Select a feasible schedule satisfying

$$\mathbf{S}^{(\text{A-MW})}[t] \in \arg\max_{\mathbf{S}[t]\in\mathcal{S}} \sum_{n=1}^{N} h(\Delta_n[t])C_n[t]S_n[t],$$

where $h(x)$ takes a linear form (i.e., $h(x) = x$) or quadratic form (i.e., $h(x) = x^2 + x$).

(iv) Largest-Age-Drop-based MaxWeight (LAD-MW) algorithm (Sun et al. 2019): Select a feasible schedule satisfying

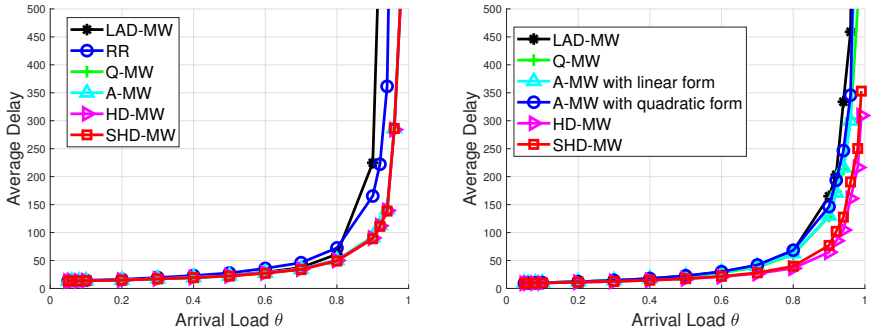
$$\mathbf{S}^{(\text{LAD-MW})}[t] \in \arg\max_{\mathbf{S}[t]\in\mathcal{S}} \sum_{n=1}^{N} (\Delta_n[t] - H_n[t])C_n[t]S_n[t].$$

(v) Round-Robin (RR) Algorithm: in the case of fully-connected networks (i.e., at most one link can be scheduled in each time slot), it serves links in turn by simply skipping transmission when a channel is OFF. However, it is unclear how to define an appropriate version of the RR Algorithm in the general wireless setup.

We consider a fully-connected ON-OFF fading network with $N = 5$ links and a 3×3 switch. While algorithms were described for wireless networks, it also applies to other communication networking applications where interference-type constraints exist, such as a switch. Since switches are widely used in data centers and the Internet, we demonstrate our algorithm in that context as well. Packets arrive at link n in each time slot with probability λ'_n. Whenever arrivals happen at link n, the number of packets has the following distribution: it is equal to M_n with probability $(\eta_n - 1)/(M_n - 1)$ and 1 otherwise, where $M_n \geq 2$ is some parameter that measures the burstiness of the arrivals and η_n is the average number of arriving packets at link n when arrivals happen. In the fully-connected network, the probability vector of channel being ON is $\mathbf{p} = [0.1, 0.1, 0.8, 0.8, 0.8]$,

$\boldsymbol{\eta} = [2, 5, 5, 10, 15]$, $\mathbf{M} = [5, 10, 10, 20, 20]$, and we consider the following specific arrival rate vector: $\boldsymbol{\lambda}' = \theta \times 0.3821 \times [0.05, 0.02, 0.16, 0.08, 0.053]$, where $\theta \in (0, 1)$. It can be verified that $\boldsymbol{\lambda}' \otimes \boldsymbol{\eta}$ is within the capacity region according to (Tassiulas & Ephremides 1993, Lemma 1), where $\mathbf{x} \otimes \mathbf{y}$ denotes the component-wise product of the vector $\mathbf{x}$ and $\mathbf{y}$. In the case of the 3×3 switch, $\eta = [5, 2, 2; 3, 10, 5; 5, 4, 5]$, and $\boldsymbol{\lambda}' = \theta \times [0.14, 0.1, 0.05; 0.05, 0.06, 0.05; 0.03, 0.05, 0.13]$, where $\theta \in (0, 1)$. Again, we can verified that $\boldsymbol{\lambda}' \otimes \boldsymbol{\eta}$ is within the capacity region.

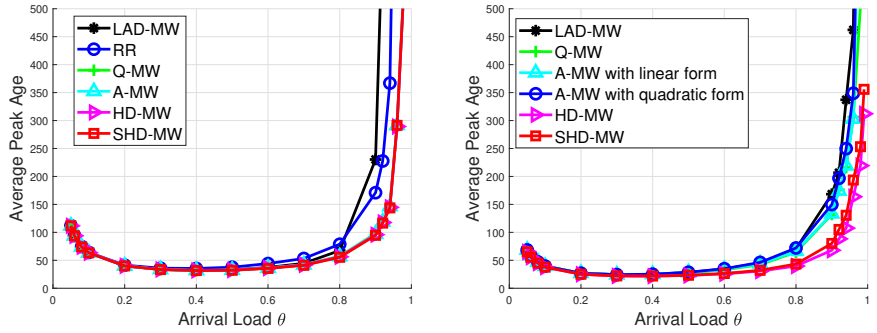
Delay Performance: Fig. 1.5 shows the average delay under our proposed SHD-MW Algorithm and other existing algorithms in both fully-connected networks and 3×3 switch. It can be observed from Fig. 1.5 that HD-MW and our SHD-MW have similar delay performance and uniformly perform better than other algorithms. In addition, SHD-MW can stabilize the system for any $\theta \in (0, 1)$ in the above network setups, which validate the throughput optimality of our proposed algorithm. However, RR and LAD-MW suffer from relatively poor delay performance, especially in the heavy-traffic regime, since they do not prioritize congested links.



(a) Fully-connected network (b) 3×3 switch

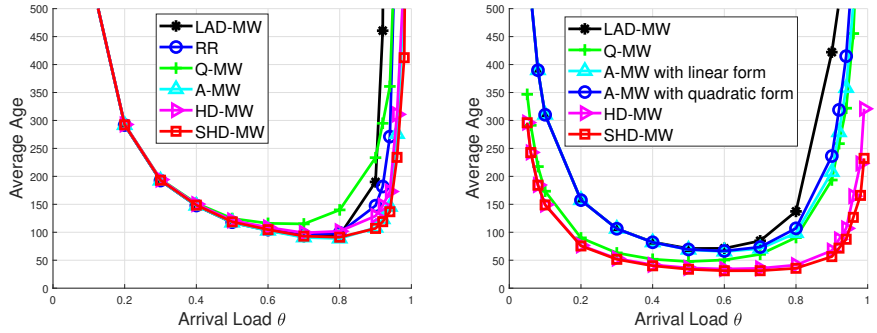
Figure 1.5 Average delay performance.

AoI Performance: We can observe from Fig. 1.6 and Fig. 1.7 that the SHD-MW performs uniformly better than all other algorithms except the HD-MW algorithm in terms of both average peak age and average age performance. Indeed, the HD-MW slightly outperforms the SHD-MW algorithm in terms of average peak age, especially in 3×3 switch in the heavy-traffic regime, as shown in Fig. 1.6b. This makes sense since the average peak age is the sum of the average HoL delay and the average inter-arrival time, and HD-MW tries to minimize the HoL delay. However, HD-MW Algorithm performs slightly worse than A-MW and SHD-MW in terms of average age performance, since it does not incorporate the inter-arrival time into the scheduling design.



(a) Fully-connected network (b) 3×3 switch

Figure 1.6 Average peak age performance.



(a) Fully-connected network (b) 3×3 switch

Figure 1.7 Average age performance.

1.2 Age-Efficient Scheduling for Inelastic Traffic

In this section, we consider the case with inelastic traffic, where packets will be dropped if they are not delivered within a specific deadline. We first consider a simple single-server queue with packet deadlines and understand the impact of system parameters on the AoI performance. Then, we study the age-efficient scheduling design for inelastic traffic.

1.2.1 AoI Analysis in the Single-Server Queue with Strict Deadlines

Different from the previous Geo/Geo/1 queueing model, here each arriving packet has a strict deadline of $T \in \{1, 2, 3, \ldots\}$, i.e., it will be dropped if it cannot be delivered within T time slots. We divide time into frames with a size of T time slots. We assume that packets arrive at the beginning of each time frame according to Bernoulli distribution with the mean of $\lambda \in (0, 1]$. In each time slot within a frame, a packet is served with the probability of $\mu \in (0, 1]$. In such a

queue, the age is equal to the number of time slots that have been passed since the packet was last delivered, i.e.,

$$\Delta[(k+1)T] = \begin{cases} \Delta[kT] + T & \text{if a packet cannot be delivered in frame } k; \\ T & \text{otherwise.} \end{cases} \tag{1.12}$$

Fig. 1.8 shows the age dynamics of the queue with deadlines. Hence, the age

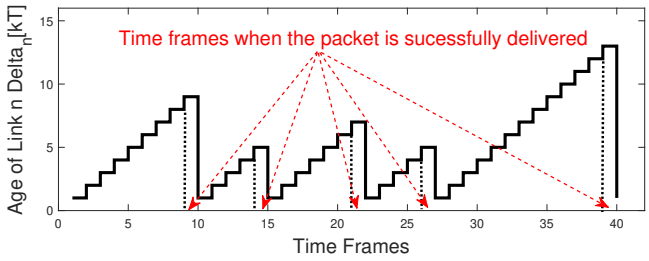


Figure 1.8 The evolution of the AoI of link n.

forms a Markov chain as illustrated in Fig. 1.9, where $q \triangleq \lambda(1-(1-\mu)^T)$ denotes the probability that a packet is successfully delivered within a frame.

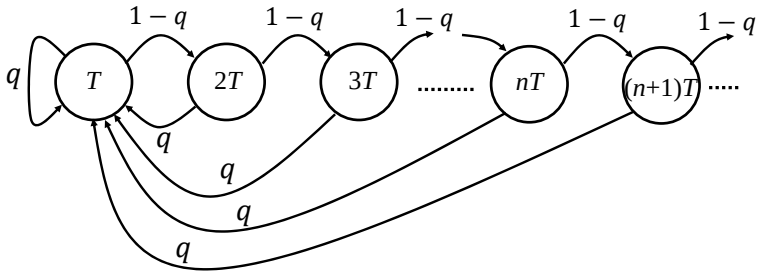


Figure 1.9 The Markov chain of a single-server queue with deadlines.

THEOREM 1.2 *The average peak age and average age are equal and can be expressed as follows.*

$$\overline{\Delta}^{(p)} = \overline{\Delta}^{(avg)} = \frac{T}{\lambda\left(1-(1-\mu)^T\right)}. \tag{1.13}$$

Proof Let π_n denote the probability of the AoI being equal to nT in the steady-state in the Markov chain shown in Fig. 1.9. According to the global balance

equations, we have

$$\pi_1 = q\sum_{n=1}^{\infty}\pi_n = q \tag{1.14}$$

$$\pi_{n-1}(1-q) = \pi_n, \quad \forall n = 2,3,4,\ldots, \tag{1.15}$$

where we use the normalizing condition $\sum_{n=1}^{\infty}\pi_n = 1$ in the first equation. Hence, we obtain π_n as follows.

$$\pi_n = q(1-q)^{n-1}, \quad \forall n = 1,2,3,\ldots \tag{1.16}$$

Therefore, the average age can be calculated as follows.

$$\overline{\Delta}^{(avg)} = \sum_{n=1}^{\infty}\pi_n \cdot nT = T\sum_{n=1}^{\infty}nq(1-q)^n = \frac{T}{q} = \frac{T}{\lambda\left(1-(1-\mu)^T\right)}. \tag{1.17}$$

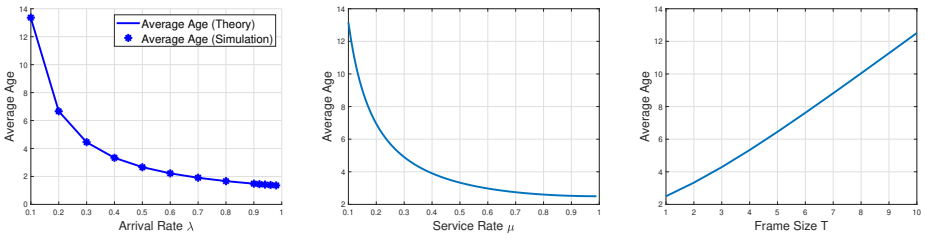
Since the peak age happens whenever there is a successful delivery (i.e., age drops), the average peak age again can be expressed as

$$\overline{\Delta}^{(p)} = \sum_{n=1}^{\infty}\pi_n \cdot nT, \tag{1.18}$$

which is equal to average age $\overline{\Delta}^{(avg)}$. □

Unlike the previous Geo/Geo/1 queue, there is no constraint on the arrival rate λ. The larger the arrival rate λ (or the service rate μ), the higher the successful packet delivery probability q and thus the better the AoI performance. On the other hand, the larger the time frame size T, both average peak age and average age increase despite the high successful packet delivery probability.

We validate our theoretical results via simulations with $\mu = 0.5$ and $T = 2$. We plot both theoretical average age and its simulated values with respect to the arrival rate λ in Fig. 1.10a. We can observe from Fig. 1.10a that the simulation result matches the theoretical result quite well, which confirms the correctness of the theoretical average age. In addition, we can see from Fig. 1.10b and Fig. 1.10c that the average age indeed decreases with respect to the service rate μ and increases with respect to the time frame size T.



(a) Validation of the average age(b) Average age vs. service rate (c) Average age vs. frame size

Figure 1.10 Average age in the single-server queue with deadlines.

From the simple analysis of the single-server queue with deadlines, we know that the successful delivery probability increases with the increase of the frame size T, which, however, deteriorates AoI performance. In practice, the frame size T is typically fixed. In the next section, we consider the age-efficient scheduling design for wireless networks with inelastic traffic.

1.2.2 Age-Efficient Deadline-Constrained Scheduling

In this section, we focus on the age-efficient scheduling design for inelastic traffic. We first present the network model and then propose an age-efficient scheduling algorithm. Finally, we validate the efficiency of the proposed algorithm via various simulations.

System Model: We consider a wireless network consisting of N links. Here, each link models a pair of transmitter and receiver that are within the transmission range of each other. Each packet has a strict deadline of T time slots, i.e., it will be dropped if it cannot be delivered within T time slots. We divide time into frames, each of which has consecutive T time slots. We assume that packets arrive at each link at the beginning of each time frame. Let $A_n[kT]$ denote the number of packets arriving at link $n \in \{1, 2, \cdots, N\}$ in frame $k \in \{0, 1, 2, \cdots\}$, which are independently distributed over links and i.i.d. over time with mean λ_n. All remaining packets are dropped at the end of a frame. Each link has a maximum allowable drop rate $\rho_n \lambda_n$, where $\rho_n \in (0, 1)$ is the maximum fraction of packets that can be dropped at link n on average.

Each link n experiences an i.i.d. ON-OFF channel fading, where the channel state is constant during a frame and can vary over frames. We use $C_n[kT]$ to capture channel fading of link n in frame k, i.e., $C_n[kT] = 1$ if the channel of link n is ON within frame k and $C_n[kT] = 0$ otherwise. Due to the wireless interference, only a subset of links can be scheduled in each time slot, called feasible schedule denoted by $\mathbf{S}[t] = (S_n[t])_{n=1}^N$, where $S_n[t] = 1$ if user n is scheduled in slot t and $S_n[t] = 0$, otherwise. We use $\mathcal{S}$ to denote the set of all feasible schedules.

Let $\Delta_n[kT]$ be the age associated with the receiver of link n in frame k, which measures the number of frames elapsed up to frame k since the most recent successful packet delivery for link n. In other words, $\Delta_n[kT]$ increases by one until link n has at least one packet delivery in frame k and then it drops to one at the beginning of the next frame.

$$\Delta_n[(k+1)T] = \begin{cases} \Delta_n[kT] + T, & \text{if } C_n[kT] \sum_{t=kT}^{(k+1)T-1} S_n[t] = 0; \\ T, & \text{if } C_n[kT] \sum_{t=kT}^{(k+1)T-1} S_n[t] > 0. \end{cases} \tag{1.19}$$

Note that the age is similar to the concept of *Time-Since-Last-Service (TSLS)* considered in (Li, Li & Eryilmaz 2014) under our system settings. Yet, the TSLS drops to zero whenever there is a packet delivery.

We provide an example in Fig. 1.11 to show an impact of real-time scheduling

design on the AoI performance. As shown in Fig. 1.11(a), the age of the user updates at the beginning of each frame. The delivery of packet a, b, and e result in the decrease of the age. If we only focus on the time window shown in the figure, Fig. 1.11(a) shows a 60% delivery ratio (3 packet deliveries over 5 packet arrivals) on average with an average age of 2.27 frames. While in Fig. 1.11(b), with the same packet arrival pattern and delivery ratio, the schedule of delivery of packet d instead of e leads to a drop of average age to 2 frames, indicating a better performance on the AoI performance.

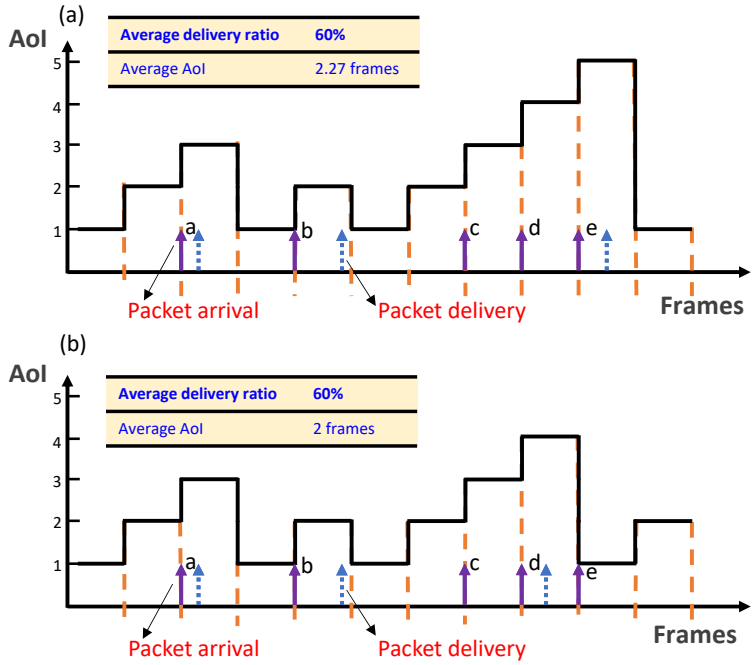


Figure 1.11 The impact of scheduling design on the AoI performance.

In (Kadota, Uysal-Biyikoglu, Singh & Modiano 2016), the authors proposed a greedy policy (AoI-Greedy) for scheduling deadline-constrained traffic in a symmetric wireless network. However, they only focused on the AoI minimization without providing any guarantees on timely throughput. In this subsection, we are interested in developing a scheduling algorithm that ensures that the average drop rate of each link should not be greater than its maximum allowable drop rate while guaranteeing the average age performance.

Algorithm Design: We use the virtual queue technique to guarantee the desired drop rate requirement. In particular, each link n maintains a virtual queue to keep track of the number of dropped packets due to deadline expiry in each frame k denoted by $Q_n[kT]$. In (Lu, Ji & Li 2018), we develop a parametric class of

maximum-weight type scheduling algorithms, where the weight of each link n consists of its own virtual queue-length $Q_n[kT]$ and age $\Delta_n[kT]$.

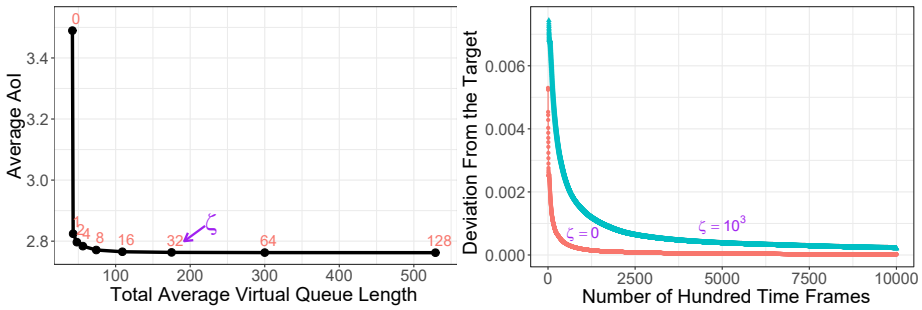
(Age-based Real-Time (AoI-RT) Algorithm). In each time frame k, given the channel state $(C_n[kT])_{n=1}^N$, select a schedule $\{\mathbf{S}^*[t]\}_{t=kT}^{(k+1)T-1}$ such that

$$\{\mathbf{S}^*[t]\}_{t=kT}^{(k+1)T-1} \in \arg\max_{\mathbf{S}[t]\in\mathcal{S}} \sum_{n=1}^{N} \left(Q_n[kT] + \zeta\beta_n\Delta_n[kT]\right) \times \min\left\{\sum_{t=kT}^{(k+1)T-1} C_n[kT]S_n[t], A_n[kT]\right\}.$$

where $\zeta \geq 0$ is some control parameter, and β_n is a parameter related to the user preference with the AoI performance.

It can be shown that the proposed algorithm is throughput-optimal in the sense that it can support any throughput that can be supported by any other algorithms. Moreover, we can derive an upper bound on a considered AoI metric, demonstrating that the network-wide data freshness is guaranteed and can be tuned under the proposed algorithm. However, the improvement of AoI performance is at the cost of slowing down the convergence of the timely throughput.

Simulation Results: Fig. 1.12a shows the trade-off between the convergence speed of timely throughput and the AoI performance. We can observe from the first figure in Fig. 1.12a that as parameter ζ increases, the value of AoI metric decreases (i.e., data freshness improves); while the total average virtual-queue length increases (i.e., the convergence of timely throughput slows down) as shown in the second figure in Fig. 1.12b. In fact, when $\zeta = 0$, the timely throughput has the fastest convergence speed. However, if the data freshness is more important to applications (as is often the case), we could trade the convergence speed of timely throughput for data freshness by increasing the parameter ζ.



(a) Average age vs. average virtual queue length (b) Convergence speed

Figure 1.12 Performance of AoI-RT Algorithm.

1.3 Age-Efficient Scheduling for Heterogeneous Traffic

In this section, we consider heterogeneous packet sizes, which can be used to model heterogeneous traffic in real-world applications, such as environmental sensing and monitoring, where the packet size depends on the actual measurements. We conduct a systematic and comparative study based on simulations to investigate the impact of scheduling policies on the AoI performance in a single-server queue. Here, scheduling policies are adopted to decide which packet in the queue will be served when the server becomes idle, so they are usually referred to as queueing disciplines.

System Model: We consider a single-sever queue with infinite buffer space. To begin with, we make the following simplifying assumptions. Packets arrive at this queue according to an *i.i.d.* Bernoulli process with parameter λ. Packet sizes follow a geometric distribution with mean $1/\mu$, and one unit of data can be served in each time slot[1]. Such a single-server queue is called a Geo/Geo/1 queue as both the inter-arrival and departure times follow a geometric distribution.

Prior studies reveal that the AoI depends on both the inter-arrival time and delay of the packets (Kaul, Yates & Gruteser 2012*b*). Although it is well-known that scheduling policies play an important role in reducing the packet delay in a single-sever queue (Harchol-Balter 2013), it remains largely unknown how exactly scheduling policies to impact the AoI performance. To that end, we aim to holistically study the impact of various aspects of scheduling policies on the AoI performance in a single-server queue and suggest guidelines that are potentially useful for the design of AoI-efficient scheduling policies in more general settings. While a similar study for the continuous-time queueing model has been conducted in (Liu, Huang, Li & Ji 2020, Liu, Huang, Li & Ji 2021, accepted), we will be focused on the discrete-time model.

While a lot of research effort has been made towards the analysis of the AoI performance, most of these studies have been focused on scheduling policies based on the packet arrival times, such as First-Come-First-Served (FCFS) and Last-Come-First-Served (LCFS), assuming that the packet-size information is unavailable (see, e.g., (Bedewy, Sun & Shroff 2016, Costa, Codreanu & Ephremides 2016, Kaul, Yates & Gruteser 2012*a*, Kam, Kompella, Nguyen & Ephremides 2015)). In some applications, such as environmental sensing and monitoring, however, the packet-size information can be obtained or fairly well estimated. It has been shown that scheduling policies that leverage the packet-size information can substantially reduce the delay, especially when the system load is high or when the packet-size variability is large (Harchol-Balter 2013). This motivates us to investigate the AoI performance of size-based policies.

Common Scheduling Policies: Following (Harchol-Balter 2013), we first give

[1] Equivalently, we can make the following assumption: packets are of unit size; one packet is served with probability μ and no packet is served otherwise. We adopt the current assumption of heterogeneous packet sizes as we will consider scheduling policies that potentially make decisions based on the packet sizes.

the definitions of several common scheduling policies that can be divided into four types: depending on whether they are size-based or not (i.e., whether a scheduling policy uses the packet-size information for making scheduling decisions or not); depending on whether they are preemptive[2] or not (i.e., whether a policy allows a packet to be stopped partway through its execution and then to be restarted at a later time without losing intermediate work).

The first type consists of policies that are non-preemptive and blind to the packet size:

- *First-Come-First-Served (FCFS)*: When the server frees up, it chooses to serve the packet that arrived first if any.
- *Last-Come-First-Served (LCFS)*: When the server frees up, it chooses to serve the packet that arrived last if any.
- *Random-Order-Service (RANDOM)*: When the server frees up, it randomly chooses one packet to serve if any.

The second type consists of policies that are non-preemptive and make scheduling decisions based on the packet size:

- *Shortest-Job-First (SJF)*: When the server frees up, it chooses to serve the packet with the smallest size if any. When there are multiple packets of the same smallest size, the tie-breaking rule randomly chooses one of them.

The third type consists of policies that are preemptive and blind to the packet size:

- *Time-Since-Last-Service (TSLS)* (Li, Li & Eryilmaz 2014): When the server frees up, it chooses to serve the packet with the largest Time-Since-Last-Service (TSLS). The TSLS of a packet is set to zero when the packet arrives, increases by one if the packet is not served, and drops to zero once the packet is served. The TSLS policy is preemptive and mimics the Round Robin policy in the discrete-time system.
- *Preemptive Last-Come-First-Served (LCFS_P)*: This is the preemptive version of LCFS. Specifically, a preemption happens when a new packet arrives.

The fourth type consists of policies that are preemptive and make scheduling decisions based on the packet size:

- *Preemptive Shortest-Job-First (SJF_P)*: This is the preemptive version of the SJF policy. Specifically, a preemption happens when there is a new packet that has the smallest size.
- *(Preemptive) Shortest-Remaining-Processing-Time (SRPT)*: When the server frees up, it chooses to serve the packet with the smallest remaining size. When there are multiple packets of the same smallest remaining size, the tie-breaking rule randomly chooses one of them. In addition, a preemption

[2] Here, we do not consider the cost of preemption.

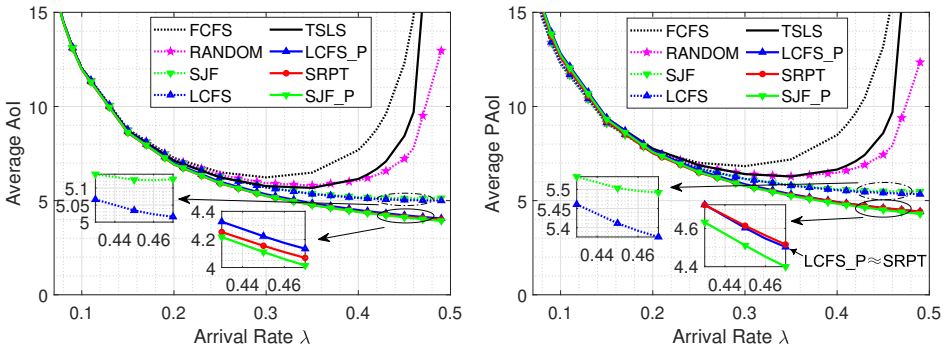
happens only when a new packet arrives, and its size is smaller than the remaining size of the packet in service.

The above common scheduling policies are summarized in Table 1.1.

	Non-preemptive	preemptive
Non-size-based	FCFS, LCFS, RANDOM	TSLS, LCFS_P
Size-based	SJF	SJF_P, SRPT

Table 1.1 Taxonomy of some common scheduling policies

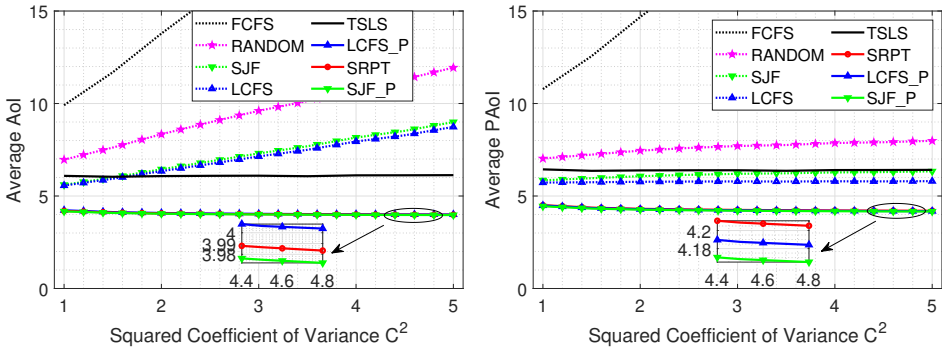
Simulation Results: To begin with, we consider a Geo/Geo/1 queue with $\lambda = 0.4$ and $\mu = 0.5$ (i.e., the mean packet size is equal to $1/\mu = 2$). By default, each data point in the plots is an average of 100 simulation runs. Each simulation run consists of 10^5 packets. In Fig. 1.13, we present the simulation results of the average AoI and average PAoI performance with varying arrival rate λ under the scheduling policies we introduced above.



(a) AoI (b) PAoI

Figure 1.13 Comparisons of the average AoI and average PAoI performance of several common scheduling policies for a Geo/Geo/1 queue with varying arrival rate λ.

We observe that for the non-preemptive case, SJF, a size-based policy, has a similar average AoI performance to LCFS but performs much better than FCFS and RANDOM, especially when the arrival rate is large. An explanation for LCFS and SJF having a similar AoI performance is the following. In the setting we consider, there is a high chance that the newest packet has the smallest size (i.e., one unit). In this case, SJF and LCFS would choose two packets of the same smallest size. They choose packets of different size only when the newest packet does not have the smallest size, which happens less often. This is why their AoI performance is similar. We also observe that LCFS performs very slightly better than SJF. The reason is that in the case that SJF and LCFS choose two packets



(a) AoI (b) PAoI

Figure 1.14 Comparisons of the average AoI and average PAoI performance of several common scheduling policies for a Geo/G/1 queue with varying squared coefficient of variance C^2.

of the same smallest size, the packet chosen by SJF is usually older due to its random tie-breaking rule.

Similarly, for the preemptive case, SJF_P and SRPT, two size-based policies, have an average AoI performance similar to LCFS_P but perform much better than TSLS, especially when the arrival rate is large. SJF_P and SRPT perform slightly better than LCFS_P because they make preemption decisions based on packet size. Moreover, we observe that preemptive policies generally have a better average AoI performance than non-preemptive ones, especially when the arrival rate is large.

Note that TSLS is also a preemptive policy. However, it performs much worse than the other preemptive policies. The reason is that TSLS mimics the Round Robin policy in an FCFS fashion. When the arrival rate becomes larger, its behavior is closer to that of FCFS, and thus, its performance becomes worse.

Similar observations can be made for the average PAoI performance.

Next, we consider a Geo/G/1 queue, where the packet size follows a Zipf distribution[3](Newman 2005). Let S be the packet size. We use $\mathbb{E}[S]$ and $\mathrm{Var}(S)$ to denote the mean and the variance of the packet size S. Then, we define the squared coefficient of variation of the packet size as $C^2 \triangleq \mathrm{Var}(S)/\mathbb{E}[S]^2$, i.e., the variance normalized by the square of the mean (Harchol-Balter 2013). A larger C^2 implies a larger variability of the packet size. While the mean packet arrival rate is fixed at $\lambda = 0.4$ and the mean packet size is fixed at $\mathbb{E}[S] = 2$, we change the value of the variance and thus the value of C^2 by choosing different values of the parameters of Zipf distribution. We present the simulation results in Fig. 1.14.

[3] The Zipf distribution comes from Zipf's law, which predicts that out of a population of N elements, the normalized frequency of elements of rank k is: $f(k;s,N) = \frac{1/k^s}{\sum_{n=1}^{N}(1/n^s)}$, where s is the value of the exponent characterizing the distribution.

We observe that the average AoI performance of preemptive policies becomes more superior as the packet-size variability (i.e., C^2) increases. Moreover, the average AoI performance of preemptive policies is only slightly impacted when the packet-size variability changes, while that of non-preemptive policies varies significantly. On the other hand, the PAoI exhibits a very different behavior for non-preemptive policies (except for FCFS). Specifically, their PAoI performance is insensitive to the packet-size variability: it either hardly changes (under LCFS and SJF) or increases very slightly (under RANDOM) as the packet-size variability increases. To the best of our knowledge, this is the first time such a counter-intuitive observation about the PAoI is reported. An explanation for this observation is the following.

First, we explain why the PAoI under FCFS is still sensitive to the packet-size variability. Note that a key difference between FCFS and other non-preemptive policies is that under FCFS, every packet will lead to an AoI drop and thus corresponds to an AoI peak. When a large packet is in service, it will block all the following packets waiting in the queue, which results in a large delay for all such packets and thus a large AoI peak corresponding to these packets. In contrast, under RANDOM, LCFS, and SJF the impact of such a blocking issue is minimal for the packets that lead to an AoI drop. Next, we explain why the PAoI has a different behavior than the AoI under RANDOM, LCFS, and SJF. We first consider LCFS. In the setting we consider, there is a high chance that the newest packet has the smallest size (i.e., one unit). Serving such unit-size packets leads to a small AoI peak (i.e., one). When the newest packet has a large size, the corresponding AoI peak would also be large. However, this happens less often. Therefore, the AoI trajectory would consist of a smaller percentage of large AoI peaks with many small AoI peaks in between. As the packet-size variability increases, there will be fewer but larger AoI peaks. In such cases, while the AoI is sensitive to the large AoI peaks (which comes from the large packet-size variability), the PAoI is much less sensitive. To illustrate this, we provide the following example. Consider a large AoI peak of 10, followed by 10 immediate small AoI peaks of 1. In this case, the average AoI is equal to $\frac{(1+2+\cdots+10)+10\times1}{10+10\times1} = 3.25$, and the average PAoI is equal to $\frac{10+10\times1}{1+10} \approx 1.82$. Now, consider a larger AoI peak of 100, followed by 100 immediate small AoI peaks of 1. Then, the average AoI is equal to $\frac{(1+2+\cdots+100)+100\times1}{100+100\times1} = 25.75$, and the average PAoI is equal to $\frac{100+100\times1}{1+100} \approx 1.98$. This example shows that a larger packet-size variability results in a larger average AoI but only minimally affects the average PAoI. A similar explanation also applies to SJF and RANDOM. SJF has a slightly worse PAoI performance due to its random tie-breaking rule. RANDOM has an even worse PAoI performance, as it randomly chooses a packet among all the packets waiting in the queue, and thus, the impact of the packet-size variability becomes more significant.

We have demonstrated that size-based policies achieve a better average AoI and average PAoI performance than non-size-based policies. However, the gain compared to the best non-size-based policies (e.g., LCFS_P) appears to be marginal.

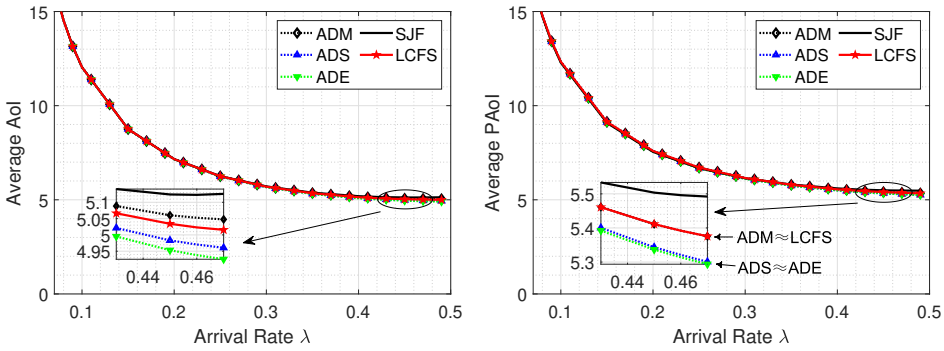
On the other hand, it is quite intuitive that when the packet-size information is unavailable, one should prioritize more recent updates. This is because while all the packets have the same mean packet size, the most recent packet arrives the last, leading to the smallest AoI once delivered.

Note that size-based policies do not utilize the arrival-time information, which also plays an important role in reducing the AoI. The packet-size information is "orthogonal" to the arrival-time information, both of which could significantly impact the AoI performance. Therefore, it is quite natural to further consider AoI-based policies that use both the size and arrival-time information of packets.

Age-based Policies: In the following, we propose three AoI-based scheduling policies, which leverage both the packet-size and arrival-time information to reduce the AoI. We begin with the definitions of three AoI-based policies that *attempt to optimize the AoI in a specific time slot in the future* from three different perspectives:

- *AoI-Drop-Earliest (ADE)*: When the server frees up, it chooses to serve a packet such that once it is delivered, the AoI drops as soon as possible.
- *AoI-Drop-to-Smallest (ADS)*: When the server frees up, it chooses to serve a packet such that once it is delivered, the AoI drops to a value as small as possible.
- *AoI-Drop-Most (ADM)*: When the server frees up, it chooses to serve a packet such that once it is delivered, the AoI drops as much as possible.

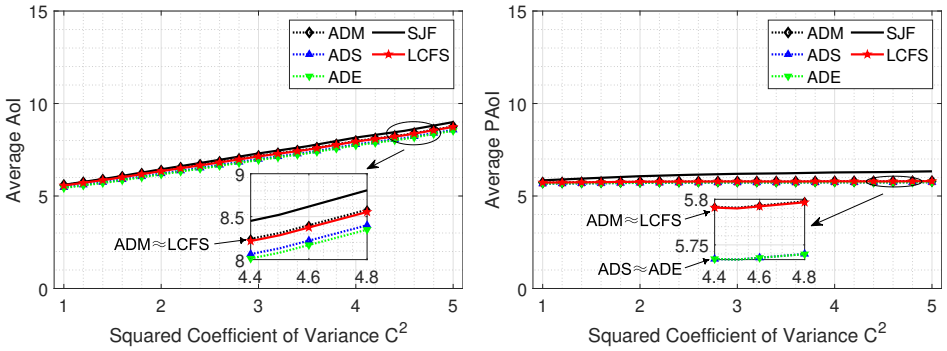
If all packets waiting in the queue are obsolete (i.e., serving such packets does not lead to an AoI drop), then the above policies choose to serve a packet with the smallest size.



(a) AoI (b) PAoI

Figure 1.15 Comparisons of the average AoI and PAoI performance of non-preemptive policies for a Geo/Geo/1 queue with varying arrival rate λ: AoI-based policies vs. non-AoI-based policies.

In Figs. 1.15 and 1.16, we present the simulation results of the average AoI and average PAoI performance of the above three AoI-based policies in comparison



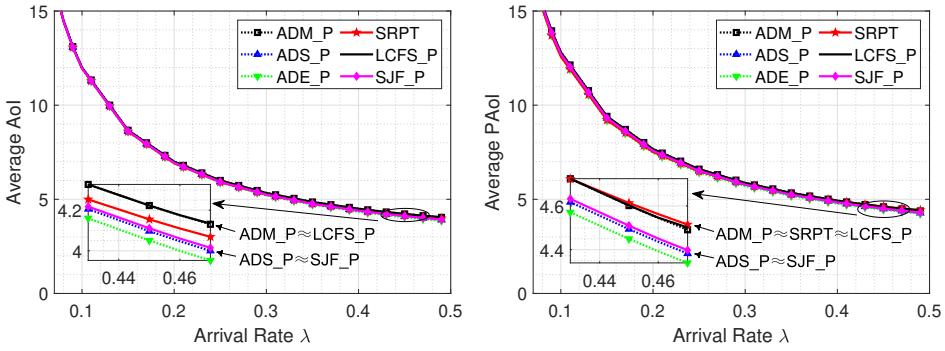
(a) AoI (b) PAoI

Figure 1.16 Comparisons of the average AoI and PAoI performance of non-preemptive policies for a Geo/G/1 queue with varying squared coefficient of variance C^2: AoI-based policies vs. non-AoI-based policies.

with two non-AoI-based policies (LCFS and SJF), with varying arrival rate λ and with varying squared coefficient of variance C^2, respectively. While ADE performs slightly better than the other policies, all the considered policies have fairly close AoI performance.

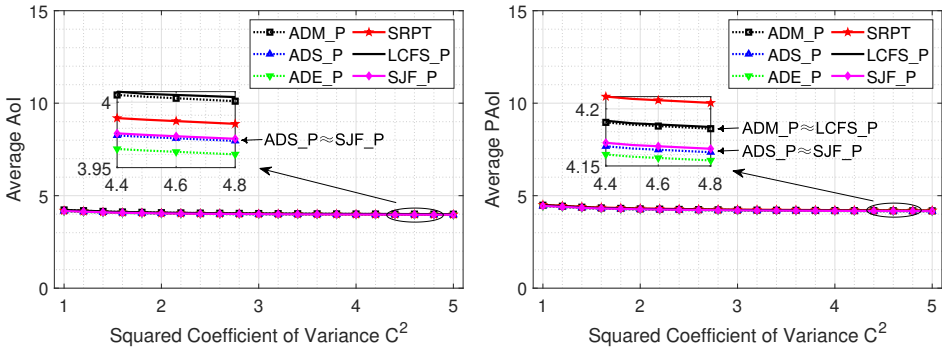
We also consider the preemptive versions of the AoI-based policies (denoted by ADE_P, ADS_P, and ADM_P) and compare them with non-AoI-based preemptive policies (LCFS_P, SJF_P, and SRPT). In Figs. 1.17 and 1.18, we present the simulation results of the average AoI and average PAoI performance of the considered policies, with varying arrival rate λ and with varying squared coefficient of variance C^2, respectively. The observations are similar to the non-preemptive case.

Remarks: In addition to the well-known guideline that one should prioritize new packets (as in LCFS), we suggest the following guidelines that will likely be useful for designing age-efficient scheduling policies. (i) Service preemption should be employed when it is allowed. (ii) When the packet-size information is available, one should prioritize packets with a small size. (iii) The AoI can be further reduced if both the packet-size and arrival-time information can be used in the design of scheduling policies (such as age-based policies). (iv) Compared to LCFS and LCFS_P, the gain of the size-based and age-based policies is rather marginal. Therefore, it is good enough to use LCFS and LCFS_P when the packet-size information is unavailable or it incurs a high computational cost to make scheduling decisions based on such packet-size information.



(a) AoI (b) PAoI

Figure 1.17 Comparisons of the average AoI and PAoI performance of preemptive policies for a Geo/Geo/1 queue with varying arrival rate λ: AoI-based policies vs. non-AoI-based policies.



(a) AoI (b) PAoI

Figure 1.18 Comparisons of the average AoI and PAoI performance of preemptive policies for a Geo/G/1 queue with varying squared coefficient of variance C^2: AoI-based policies vs. non-AoI-based policies.

1.4 Fresh Scheduling for Remote Estimation

In our discussion of the previous sections, we have predominantly focused on the timely transfer of dynamically generated and enqueued content over various circumstances and under various metrics, especially AoI, or *freshness* of updates at the receiver. A common characteristic of these investigations has been that the generation of content was not controllable by our designs, but was governed by external phenomena beyond our control. Also, the significance of a packet or file was directly measured by age metrics (e.g., mean delay, HOL-delay, mean AoI, peak-AoI, etc.). Both of these characteristics fail to hold in an important

class of scenarios that are exemplified by the remote estimation/tracking setting that we will overview in this section.

Our discussion will reveal several interesting insights in maintaining *freshness* of useful information for dynamically evolving sources. In particular, we will discuss the design of both *transmitter-driven* and *receiver-driven* update mechanisms that aim to optimally balance the tradeoff between estimation accuracy and the communication cost of updates in single-sensor (cf. Section 1.4.1) and multi-sensor (cf. Section 1.4.2) environments.

1.4.1 Single-Source Remote Estimation

The basic remote estimation problem is concerned with closely tracking the evolving state of a remote agent (such as a sensor or any other device with a dynamically changing state) with updates. This scenario is depicted in Fig. 1.19 whereby $X[t]$ is the state of the source in slot[4] t and $\hat{X}[t]$ is its estimate at the receiver at the same time. The *source* S and the *receiver* is separated by: a *transmitter* (which, in turn may be further separated into a *sampler* that decides when to sample the state, and a *queue* in to which the sampled states are enqueued for transmission) and a *communication channel* between the transmitter and the receiver.

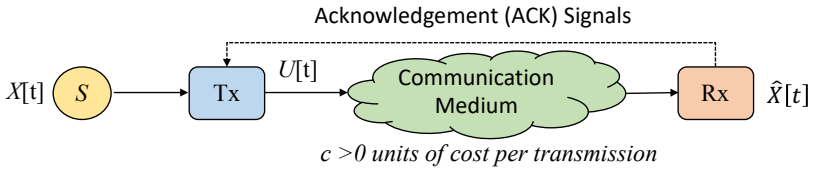


Figure 1.19 Basic remote estimation setting for a single source

The broad purpose of remote estimation is to develop sampling and transmission strategies to keep the difference between $X[t]$ and $\hat{X}[t]$ small (in terms of an appropriate measure) while also accounting for the impact and costs of the communication channel that must be utilized in the process. Therefore, this problem naturally concerns the transfer of fresh updates of the source state that is just enough to keep the estimation error low for the associated impact and cost of communication. As such, it is necessary to discuss the possible impact of the communication channel in this process.

• *Impact of the Communication Channel:* The communication channel between the transmitter and the receiver can have diverse effects on the remote sensing process. Transmissions may be erased, distorted, delayed, etc., based on the channel conditions and the choice of communication protocol. Each of these different

[4] In the spirit of the rest of the chapter, we keep the discrete-time evolution of the system dynamics here. However, the discussion can be extended to the evolution and update in continuous time.

effects may call for substantially different strategies to manage them. In our subsequent discussion, we will focus on a *cost-based* channel model by assigning a flexible *cost* value $c > 0$ to each transmission. Within this model, by choosing this cost value appropriately, we can increase or decrease the relative cost of a successful transmission under different channel impacts within a common setting. This allows us to focus on the design and analysis of policies that can optimize the tradeoff between the estimation accuracy and communication cost. Next, we make a distinction between transmitter and receiver-driven update rules.

• *Transmitter vs. Receiver-Driven Update Policies:* One important question in the remote estimation of a dynamic source is whether the update decision is made by the transmitter or the receiver. These two entities have different information on the state of the source, namely, the transmitter can be expected to know the state $X[t]$ while the receiver only has an estimate $\hat{X}[t]$. Therefore, a transmitter-driven policy can make its decisions based on the accurate knowledge of $X[t]$ while a receiver-driven policy must work with other information, such as the time-since-last-update, for its update decision. In the multi-sensor setting in Section 1.4.2, we discuss the tradeoff between these two paradigms in more detail.

In this basic setting, it is clear that the transmitter-driven update rule is better than its receiver-driven counterpart since its information is more accurate without any extra cost. The following example from (Yun, Joo & Eryilmaz 2018) summarizes useful insights on the optimal transmitter and receiver-driven design approaches for a randomly evolving source.

Example 1 (Optimal Remote Estimation of a Random Walk (Yun et al. 2018)) Suppose we aim to remotely track a state $X[t]$ that evolves as a random walk according to $X[t+1] = X[t] + W[t]$, where $\{W[t]\}_t$ are i.i.d. random variables with

$$W[t] = \begin{cases} 1 & \text{with probability } \theta, \\ 0 & \text{with probability } 1-2\theta, \\ -1 & \text{with probability } \theta, \end{cases}$$

for some $\theta \in (0, \frac{1}{2}]$. Thus, the θ parameter determines how dynamic the state of $X[t]$ is within this basic structure. In each slot t, the transmitter-driven policy uses $X[t]$ and $\hat{X}[t]$ information[5] to decide whether to send an update at the time by choosing $U[t] \in \{0, 1\}$. Every update causes a fixed cost of c units, as modeled above. Supposing that $X[0] = \hat{X}[0] = 0$, and defining the error in the estimation as $\varepsilon[t] = |X[t] - \hat{X}[t]|$ at time t, the objective of the design is to

$$\text{minimize} \quad \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[f\left(\varepsilon[t]\right) + c \cdot U[t]\right], \tag{1.20}$$

where $f(\cdot)$ is a nonnegative-valued function with $f(0) = 0$ that measures the cost associated with the error. While this function can in general be an arbitrary non-decreasing function of its input, a very common case is when $f(\varepsilon) = \varepsilon^2$, which

[5] The transmitter can know $\hat{X}[t]$ since it knows the estimation strategy of the receiver.

corresponds to *mean-squared-error* (MSE) measure of the estimation. As such, the objective (1.20) aims to optimize the tradeoff between the estimation accuracy and the communication cost of updates. Investigation in (Yun et al. 2018) proves that the optimal transmitter-driven policy that solves this problem is *threshold-type* whereby the update is made whenever $\varepsilon[t] \geq \gamma$ for some γ that depends on the source dynamics θ, the estimation error measure $f(\cdot)$ and the update cost c. In particular, when $f(\varepsilon) = \varepsilon^2$, the optimal threshold is given by $\approx \sqrt[4]{12c\theta}$. If the parameters θ and c are unknown to the controller, the paper also develops sequential learning strategies to find the optimal threshold level adaptively. Fig. 1.20 illustrates the average cost (1.20) with MSE measure of a

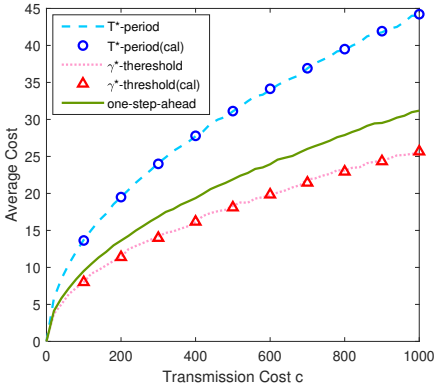


Figure 1.20 Average cost comparison with $\theta = 1/2$ of the three policies. Both the optimal transmitter-driven γ^*-threshold policy and the suboptimal transmitter driven one-step-ahead policy performs increasingly better than the optimal receiver-driven T^*-threshold policy as c increases.

simple random walk, i.e., $\theta = 1/2$, for the *optimal receiver-driven policy*, which performs updates periodically with the optimal choice of period T^*, and the *optimal transmitter-driven policy* with threshold-level γ^* selected as described above. This reveals that as the communication cost c increases, the improvement from transmission-driven updates grows since it avoids unnecessary transmissions. The figure also includes another heuristic transmitter-driven policy, called *one-step-ahead* policy, which shows that even partial use of source state information can substantially improve over the optimal receiver-driven strategy. ◇

This basic example, and many other works that we will partially overview below, reveals several interesting new elements in maintaining fresh information at the receiver. In particular, we see that the need for an update is determined by the *level of error* in the estimation, and *not the age or the delay of the last update*. The error level, in turn, depends on the dynamics of the source (i.e., how predictable it is) as well as the cost of communication needed for the update.

In the next section, we will see that these effects raise further new consid-

erations when we expand the setting to a multi-source environment, as needed in future networks. We note that, in the single-source setting, there have been numerous other works (e.g., (Chakravorty & Mahajan 2015, Gao, Akyol & Başar 2015, Sun, Polyanskiy & Uysal-Biyikoglu 2017, Yun et al. 2018, Ornee & Sun 2019, Chakravorty & Mahajan 2019, Arafa, Banawan, Seddik & Poor 2020) with: a range of source dynamics (such as Ornstein-Uhlenbeck processes, Wiener processes, Gauss-Markov processes, first-order auto-regressive models, etc.); varying communication channel models (such as random service time, noisy channels subject to erasures or errors, cost-based models); and various objectives (such as MSE minimization under update frequency bound, age-based nonlinear cost minimization under sampling frequency constraints, etc.). While the specific results differ in these work due to differences in the dynamics, costs, and constraints, the threshold-based design that is achieved in Example 1 remains a frequent characteristic of optimal designs for remote estimation.

1.4.2 Multi-Source Remote Estimation

In this section, we move from the single-source to a distributed multi-source remote estimation setting, where the dynamically evolving states $\mathbf{X}[t] = (X_i[t])_{i=1}^n$ of n sources are accessible by k distributed transmitters that must share a wireless communication medium to update the common receiver, which estimates the state with $\hat{\mathbf{X}}[t] = (\hat{X}_i[t])_{i=1}^n$ as depicted in Fig. 1.21.

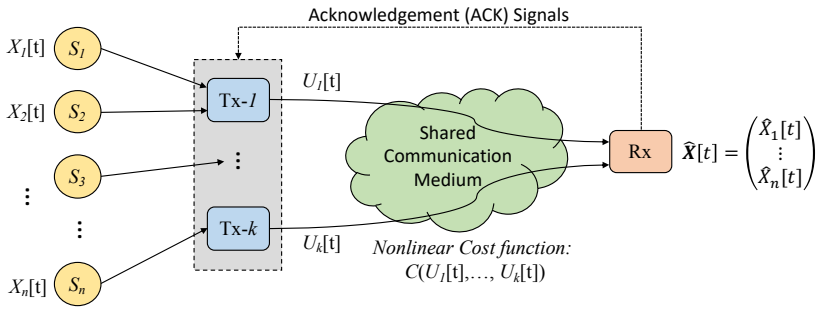


Figure 1.21 Multi-source remote estimation setting.

Similar to the single-source setting, each transmitter k must make its update decision by selecting $U_k[t] \in \{0, 1\}$. Also, as before, the update policy can be transmitter-driven (with access to the true states of the connected sources) or receiver-driven (without access to the true source states). However, in this distributed setting, a new challenge emerges that concerns the cost characteristics of the multi-user communication medium. In what follows, we discuss the diverse nature that this communication cost can take under different circumstances.

• *Emergence of Nonlinear Cost Structures of the Multi-User Communication:* Recall that in the single-source setting, we assumed that the cost of each update

communication was fixed to be $c > 0$. However, the cost of communication in the multi-user case can differ drastically depending on the medium access strategy, the coding policy, the interference effects, and protocol overhead, just to name a few. It is easy to see that, if $k = n$ and the cost function becomes a linear function of $\sum_{i=1}^{k} U_k[t]$, then the multi-user problem can be decomposed to k independent single-user problems. However, such a cost structure presumes completely orthogonal channels between each transmitter and the receiver, which requires increased coordination between transmitters. More often, the cost function will take a nonlinear structure, in which case interesting multi-user remote estimation problems emerge. In the rest of this section, we discuss the cases of distributed and co-located sources separately since they tend to exhibit *super-linear* versus *sub-linear* cost functions, respective, which raise different research questions and solutions.

Remote Estimation for Distributed Sources:
In this setting, we consider the case of physically distributed sources so that the number of transmitters k in the generic setting is large. As the number k of transmitters in the wireless network increase, the multi-user uplink communication gets increasingly congested and distributed. Therefore, as the total number $\sum_{i=1}^{k} U_i[t]$ of updates increase in a slot, the communication will be subject to super-linear costs due to increased interference effects and coordination needs. A particular instance of such super-linear growth is illustrated in Fig. 1.22 whereby we plot the energy consumption of k distributed transmitters making simultaneous updates over a shared random access channel. We see that for varying values of the attempt probabilities q of each transmitter, the cost function grows at various super-linear growth rates. This super-linear cost structure raises an

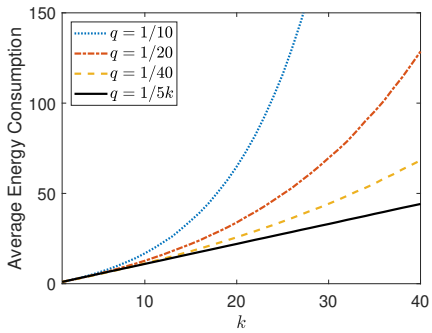
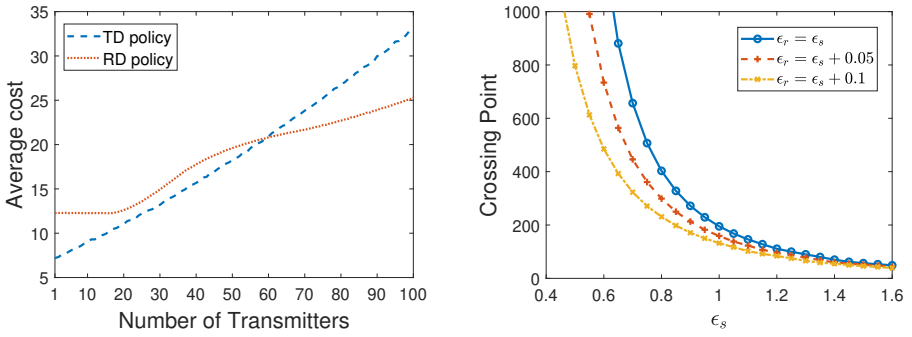


Figure 1.22 Average energy consumed for successful update with k simultaneous transmitters, when each transmitter attempts with probability q.

interesting tradeoff between transmitter-driven (TD) and receiver-driven (RD) update paradigms, which did not exist in the single-source setting, as discussed in the next example.

Example 2 (TD vs. RD Updates for Distributed Sources (Kang, Eryilmaz & Joo 2021)) Example 1 showed that the optimal transmitter-driven update policy always outperforms the optimal receiver-driven policy for a single source. This is no longer the case for distributed sources. In particular, the randomized timing of the threshold-based transmitter-driven design, such as the γ^*-threshold policy of Example 1, results in a random number of updates at a given slot. Since the communication cost function takes a super-linear form in this distributed setting, the mean communication cost (due to Jensen's Inequality) will be boosted. In contrast, a receiver-driven policy can schedule the update timings at the outset so that a deterministic number of transmissions are made at a given slot, thereby keeping the mean communication cost steady. This communication cost gain of receiver-driven design can compensate for some of its disadvantages from its knowledge of the source state information.



(a) Average cost when $\epsilon_s = \epsilon_r = 2$.

(b) Crossing point with varying ϵ_s and ϵ_r.

Figure 1.23 Performance comparison between TD and RD policies.

In Fig. 1.23, we illustrate these tradeoffs by assuming that transmitter-driven communication cost function take the form $C(\mathbf{U}[t]) = c(\sum_i U_i[t])^{1+\epsilon_s}$ and the receiver-driven communication cost function take the form $C(\mathbf{U}[t]) = c(\sum_i U_i[t])^{\epsilon_r}$, with $\epsilon_s \geq \epsilon_r \geq 0$. For $c = 50$ random walk parameter $\theta = 0.3$, and $\epsilon_s = \epsilon_r = 2$, Fig. 1.23a shows the average MSE and communication cost performances of both the optimized TD and the RD policies as the number of transmitters k grow. We can see that the TD policy outperforms RD for small enough number of transmitters until a *crossing point* is reached. Beyond this crossing point number of users, the RD policy starts outperforming since its advantage of centralized scheduling dominates advantage from the individual state information that is available to TD policy. Fig. 1.23b illustrates how this crossing point changes as a function of ϵ_s and ϵ_r parameters of the TD and RD communication cost functions. ◇

Remote Estimation for Co-located Sources:
In this setting, we envision the remote estimation of many co-located sources that share common transmitter(s) for their updates. In contrast to the super-linear

cost structures when the number of transmitters k increases, the communication cost function typically takes a sub-linear form as the number of sources n increases while k remains small. In this new regime, more and more sensors are monitored by a common transmitter, which allows the transmitter to perform more efficient strategies to encode and time the updates in its transmissions, thereby reducing the cost per unit update. Theoretically, this is supported by the information-theoretic foundations that encoding more information into the transmission increases the coding rate (since the codeblock length increases). Practically, this is supported by the fact that sending multiple updates at once allow the use of less overhead in each transmission (for synchronization, probing, channel estimation, etc.). Accordingly, in this regime it is sensible to assume $C(\mathbf{U}[t])$ to be a sub-linear function of $\sum_{i=1}^{k} U_i[t]$ such as $C(\mathbf{U}[t]) = c\,(\sum_i U_i[t])^{\alpha}$ for $\alpha \in [0, 1]$. In this setting, interesting new dynamics emerge that call for the design of new policies for performing updates.

In particular, these designs need to optimize the tradeoff between two forces: on the one hand, sending updates without delay is desirable to keep the error low, but this calls for sending a small number of updates at a time, hence operating in the inefficient transmission regime; on the other hand, delaying updates is desirable so that multiple updates can be made at a time, hence operating in the efficient transmission regime, but this causes higher errors in the estimation of the source state. Good update policies must balance these two forces in order to maintain tolerable estimation error levels with low communication costs. More work is needed in this direction.

We would like to end the section by noting that the remote estimation problem for multiple sources has been attracting increasing attention in the literature, and continues to be an active area of research. Numerous interesting works have attacked the question of optimal update strategies for multi-source settings (e.g., (Nar & Başar 2014),(Li, Quevedo, Lau & Shi 2014),(Leong, Dey & Quevedo 2016),(Han, Wu, Zhang & Shi 2017),(Ding, Li, Dey & Shi 2017),(Vasconcelos, Nayyar & Mitra 2017), (Gatsis, Ribeiro & Pappas 2018),(Wu, Ding, Cheng & Shi 2020)) under various source models, objectives, and/or the dynamics.

1.5 Conclusions and Future Works

In this chapter, we focused on the age-efficient scheduling design that is essential for information monitoring, tracking, and control in communication networks. In particular, we first considered the age-efficient scheduling algorithm design in the presence of elastic traffic, inelastic traffic, and heterogeneous traffic, respectively. Then, we discussed the fresh scheduling design for remote estimation. All these discussions cover some of the most recent advances in the age-efficient algorithm design in communication networks, and motivate several interesting research efforts. Some of the possible future research directions include the following:

- In the presence of elastic traffic, age-efficient algorithm design is as hard

as its delay-efficient counterpart. While there are delay-optimal algorithms in some special regimes (e.g., heavy-traffic regime or large-system regime), it is interesting to study age-optimal algorithms in these special regimes.

• With regard to the inelastic traffic, the existing age-efficient algorithm achieves a tradeoff between the AoI and the algorithm convergence speed, which may not be an optimal tradeoff. An interesting question is whether there are any algorithms that can achieve a provably better or even optimal tradeoff.

• For the heterogeneous traffic, we discussed the impacts of various scheduling design on the AoI performance via simulations. Can we analytically quantify the AoI performance in such a case? In addition, how do we develop an analytical framework for age-efficient algorithm design for heterogeneous traffic?

• For distributed remote estimation, there are numerous interesting research directions to pursue whereby the source dynamics, the communication channel imperfections and limitations, and the objectives can differ to accommodate various network and user characteristics in future IoT and cyber-physical systems.

Notes

References

Arafa, A., Banawan, K., Seddik, K. G. & Poor, H. V. (2020), 'Timely estimation using coded quantized samples', *arXiv preprint arXiv:2004.12982* .

Bedewy, A. M., Sun, Y. & Shroff, N. B. (2016), Optimizing data freshness, throughput, and delay in multi-server information-update systems, *in* '2016 IEEE International Symposium on Information Theory (ISIT)', IEEE, pp. 2569–2573.

Chakravorty, J. & Mahajan, A. (2015), Distortion-transmission trade-off in real-time transmission of gauss-markov sources, *in* 'IEEE International Symposium on Information Theory (ISIT)', pp. 1387–1391.

Chakravorty, J. & Mahajan, A. (2019), 'Remote estimation over a packet-drop channel with markovian state', *IEEE Transactions on Automatic Control* **65**(5), 2016–2031.

Costa, M., Codreanu, M. & Ephremides, A. (2016), 'On the age of information in status update systems with packet management', *IEEE Transactions on Information Theory* **62**(4), 1897–1910.

Ding, K., Li, Y., Dey, S. & Shi, L. (2017), 'Multi-sensor transmission management for remote state estimation under coordination', *IFAC-PapersOnLine* **50**(1), 3829–3834.

Gao, X., Akyol, E. & Başar, T. (2015), Optimal estimation with limited measurements and noisy communication, *in* '2015 54th IEEE Conference on Decision and Control (CDC)', pp. 1775–1780.

Gatsis, K., Ribeiro, A. & Pappas, G. J. (2018), 'Random access design for wireless control systems', *Automatica* **91**, 1–9.

Han, D., Wu, J., Zhang, H. & Shi, L. (2017), 'Optimal sensor scheduling for multiple linear dynamical systems', *Automatica* **75**, 260–270.

Harchol-Balter, M. (2013), *Performance modeling and design of computer systems: queueing theory in action*, Cambridge University Press.

Joo, C. & Eryilmaz, A. (2018), 'Wireless scheduling for information freshness and synchrony: Drift-based design and heavy-traffic analysis', *IEEE/ACM transactions on networking* **26**(6), 2556–2568.

Kadota, I., Uysal-Biyikoglu, E., Singh, R. & Modiano, E. (2016), Minimizing the age of information in broadcast wireless networks, *in* '2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)', IEEE, pp. 844–851.

Kam, C., Kompella, S., Nguyen, G. D. & Ephremides, A. (2015), 'Effect of message transmission path diversity on status age', *IEEE Transactions on Information Theory* **62**(3), 1360–1374.

Kang, S., Eryilmaz, A. & Joo, C. (2021), Comparison of decentralized and centralized update paradigms for remote tracking of distributed dynamic sources, *in* 'Proceedings of Infocom'.

Kaul, S. K., Yates, R. D. & Gruteser, M. (2012*a*), Status updates through queues, *in* '2012 46th Annual Conference on Information Sciences and Systems (CISS)', IEEE, pp. 1–6.

Kaul, S., Yates, R. & Gruteser, M. (2012*b*), Real-time status: How often should one update?, *in* '2012 Proceedings IEEE INFOCOM', IEEE, pp. 2731–2735.

Leong, A. S., Dey, S. & Quevedo, D. E. (2016), 'Sensor scheduling in variance based event triggered estimation with packet drops', *IEEE Transactions on Automatic Control* **62**(4), 1880–1895.

Li, B., Li, R. & Eryilmaz, A. (2014), 'Throughput-optimal scheduling design with regular service guarantees in wireless networks', *IEEE/ACM Transactions on Networking* **23**(5), 1542–1552.

Li, Y., Quevedo, D. E., Lau, V. & Shi, L. (2014), Multi-sensor transmission power scheduling for remote state estimation under sinr model, *in* '53rd IEEE Conference on Decision and Control', pp. 1055–1060.

Liu, Z., Huang, L., Li, B. & Ji, B. (2020), Anti-aging scheduling in single-server queues: A systematic and comparative study, *in* 'IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)', pp. 309–316.

Liu, Z., Huang, L., Li, B. & Ji, B. (2021, accepted), 'Anti-aging scheduling in single-server queues: A systematic and comparative study', *Journal of Communications and Networks* .

Lu, N., Ji, B. & Li, B. (2018), Age-based scheduling: Improving data freshness for wireless real-time traffic, *in* 'Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing', pp. 191–200.

McKeown, N., Mekkittikul, A., Anantharam, V. & Walrand, J. (1999), 'Achieving 100% throughput in an input-queued switch', *IEEE Transactions on Communications* **47**(8), 1260–1267.

Nar, K. & Başar, T. (2014), Sampling multidimensional wiener processes, *in* '53rd IEEE Conference on Decision and Control', pp. 3426–3431.

Newman, M. E. (2005), 'Power laws, pareto distributions and zipf's law', *Contemporary physics* **46**(5), 323–351.

Ornee, T. Z. & Sun, Y. (2019), 'Sampling for remote estimation through queues: Age of information and beyond', *arXiv preprint arXiv:1902.03552* .

Srikant, R. & Ying, L. (2013), *Communication networks: an optimization, control, and stochastic networks perspective*, Cambridge University Press.

Sun, Y., Kadota, I., Talak, R. & Modiano, E. (2019), 'Age of information: A new metric for information freshness', *Synthesis Lectures on Communication Networks* **12**(2), 1–224.

Sun, Y., Polyanskiy, Y. & Uysal-Biyikoglu, E. (2017), Remote estimation of the wiener process over a channel with random delay, *in* 'IEEE International Symposium on Information Theory (ISIT)', pp. 321–325.

Talak, R., Karaman, S. & Modiano, E. (2019), 'Optimizing information freshness in wireless networks under general interference constraints', *IEEE/ACM Transactions on Networking* **28**(1), 15–28.

Tassiulas, L. & Ephremides, A. (1990), Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks, *in* '29th IEEE Conference on Decision and Control', IEEE, pp. 2130–2132.

Tassiulas, L. & Ephremides, A. (1993), 'Dynamic server allocation to parallel queues with randomly varying connectivity', *IEEE Transactions on Information Theory* **39**(2), 466–478.

Vasconcelos, M. M., Nayyar, A. & Mitra, U. (2017), Optimal sensor scheduling strategies in networked estimation, *in* '2017 IEEE 56th Annual Conference on Decision and Control (CDC)', pp. 5378–5384.

Wu, S., Ding, K., Cheng, P. & Shi, L. (2020), 'Optimal scheduling of multiple sensors over lossy and bandwidth limited channels', *IEEE Transactions on Control of Network Systems* .

Yun, J., Joo, C. & Eryilmaz, A. (2018), Optimal real-time monitoring of an information source under communication costs, *in* 'IEEE Conference on Decision and Control (CDC)', pp. 4767–4772.