Optimizing the Age of Information with Segmentation and Predictive Scheduling

Jin Zhang, Peng Zou, Suresh Subramaniam
ECE Department, George Washington University, Washington DC, 20052, USA
zhangjin, pzou94, suresh@gwu.edu

***Abstract*—As the metric of Age of Information (AoI) has gained its popularity in recent years, a collection of scheduling policies has been proposed in order to optimize AoI. It is natural to raise the question of how much these scheduling policies could be improved and how many predictive packets are necessary to achieve optimum. The performance of scheduling policies can be quantitatively evaluated by comparing them with the optimal AoI. Given a packet sequence, there must be at least one packet combination to achieve optimal AoI, which can be obtained by exhausting all possible decisions of preserving/rejecting packets. However, it is not effortless to attain prior knowledge of each update's arrival and service time. Moreover, this simple scheduling policy of traversing every combination of 2^n consumes unaffordable resources of time and energy and computing power.**

Through mathematical analysis, we found that, given a sufficiently long packet sequence, it could be segmented into local epochs with invariant global optimal policy, and local optimization of each epoch incrementally aggregates the global optimal AoI of the entire sequence. From this new perspective, this also explains the counter-intuitive phenomenon [1] [2] [3] of idle waiting time instead of transmitting updates immediately. By taking advantage of the segmentation proposed in this paper, we only utilize part of inter-arrival times and service times of a finite number of future updates to acquire the global optimal AoI for a full sequence. Numerical results also show that under exponential distribution, the segmentation algorithm with partial prediction needs only to predict 2-10 subsequent updates in future and accurately achieve the optimal AoI. After comparing with the AoIs obtained from other scheduling policies, optimization performance of predictive scheduling policy prevails over others.

I. Introduction

To explore optimal policies that can guarantee timeliness of wireless network, researchers recently proposed a novel metric AoI (age of information) [1] and its derivations peak age [2] [5], overage probability [4] [9], Max-AoI [8] [14] and value of information [16]. Given the probability distributions of service time and arrival time of packets, the threshold policies [7] [12] [13] [15] achieved their average optimal AoIs in statistics. In addition, some researchers have also proposed deterministic policies with fixed-probability [6] [7] and obtained an optimal scheduling probability to determine whether to abandon or preserve being-served packets. These two categories of policies for improving wireless networking performance can make determinations with distribution knowledge to ensure that AoI reaches a certain degree of optimization. Nevertheless, these policies are featured with average-based AoI optimization in statistics rather than accurate optimum.

A. Motivation and Problem Statement

At present, there has been no strict theoretical basis regarding evaluating which of these policies have better performance since the disadvantage of the optimization policies in statistics is concerned with introducing predictive policy. For instance, the threshold-based optimization policy derives fixed-threshold from existing information of packet sequence [13] and explores the optimal peak AoI [2]. Once the service time exceeds the fixed threshold, due to particular packets with lengthened service times, the scheduling policy takes actions (such as an deterministic policy) to preempt those timeout packets and ensure that the information age does not evolve excessively and even diverge as time increases. It is supposed to be continuing and completing this transmission session with a bit of timeout for benefit of long-term AoI deduction, when a fixed threshold is triggered with a small amount of timeout. However, traditional preemptive scheduling policies [7] [13] [15] would abandon this timeout packet with threshold constraint violation that should have been delivered to server within a short period of time. As a result, the corresponding AoI value that could have been smaller becomes greater instead. Unfortunately, it cannot achieve the perfect effect of optimizing transmission in this case. In addition, the predefined configuration of fixed threshold requires prior knowledge of the entire packets' distribution, which is not precisely obtained in advance. Moreover, the threshold based on overall distribution can lead to a slower response speed for immediate optimization with respect to local-based policy. As in alternative words, the threshold-based and fixed-probability scheduling policies have flaws. Although these threshold-based approaches on wireless network and edge computing may obtain average optimal in statistics and near-optimal in theoretic for a given sequence of packets, they cannot achieve the optimal AoI.

There are difficulties in finding optimal AoI for a long sequence in mobile communication with resource constraints. The total number of packets in transmission tasks could be significant (such as millions and more). Furthermore, the exponential growth of possible combinations of 2^n makes a simple exhaustive scheduling policy impossible to implement in a wireless circumstance.

B. Solution and Contribution

A theorem in this article reveals that one or more idle states (called separator) exist between the optimal packet distribu-

tions in the FIFO queue system with infinite buffer. It divides a single long sequence into multiple short sequences (defined as epochs) and superimposes them to reconstruct a full optimal AoI process. The intuition originates from the observation that the total transmission time of overall packets tends to be much longer than single inter-arrival and service time. Some works have made progress on inter-arrival time prediction [10] and also contribute to our long sequence segmentation approach. Specifically, our contributions are outlined as follows.

- We propose a partial predictive approach of segmenting an arbitrary packet sequence following an exponential distribution. By applying queuing system model, we formulate general mathematical expressions for calculating AoI for any scheduling policy in a system with infinite buffer. Theorems provide that our algorithm divides long sequence into short epochs, by proving the properties of long sequence segmentation and separator periodicity.
- Only finite number of packets are necessary to obtain optimum with respect to global optimal AoI. Predictive scheduling policy with local packets achieves global optimal AoI with partial prediction. More future packets are predicted and higher optimization accuracy it reaches.
- This article analyzes fault tolerance for packet prediction error and evaluates the impact of estimation errors on achieving optimal AoI. Experiments shows that limited prediction error does not affect global optimum as well as error range results in parts of incorrect decision making and deviation from the optimal AoI to some extent.
- Simulation demonstrates quantitative comparison results with partial or full prediction-based and fixed probability-based average age. The probability of separator presence reaches up to 99% and even higher within ten packets.

The remainder of this article is organized as follows. The system model is introduced in Section 2. In Section 3, optimization problem is formulated. In Section 4, by theoretical analysis, we present the segmentation mechanism. In section 5, we provide numerical results to substantiate the findings.

II. System Model

A point-to-point communication system is considered with a single sender-receiver pair transmitting status updates from a source to a destination. A predictor estimates inter-arrival time [10] and service time of the future updates to determine whether to discard or preserve the update under being serviced, as shown in Fig. 1. The model has an infinite buffer with first-come first-served (FCFS) that allows the system to store all updates generated by a source. It also assumes that future updates' inter-arrival time and service time can be predicted either fully or partially.

The packets arrive at the predictor as a Poisson process with arrival rate λ at instants t_i. A packet may be discarded in the queuing phase by scheduler, as well as those that remain from removal are transmitted to the server and are received by the receiver after the service time X_i and the delay time τ_i at $t_i' = t_i + \tau_i + X_i$, which is defined as the delivery time t_i' of the i^{th} packet. Here, the inter-arrival time I_i and service time X_i of

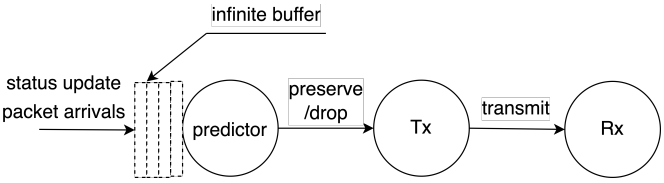

Fig. 1. System model with status update packets arriving at a single server with queues (infinite buffer), through predictor and scheduler determining to either preserve or drop updates and minimizing the age of information.

packets are instantiated into concrete values in accordance with independent and identical exponential distribution. In addition, we use I_i to denote inter-arrival time [1] and define it as $I_i = t_i - t_{i-1}$, where $t_0 = 0$ and $i \in \{1, 2, 3, ..., N\}$.

III. Optimization Problem Formulation

In the model with full packet prediction, as shown in Fig 2, a simple scheduling policy exhausts every possible combination and compares all average AoI values from each other. Thus, a simple exhaustive policy can achieve the optimal AoI process. However, due to huge search space and resource consumption of exhaustive scheduling policy, we propose segmentation approach instead of exhausting all possible combinations.

A. Symbol declarations and variable definitions

The initial age $A_0^{(i)}$ of an AoI process represents the initial AoI value at the very beginning instant of a process, with which the next new process begins. The final age $A_{fin}^{(i)}$ of an AoI process denotes the remains of the age of information at the exact end instant of a packet sub-sequence age evolution, as shown in Fig.2(b), and will be regarded as the initial age of next AoI process of the pending consecutive packet sequence. The age of information between $A_0^{(i)}$ and $A_{fin}^{(i)}$ evolves as

$$A_i(t) = t - t_i$$

An idle waiting time among long sequence is defined as an explicit or implicit separator and emerges periodically. The original distribution generates an idle state called $explicit\ separator$ when the previous update is successfully transmitted, but the next update is not generated immediately. There is no idle state between originally distributed updates, but an idle state exists after optimization. In this case, we regard an optimized idle state as $implicit\ separator$, as shown in Fig.3(a). The three elements (separator, initial and final age, AoI) and scheduling policy divide one global AoI evolution process into multiple sub-sequences, which are defined as independent $epochs$. For more details, the main notations in this article is shown in the table I.

B. Problem formulation

Given the initial age of information, the delivery of each packet dynamically determines an AoI evolution process. The

TABLE I
MAIN NOTATION TABLE

I_i	inter-arrival time of the i^{th} packet
X_i	service time of the i^{th} packet
$A(t)$	age of information at the time instant t
$A_0^{(i)}$	initial age of the i^{th} epoch
$A_{Fin}^{(i)}$	final age of the i^{th} epoch under any policy π
t_i	generation/arrival time of the i^{th} packet
u_i	the i^{th} packet
d_i^{π}	delivery time for any policy π
τ_i	delay time of the i^{th} packet
τ_q^*	equivalent delay is service time minus inter-arrival
π	any policy of full packet sequence
π^*	optimal policy of full packet sequence
$\pi_{ep_n}^*$	local optimal policy of the n^{th} epoch
b_n	decision of preserving or rejecting the n^{th} packet
$\hat{b}_n$	decision of preserving/rejecting estimated n^{th} packet
Φ_i	cumulative AoI for a single packet (e.g., i^{th} packet)
Q_n^{π}	local cumulative AoI for the n^{th} epoch
$Q_i^{\pi_i^*}$	local optimal cumulative AoI of i^{th} epoch with $A_0^{(i)}$
Q^{π^*}	global optimal cumulative AoI for optimal policy π^*
$l_m^{(1)}$	index number of preserved packets with total of m
$l_m^{(0)}$	index number of rejected packets with total of m
$\overrightarrow{\mathbf{l}^{(1)}}$	ordinal vector $\overrightarrow{\mathbf{l}^{(1)}} = \{l_1^{(1)}, l_2^{(1)}, ..., l_m^{(1)}\}$
$\delta_I^{(k,l)}$	estimated error of inter-arrival time for $k+l$
$\varepsilon_X^{(k,l)}$	estimated error of service time for $k+l$ packets
M_i^{π}	maximum delay of i^{th} packet under policy π
a_{max}^*	requirement of maximum estimated deviation
ΔA_{ave}^*	average deviation between estimated and optimal

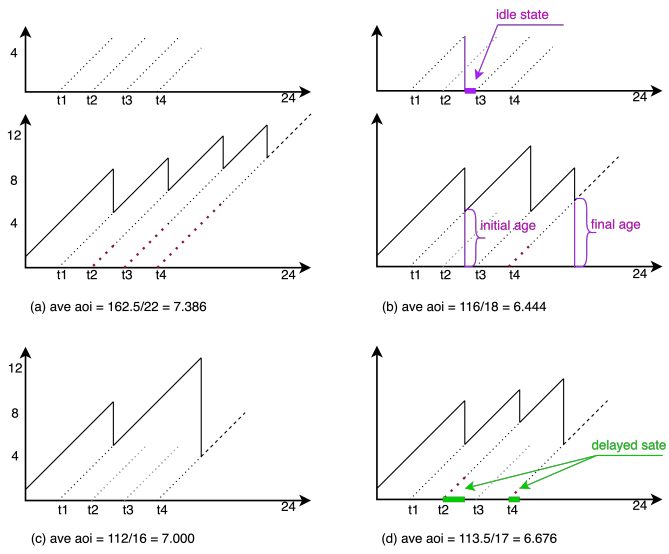

Fig. 2. The variant combinations of updates generate various AoI processes, even with exact same updates' arrival and service time. (a) shows the original arrival and service time. The AoI evolution of discarding the second update is illustrated in (b), rejecting second and third in (c), abandoning third in (d).

packet delivery timing equation under an arbitrary policy π for each update is derived as

$$d_i^{\pi} = t_i' - t_i = \tau_i^{\pi} + X_i = \sum_{q=i-k}^{i-1} (\tau_q^*)^{\pi} + X_i \tag{1}$$

where the τ_i^{π} denotes the delay time of each packet and τ_q^{π} represents multiple overlapping $equivalent\ delay$ times of k updates and contributes to current update's delay time, under scheduling policy π, since the nearest idle state at the $(i-k)^{th}$ update, where $i \geq 1$, $k \geq 0$ and $i - k \geq 1$. The equivalent delay is not the actual delay but is used to calculate the actual delay of the current packet. The equivalent delay $\tau_q^* = X_q - I_q$ equals the service time minus the inter-arrival. For instance, the dotted bold line in the Fig.2(a) represents delay τ_i, the second packet u_2 with equivalent delay value of 2 is actually delayed by 2 time units, the third packet u_3 with equivalent delay value of 2 is actually delayed by 4 time units, but the fourth u_4 with equivalent delay value of 2 is actually delayed by 6 time units. It shows the accumulation process of every delay time as time increases and packets arrive.

The AoI process evolves as each packet is generated between inter-arrival time I_n and transmission is delivered at the instant $t_i' = t_i + d_i(t)$ shown in the Fig.3(b). The general evolution equation [1] [11] and the corresponding local cumulative AoI Φ_n for each packet is defined as

$$\begin{aligned}\Phi_n &= \frac{1}{2}(I_n + d_n)^2 - \frac{1}{2}(d_n)^2 \\ &= \frac{1}{2}\left(I_n^2 + d_n^2 + 2I_n d_n\right) - \frac{1}{2}{d_n}^2 = I_n d_n + \frac{1}{2}I_n^2\end{aligned} \tag{2}$$

Every element in the ordinal vector $\overrightarrow{\mathbf{l}^{(1)}} \in \{l_1^{(1)}, l_2^{(1)}, ..., l_m^{(1)}\}$ sequentially denotes each index number of successful transmitted updates. The number of delivered updates m and the total number of updates k generated by source necessarily satisfies the relationship $m \leq k$ in an epoch. By rewriting a decimal variable in binary form, for any scheduling policy $\pi \leftarrow \{b_1 b_2 ... b_k\}$, we define an auxiliary intermediate variable b_x as each decision for every update when the '1' stands for transmission permission and '0' for packet rejection. The value b_x for each decision is defined as

$$b_x = \begin{cases} 0 & x \notin \overrightarrow{\mathbf{l}^{(1)}} \\ 1 & x \in \overrightarrow{\mathbf{l}^{(1)}} \end{cases}$$

where $\overrightarrow{\mathbf{l}^{(1)}} \in \{l_1^{(1)}, l_2^{(1)}, ..., l_m^{(1)}\}$, $x \in \{1, 2, ..., k\}$ and $m \leq k$.

To index the i^{th} packet at t_i, the variable i is introduced. Correspondingly, in order to index the n^{th} epoch, a variable n is introduced so that the packet of the n^{th} epoch can be indexed, and the first packet at t_{i_n} in the n^{th} epoch can be indexed through i_n.

Employing any scheduling policy, we have the age of information $\Phi_n^{\pi}(k)$ for k^{th} update indexed from $(i_n + k - 1)^{th}$ to $(i_n + k)^{th}$ in Fig. 3(b), where k begins with the update delayed by current update under being served. An age sample path in the Fig.3 shows updates in an epoch $(i_n, i_n + 1, ..., i_n + k)$

from a source arrive at times $t_{i_n}, t_{i_n+1}, ..., t_{i_n+k}$ and are received at times $t'_{i_n}, t'_{i_n+1}, ..., t'_{i_n+k}$. But the update at (i_n+1) is discarded under a policy π.

Therefore, the policy skips the update at (i_n+1) as well as ordinal vector is initialized as $l_1^{(1)} = i_n$ and $l_2^{(1)} = i_n + 2$, ..., $l_m^{(1)} = i_n + k$. For the n^{th} delivered update, I_n and d_n are the inter-arrival, delivery times, and Φ_n, Q_n is the age and cumulative age for one update and an epoch, respectively.

By substituting multiple inter-arrival $I_n = \sum_{s=l_x^{(1)}}^{l_{x+1}^{(1)}} I_{i_n+s}$ and latest delivery time $d_n = d^{\pi}_{i_n+l_{x+1}^{(1)}}$ into the equation (2), when discarding multiple consecutive updates between $[l_{x+1}^{(1)} - l_x^{(1)}]$ in arbitrary policy, we have the generic equation for two preservation actions at $l_x^{(1)}$ and $l_{x+1}^{(1)}$ of any scheduling policy.

$$\Phi_n^{\pi}(l_x^{(1)}) = \left(\sum_{s=l_x^{(1)}}^{l_{x+1}^{(1)}} I_{i_n+s} \right) d^{\pi}_{i_n+l_{x+1}^{(1)}} + \left(\sum_{s=l_x^{(1)}}^{l_{x+1}^{(1)}} I_{i_n+s} \right)^2 \Big/ 2 \tag{3}$$

where $[l_{x+1}^{(1)} - l_x^{(1)}]$ represents the total number of consecutive packets rejected in any scheduling policy π. And i_n denotes index number of the first one out of two successful delivered packets within an epoch. The n in the above equation stands for the index number of the n^{th} epoch, illustrated in Fig.3(b).

By adding together age of information $\Phi_n^{\pi}(l_x^{(1)})$ for every update in an epoch, which is defined in equation (3), then it gives a general AoI equation of an epoch under policy π.

$$\begin{aligned} Q_n^{\pi}(i_n, k_n) &= Q_n^{\pi}(i_n, k_n, \vec{\mathbf{b}_n}) = \sum_{x=1}^{m} \Phi_n^{\pi}(l_x^{(1)}) \\ &= \sum_{x=1}^{m} \left[\left(\sum_{s=l_x^{(1)}}^{l_{x+1}^{(1)}} I_{i_n+s} \right) d^{\pi}_{i_n+l_{x+1}^{(1)}} + \left(\sum_{s=l_x^{(1)}}^{l_{x+1}^{(1)}} I_{i_n+s} \right)^2 \Big/ 2 \right] \end{aligned} \tag{4}$$

where the policy decision vector $\pi \leftarrow \vec{\mathbf{b}_n} = b_{n1} b_{n2} b_{n3} ... b_{nk_n}$, the continuous magnitude $\|\mathbf{b_n}\| \in \{0, 1, 2, ..., 2^{k_n} - 1\}$, $b_{nj} \in \{0, 1\}$ and $\forall j \in \{1, 2, ..., k_n\}$, when the epoch number $n \in \{1, 2, ..., p\}$ as well as $p \to \infty$. In addition, the index vector $\vec{\mathbf{l}^{(1)}} \in \{l_1^{(1)}, l_2^{(1)}, ..., l_{m_n}^{(1)}\}$ for each successfully transmitted update is defined as well as the number m_n of transmitted updates and the total number k_n of generated updates satisfies the relationship $m_n \le k_n$ in an epoch. The k_n consecutive packets are separated from i_n^{th} to $(i_n + k_n)^{th}$ update of the long sequence and are composed of a segmented epoch.

The objective function $\mathbb{E}[J^{\pi^*}]$ of minimizing average age of information in the search space $\prod$ when the total transmission time $T = \sum_{h=1}^{\infty} I_h$ approximates toward infinity is formulated

$$\mathbb{E}[J^{\pi^*}] = \mathbb{E}\left[\sum_{n=1}^{\infty} J_n^{\pi_n^*} \right] = \underset{\pi \in \prod}{\text{minimize}} \left\{ \frac{\sum_{n=1}^{\infty} Q_n^{\pi_n}(i_n, k_n, \vec{\mathbf{b}_n})}{\sum_{h=1}^{\infty} I_h} \right\} \tag{5}$$

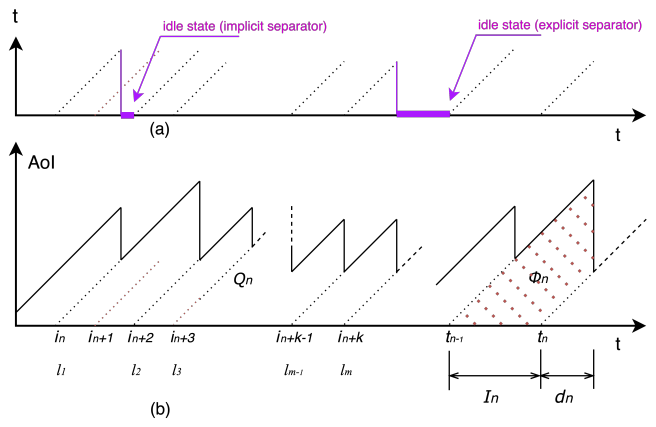

Fig. 3. (a) Original distribution drives age to evolve as time increases. (b) An age sample path: updates in an epoch $(i_n, i_n+1, ..., i_n+k)$ from a source arrive at times $t_{i_n}, t_{i_n+1}, ..., t_{i_n+k}$ and are received at times t'_{i_n}, $t'_{i_n+1}, ..., t'_{i_n+k}$. But the update at (i_n+1) is discarded under a policy π. Therefore, the policy skips the update at (i_n+1) as well as ordinal vector is initialized as $l_1^{(1)} = i_n$ and $l_2^{(1)} = i_n + 2$, ..., $l_m^{(1)} = i_n + k$. For the n^{th} delivered update, I_n and d_n are the inter-arrival, delivery times, and Φ_n, Q_n is the cumulative age and cumulative age for single update and an epoch, respectively.

We will apply sequence segmentation to the objective function (5) and find the optimal policy π^*, decision vector $\vec{\mathbf{b}}^*$.

IV. AoI Optimization with Segmentation

The equivalence of global optimal to a summation of local optimums can be mathematically expressed as follows.

$$\sum_{n=1}^{\infty} \left\{ \min_{\pi_n \in \prod_n^*} \left[Q_n^{\pi_n}\left(i_n, k_n\right) \right] \right\} = \min_{\pi \in \prod_{n=1}^*} \left[\lim_{k_1 \to \infty} Q_{n=1}^{\pi}\left(1, k_1\right) \right] \tag{6}$$

The summation of minimum local AoI $Q_n^{\pi_n}(i_n, k_n, \vec{\mathbf{b}_n})$ in the first term under local policy π_n equals to the counterpart of the global optimal achieved by the second term $Q_1^{\pi}(1, k_1, \vec{\mathbf{b}})$.

Theorem 1 *With respect to optimal AoI, the local optimal scheduling policies for a segmented epoch are exact the same ones as the global optimal for the entire long sequence, when the update sequence formulation in the epoch simultaneously satisfies the following three conditions:*

1) *minimum AoI value out of all possible policies,*
2) *minimum final age out of all possible policies,*
3) *idle state recurrently appears between optimal epochs.*

In the following, this article proves this theorem and also provides concrete examples to examine the instantiated cases for systems with infinite, one and no buffer. To begin with, we present the other two auxiliary theorems (Theorem 2 and 3) to prove segmentation Theorem 1, by showing how to reconstruct global optimal with sub-policies and why multiple explicit or implicit separators (idle state) periodically exist between optimized updates in the optimal distribution.

A. Sequence segmentation and global optimal reconstruction

Lemma 2 *With two different initial ages, the sequence with smaller initial age $A_0^{(1)}$ has less optimal AoI value $Q^{A_0^{(1)}}(\pi_1^*)$ than with greater initial age $A_0^{(2)}$ and optimal AoI value $Q^{A_0^{(2)}}(\pi_2^*)$ if they both have the same arrival time and service time of the same subsequent updates. Thus, the function of minimum AoI $Q(\pi^*) = f^{\pi^*}(A_0)$ is monotonic and entirely non-decreasing with respect to A_0.*

$$Q^{A_0^{(1)}}(\pi_1^*) \leq Q^{A_0^{(1)}}(\pi_2^*) \leq Q^{A_0^{(2)}}(\pi_2^*) \tag{7}$$

where the initial ages $A_0^{(1)} \leq A_0^{(2)}$ and π_1^*, π_2^* are respectively optimal policies with two different arbitrary initial ages.

Here, it is the proof of Theorem 2. An AoI value for any sequence is determined by four factors: initial age, arrival time, service time and scheduling policy. The first part of inequity (7) must hold $Q^{A_0^{(1)}}(\pi_1^*) \leq Q^{A_0^{(1)}}(\pi_2^*)$ because both sides have the same packet sequence and initial age but different policies π_1^* and π_2^*. According to the definition of optimal age of information, we know that $Q^{A_0^{(1)}}(\pi_1^*)$ under policy π_1^* must be the minimum among all possible policies for a given packet sequence and a specific initial age. For the second part of the inequity, both sides adopt same policy π_2^* and packet sequence but different initial age. It is evident that the sequence with less initial age has less optimal AoI value than that with greater initial age. Finally, the inequity (7) holds. Then we have the mathematical expression for long sequence segmentation.

$$\begin{aligned} Q(\pi^*) &= Q_1(\pi^*_{ep_1}) + \underline{Q[\pi^*_{(ep_2+\ldots+ep_n)}]} \\ &= Q_1(\pi^*_{ep_1}) + Q_2(\pi^*_{ep_2}) + \underline{Q[\pi^*_{(ep_3+\ldots+ep_n)}]} \\ &= \sum_{n=1}^{\infty} Q_n(\pi^*_{ep_n}) \end{aligned} \tag{8}$$

Under the sequence segmentation equation (8) with respect to optimal policies, multiple optimal local sub-policies $\pi^*_{ep_n}$ $(\forall n \in \{1, 2, ..., \infty\})$ of epochs separated from long sequence add up together and successfully reconstruct the optimal global policy π^* as well as corresponding optimal global AoI $Q(\pi^*)$.

B. Periodical presence of separator in optimal policy

Here, we will prove that separator inevitably and periodically appears in an optimal policy. This property also explains the counter-intuitive phenomenon [1] [2] [3] of idle waiting time existing between updates. Let $q_i^\pi(t)$ be the number of packets in queue i at the beginning of time t when policy π is employed. Then, it guarantees that queue i is stable if $\lim_{T\to\infty} \mathbb{E}[q_i^\pi(T)] \leq \infty$ [2]. As shown in Fig. 2(a) and 2(b), the equivalent inequality of system stability τ_n is expressed as

$$\lim_{n\to\infty} \left[\sum_{i=n-c}^{n} \left(\tau_i\right)^\pi \right] = M_n^\pi \leq \infty \tag{9}$$

where c denotes the number of overlapping packets since the nearest idle state, the finite value M_n^π represents the maximum limit of delay time for an arbitrary packet at queuing.

Lemma 3 *A sub-sequence, starting from an arbitrary update of an infinite update sequence and containing all the rest of consecutive updates, must have explicit or implicit separators between optimized updates with respect to optimal global AoI. The actual delay time $\tau_n \leq 0$ $n \in (1, 2, \ldots, \infty)$ of the n^{th} packet is negative under the optimal policy π^*.*

We exploit logical contradiction to prove the theorem. Firstly, it assumes that there are a total of k packets in the optimal distribution, and there was no idle state between the packets. That is, there was no separator in the optimal policies. Under this assumption, every next packet would be continuously postponed due to queuing in the FIFO buffer. The total amount of delay time of the k^{th} packet accumulates from the first packet to the nearest previous package $(k-1)^{th}$, because each previous package contributes a equivalent delay $\tau_q^* > 0$ to the final packet and $q \in \{1, 2, ..., k-1\}$ with mathematical expression $\tau_k = \sum_{q=1}^{k-1} \tau_q$, where every $\tau_q^* > 0$ means no idle waiting time. As the number k of packets increases, the corresponding delay time τ_k of the final packet would grow greater and greater. Its mathematical expression is as follows:

$$\tau_k = \lim_{k\to\infty} \tau_k = \lim_{k\to\infty} \sum_{q=1}^{k-1} \tau_q^* = \infty$$

According to the requirements of the network system stability in equation (9), when a packet is postponed for an infinite time, it means that the packet would no longer be transmitted to the server. Such a system is considered an unstable network system. Correspondingly, we would conclude that the network system was unstable. The optimal policy must be a policy that makes the network system stable, so separators shall appear in the optimal distribution. Thus, theorem 3 is proved.

The general mathematical expression of the delay time for a long packet sequence (e.g. given four updates defined with $\{I_1, I_2, I_3\}$ and $\{X_1, X_2, X_3, X_4\}$) under a scheduling policy $\pi = (b_1 b_2 ... b_{k_n})$ can be derived as

$$\begin{aligned} Q_n^\pi(x_n, k_n) &= Q_n^{(b_1 b_2 \ldots b_{k_n})}(x_n, k_n) \\ &= \sum_{s=1}^{m} \left[\left(\sum_{i=l_s^{(1)}}^{l_{s+1}^{(1)}} I_i \right) d^\pi_{l_{s+1}^{(1)}} + \left(\sum_{i=l_s^{(1)}}^{l_{s+1}^{(1)}} I_i \right)^2 \Big/ 2 \right] \\ &= \sum_{s=1}^{m} \left[\left(\sum_{i=l_s^{(1)}}^{l_{s+1}^{(1)}} I_i \right) \left(\sum_{q=i-k}^{i} \tau_q^\pi + X_i \right) + \left(\sum_{i=l_s^{(1)}}^{l_{s+1}^{(1)}} I_i \right)^2 \Big/ 2 \right] \end{aligned}$$

where $b_{k_n} \in \{0, 1\}$, all possible policies can be obtained by exhausting total combinations of $(b_1 b_2 ... b_{k_n})$ and k_n denotes

total number of packets in the n^{th} epoch.

$$\begin{aligned}\tau_k &= \sum_{q=1}^{k-1} \tau_q^* = \sum_{q=1}^{k-1} (X_q - I_q) \\ &= \sum_{q=1}^{k-1} (X_q - I_q) - X_{l_1^{(0)}} - X_{l_2^{(0)}} - ... - X_{l_m^{(0)}} \\ &= \sum_{q=1}^{k-1} (X_q - I_q) - \sum_{p=l_1^{(0)}}^{l_{k-m}^{(0)}} X_p \\ &= \sum_{q=1}^{k-1} (X_q - I_q) + \sum_{p=l_1^{(1)}}^{l_m^{(1)}} X_p - \sum_{s=1}^{k} X_s\end{aligned}$$

where cumulative delay time (τ_k) starts from the nearest idle separator and ends at the completion of transmitting the current update, $k \in 2, 3, ..., N$. Every update contributing to the final delay has to have overlapping times with each other. The second part of subtraction specifies those updates of m in the above equation that have been preempted in a scheduling policy.

C. Proof of Theorem 1

As given in theorem 2, the AoI evolving from minimum initial age $A_0^{(1)}$ and it always satisfies $Q_n^{A_0^{(1)}}(\pi_1^*) \leq Q_n^{A_0^{(m)}}(\pi_1^{[m]})$. Additionally, the final age of the local optimal policy $A_F^{(1)}$ is also less than other final ages $A_F^{(m)}$ under any other local policies and $Q_{n+1}^{A_F^{(1)}}(\pi_2^*) \leq Q_{n+1}^{A_F^{(m)}}(\pi_2^{[m]})$. Correspondingly, $Q_{n+k}^{A_F^{(1)}}(\pi_k^*) \leq Q_{n+k}^{A_F^{(m)}}(\pi_k^{[m]})$ holds. Finally, theorem 1 is proved.

D. Formula under $M/GI/1/1$, $M/GI/1/2$ and $M/GI/1/2^*$

For the cases of one or without buffer, we introduce two other variables $\gamma_i \in \{0, 1, 2, ..., N\}$ and $t_{r_i}^{(\gamma)}$, $r_i \in \{1, 2, ..., \gamma_i\}$ that denotes the total number and the arrival time of future updates (completely or partially) contained by the current i^{th} update, respectively, while r_i denotes the index number of contained updates under the i^{th} update. Thus, it defines the containment update vector as $\overrightarrow{\mathbf{t}_{r_i}^{(\gamma)}} = \{t_{r_1}^{(\gamma)}, t_{r_2}^{(\gamma)}, ..., t_{r_i}^{(\gamma)}\}$ for the i^{th} update and then the index number s_{i+1} of filtered updates with one or without buffer is mathematically expressed in the following iterative form.

$$s_{i+1} = s_i + \gamma_i$$

where combination vector $\overrightarrow{\mathbf{l}^{(1)}} = \{l_1^{(1)}, l_2^{(1)}, ..., l_m^{(1)}\}$, and index number vector with one or without buffer $\overrightarrow{\mathbf{s}} = \{s_1, s_2, ..., s_v\}$, $i \in \{1, 2, ..., v\}$ where v represents the total number of final updates filtered by a system with one or without buffer as well as it satisfies $\overrightarrow{\mathbf{s}} \subseteq \overrightarrow{\mathbf{l}^{(1)}}$ and $v \leq m$.

By substituting the filtered update vector $\overrightarrow{\mathbf{s}}$ into the general equation, it gives the age of information $Q_n^{\pi}(l_x^{(1)})$ for every filtered update in an epoch, which is defined in equation (3), under policy π.

$$\begin{aligned}Q_n^{\pi}(x_n, k_n) &= \sum_{s=1}^{u} Q_s^{\pi}(\gamma_s) \\ &= \sum_{s=1}^{u} \left[\left(\sum_{i=\gamma_u}^{\gamma_{u+1}} I_{y_n+i} \right) d_{x+l_{x+1}^{(1)}}^{\pi} + \left(\sum_{i=\gamma_u}^{\gamma_{u+1}} I_{y_n+i} \right)^2 \Big/ 2 \right]\end{aligned} \tag{10}$$

where the policy decision vector $\pi \leftarrow \vec{\mathbf{b}}_n = b_{n1} b_{n2} b_{n3} ... b_{nk_s}$, the continuous magnitude $\|\mathbf{b_n}\| \in \{0, 1, 2, ..., 2^{k_s} - 1\}$, $b_{nj} \in \{0, 1\}$ and $\forall j \in \{1, 2, ..., k_s\}$, when the epoch number $n \in \{1, 2, ..., q\}$ as well as $q \to \infty$. In addition, the index vector $\overrightarrow{\mathbf{l}^{(1)}} \in \{l_1^{(1)}, l_2^{(1)}, ..., l_m^{(1)}\}$ for each successfully transmitted update is defined as well as the number of transmitted updates m_n and the total number of generated updates k_n satisfies the relationship $m_n \leq k_n$ in an epoch. The k_n consecutive packets are separated from x_n^{th} to $(x_n + k_n)^{th}$ update of the long sequence and are composed of an segmented epoch.

V. Limited Prediction and Fault Tolerance

This chapter analyzes fault tolerance of prediction error in order to satisfy the timeliness requirement and average deviation ΔA_{ave}^* between estimated and optimal AoIs is expressed as $|\Delta A_{ave}^*| \leq a_{max}^*$. Based on the segmentation mechanism, we also propose *N-prediction* algorithm when only limited number of packets are predictable.

In the limited N-prediction algorithm, packets are respectively classified as four types in a limited prediction-based circumstance, such as number p of earlier packets $\{I_{n-p}^{(n-p,n-1)}, X_{(n-p)}^{(n-p,n-1)}\}$ in the past, being transmitted packet $\{I_n, X_n\}$, predictable future packets $\{I_n^{(n,k)}, X_n^{(n,k)}\}$ and unpredictable future packets $\{I_{(n+k)}^{(n+k,l)}, X_{n+k}^{(n+k,l)}\}$. The four sets are expressed as

$$\begin{aligned}I_n^{(n,k)} &= \{I_n, I_{n+1}, ..., I_{n+k-1}\} \\ X_n^{(n,k)} &= \{X_n, X_{n+1}, ..., X_{n+k-1}\} \\ I_{n+k}^{(n+k,l)} &= \{I_{n+k}, I_{n+k+1}, ..., I_{n+k+l-1}\} \\ X_{n+k}^{(n+k,l)} &= \{X_{n+k}, X_{n+k+1}, ..., X_{n+k+l-1}\}\end{aligned}$$

The estimation errors for inter-arrival $\tilde{I}_{n+k}^{(n+k,l)}$ and system times $\tilde{X}_{n+k}^{(n+k,l)}$ are respectively defined as $\delta_I^{(k,l)}$ and $\varepsilon_X^{(k,l)}$ in accordance with uniform distribution of $[-\delta_{max}, +\delta_{max}]$ and $[-\varepsilon_{max}, +\varepsilon_{max}]$ as well as thus the packet estimations are expressed as,

$$\begin{aligned}&\tilde{I}_{n+k}^{(n+k,l)} = I_{n+k}^{(n+k,l)} + \delta_I^{(k,l)} = \{\tilde{I}_{n+k}, \tilde{I}_{n+k+1}, ..., \tilde{I}_{n+k+l-1}\} \\ &\tilde{X}_{n+k}^{(n+k,l)} = X_{n+k}^{(n+k,l)} + \varepsilon_X^{(k,l)} \\ &= \{\tilde{X}_{n+k}, \tilde{X}_{n+k+1}, ..., \tilde{X}_{n+k+l-1}\}\end{aligned}$$

According to the Theorem 1, a function of finding optimal AoI and separator for local epoch is named as *segment function* and defined as

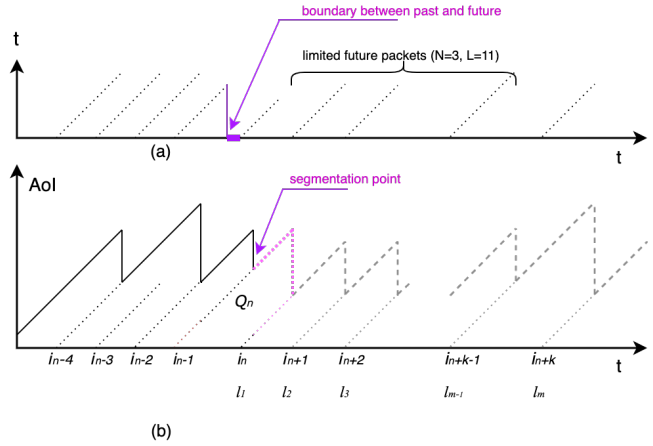

Fig. 4. Four types of packets formulate attempt epoch and predictive scheduling algorithm found near-optimal decision when K packets are predictable. The sub-figure (a) shows boundary between past and future. Segmentation origin point is illustrated in the (b) and AoI evolution is driven by packet deliveries.

$$Q_n(\pi_n^*) = \mathrm{sgmt}(I_n^{(n,k)}, X_n^{(n,k)}, \pi_n^*) \tag{11}$$

Each estimated decision $\tilde{b}_n$ of preserving or discarding current packet, the distance ΔA_n^* between perfect AoI and limited predictive policy as well as the long-term average cumulative difference ΔA_{ave}^* are expressed as

$$\begin{aligned}
\tilde{b}_n &= \underset{(k,l)}{\mathrm{argmin}}\{\mathrm{sgmt}(I_n^{(n,k)}, X_n^{(n,k)}, \tilde{I}_{n+k}^{(n+k,l)}, \tilde{X}_{n+k}^{(n+k,l)})\} \\
\Delta Q_n^* &= \left[Q_n^{\tilde{\pi}_n^*}(i_n, k_n, b_n) - Q_n^{\pi_n^*}(i_n, k_n, b_n)\right] \\
\Delta A_{ave}^* &= \lim_{N\to\infty}\left\{\sum_{n=1}^{N}\Delta Q_n^* \Big/ \sum_{n=1}^{N} I_n\right\}
\end{aligned} \tag{12}$$

The objective function in N-prediction problem is to minimize long-term total average deviation between age of information under full prediction and limited prediction-based policy.

$$\begin{aligned}
\limsup_{n\to\infty}\{\Delta A_{ave}^*\} &= \min_{\forall s\in S_j} \Delta A_s^* \\
&= \min_{\forall s\in S_j} \lim_{N\to\infty}\left\{\sum_{n=1}^{N}\Delta Q_n^* \Big/ \sum_{n=1}^{N} I_n\right\}
\end{aligned} \tag{13}$$

To find out how the error of first estimated packet affects each decision under limited N-prediction policy, we propose approximation in this case. The symbols of $\tilde{b}_n$ denotes near-optimal decision under limited N-prediction scheduling policy.

A. error deviation convergence in long-term

Theorem 4 *Segmentation for the minimum deviation between epochs converges to zero and it is mathematically expressed as*

$$\lim_{N\to\infty}\left\{\Delta A_{ave}^* \middle| \left(\tilde{\pi}_n^{(k,l)}, \pi_n^{(k,l)}\right)\right\} = 0 \tag{14}$$

where the N-prediction policy with estimation error $\tilde{\pi}_n^{(k,l)} = \{b_n, b_{n+1}, ..., \tilde{b}_{n+k}, \tilde{b}_{n+k+1}, ..., \tilde{b}_{n+k+l-1}, b_{n+k+l}, ...\}$ *and the optimal policy with accurate prediction* $\pi_n^{(k,l)} = \{b_n, b_{n+1}, ..., b_{n+k}, b_{n+k+1}, ..., b_{n+k+l-1}, b_{n+k+l}, ...\}$.

As only policies are different between Full and N Partial predictions, the original distributions and initial ages are the same and thus the long-term average deviation approximates zero as shown in the proof.

$$\begin{aligned}
&\sum_{n=1}^{N}\left[Q_n^{*(L)} - Q_n^{*(\infty)}\right] \\
&= \sum_{n=1}^{N}\left[Q_n^{\tilde{\pi}_n}(i_n, k_n, \tilde{b}_n) - Q_n^{\pi_n}(i_n, k_n, b_n)\right] \\
&= \sum_{n=s}^{s+p}\left[Q_n^{\tilde{\pi}_n}(i_n, k_n, \tilde{b}_n) - Q_n^{\pi_n}(i_n, k_n, b_n)\right] \\
&= M_2 < \infty.
\end{aligned} \tag{15}$$

$$\begin{aligned}
&\lim_{N\to\infty}\left\{\Delta A_{ave}^* \middle| \left(\tilde{\pi}_n^{(k,l)}, \pi_n^{(k,l)}\right)\right\} \\
&= \lim_{N\to\infty}\left\{\sum_{n=s}^{s+p}\left[Q_n^{\tilde{\pi}_n}(i_n, k_n, \tilde{b}_n) - Q_n^{\pi_n}(i_n, k_n, b_n)\right] \Big/ \sum_{n=1}^{N} I_n\right\} \\
&= \lim_{N\to\infty}\left\{M_2 \Big/ \sum_{n=1}^{N} I_n\right\} = 0.
\end{aligned} \tag{16}$$

The proof of theorem 4 is completed when the number N of epochs go to infinity as time increases.

B. fault tolerance for estimation errors

The maximum prediction errors of inter-arrival and service times could be possibly accepted by segmentation optimization algorithm, and it still can achieve optimal policy with fault tolerance. The error estimation σ_n^{err} for the n^{th}epoch is mathematically expressed as

$$\sigma_n^{err} = \mathrm{smgt}(\mathbb{S}_n, \tilde{b}_n) - \mathrm{smgt}(\mathbb{S}_n, b_n) \tag{17}$$

The state $\mathbb{S}_n^{(k,l)}$ for an epoch stands for epoch formulation with four types of packets and the notation is expressed as $\mathbb{S}_n^{(k,l)} = \left\{A_0^{(i)}, \left(I_n^{(n,k)}, X_n^{(n,k)}, \tilde{I}_{n+k}^{(n+k,l)}, \tilde{X}_{n+k}^{(n+k,l)}\right)\right\}$.

By comparing with the perfect optimal AoI and substituting the equation (4) into the above equation (17), we solve the following inequality and obtain the acceptable estimation error range for the assigned requirement.

$$\left|\left[Q_n^{\tilde{\pi}_n^*}(i_n, k_n, b_n) - Q_n^{\pi_n^*}(i_n, k_n, b_n)\right] \Big/ \sum_{s=i_n}^{i_n+k_n-1} I_s\right| \le a_{max}^* \tag{18}$$

As shown in the Fig.5 , when an estimated future packet does not change optimal policy, it means that the AoI evolution process would achieve optimal value. The deviation between estimated and optimal will be the same and it is expressed as $\Delta Q_n^* = 0$ in this case. If estimated future packets result in

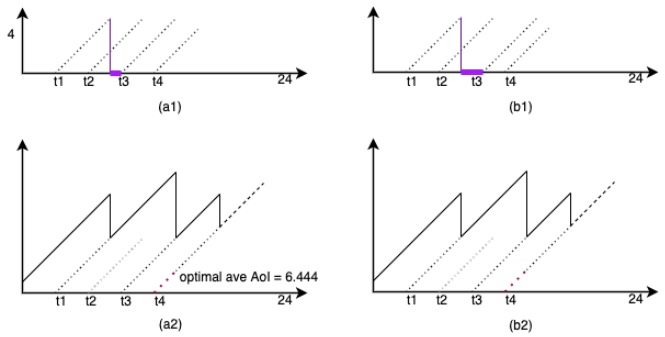

Fig. 5. (a1) original distribution and accurate predictions. (b1) estimated distribution and error predictions. (a2) optimal average AoI value (6.444) and optimal policy b = 1,3,4 under the accurate prediction of original inter-arrival and service times. (b2) optimal average AoI value (6.444) and optimal policy b = 1,3,4 under the prediction error but with the same optimal AoI evolution process.

different policies, we will use the equation (18) to evaluate its deviation and calculate the maximum estimation error δ_{max} and ε_{max} under the given requirement of a^*_{max}.

VI. Approximation on Predictive Scheduling Policy and Algorithm Design

This chapter describes how to identify and search a separator. To begin with, we introduce a few definitions. *Starting time of separator*: the delivery time of the last packet of an epoch, which is the time instant on the right side of epoch AoI evolution. If the final age at this time instant achieves the minimum value among all possible combinations. In addition, the cumulative AoI of last epoch from last starting time to the current starting time also achieves the minimum local AoI. At this instant, we call it as the *starting time* of the newly discovered *separator*. The difference between the *actual delivery time* and the *natural delivery time* is whether or not it has a delay before the it delivers to server. The natural delivery time refers to the time instant delivering to server without any delay, but the actual delivery time may or may not possibly contain a delay time due to specific scheduling policy.

A. separator searching procedure and segmentation

(1) After the starting time of a separator, algorithm traverses packets and finds a natural delivery time of the newly generated packet. From the starting time of separator to the generation time of the first packet, all the packets during this period constitute a new *first attempt epoch*, and the optimal AoI evolution process can be found through exhausting all of packets in this epoch. Then it applies the three criteria of Theorem 1 to determine the starting time of a new separator.

(2) If separator is not found at the first attempt, the algorithm appends a newly generated packet to the tail of the first attempt epoch so as to form the *second attempt epoch*. It repeats appending next new packet and performs the exhaustive optimization as described in the procedure 1, by using the criteria, until it successfully finds the starting time of a new separator.

(3) If a *first attempt epoch* would be optimized and two or more separators would appear. After checking with the criteria, the starting times of multiple separators could also be determined.

In the segmentation algorithm, the starting time of separator is appointed as the origin point of next segment, and the packet with respect to this separator is regarded as the first packet of the new epoch. Packets to the left of the origin point have a negative generation time $t_i \leq 0$, and packets to the right of the epoch have positive generation time $t_i \geq 0$. The new epoch origin will be used as a reference point for reconstructing overall optimal policy in the segmentation algorithm. Starting from the epoch origin $(A_0^{(x)}, A_0^{(y)})$ and ending at the generation time t_2 of the first packet after the completion of the first generated packet u_1 transmission X_1 (that is, $t1 \geq 0$, $t_2 \geq t_1 + X_1$), all the packets generated in this interval plus the packets of negative generation time of this epoch to formulate the first attempt epoch.

[Commentary]: There must be at least one *optimal packet* in the first attempt epoch. The first involved packet u_1 should be generated after the epoch origin and will be naturally delivered before the end of first attempt epoch. The packet u_1 can be regarded as a special instance of all possible combinations of any policies. We use logical contradiction to prove at least one optimal packet in the first attempt epoch. It assumes that after optimized no optimal packet exists within first attempt epoch. The cumulative AoI value during this period must be greater than the special policy of retaining packet u_1, because the interval period immediately follows the segment origin and at least one delivery reduces the optimized AoI value. So in this first attempt epoch, there must be at least one packet existing in the optimal policy. For the above reason, we made the rules for searching for separators in the segmentation algorithm.

B. partial prediction and approximation

If only a limited number of packets (such as two updates) can be accurately predicted. They are defined as *predictable packets*. In this setting, three cases would be discussed. First, the predictable packets and the packets in the past can possibly form a separator and it falls into the time period of the predictable updates. It determines the optimal local and optimal global decisions in this case. Secondly, when the past and predictable updates fail to form a separator, we only consider the local optimum to make decisions, or fill the unpredictable future packets with average inter-arrival and service time by averaging past packets. Under this assumption, with respect to local and global, it finally makes a decision with approximation.

Regarding the approximation method in the second case, it is necessary evaluate and quantify the positive and negative impact of decisions on the global optimal AoI. For example, the estimation policy based on estimated packets deviates from the mean square error of the ideal optimal AoI value. Thirdly, this paper considers tolerance on prediction error. How much the prediction is biased does not affect the optimal strategy.

This paragraph discusses the searching method for near-optimal decision in partial prediction. For each iteration, one estimated/unpredictable update is appended to attempt epoch. In addition, all predictable updates and the delivered updates in the past formulate the attempt epoch in partial prediction setting. By using packets in attempt epoch, predictive scheduling algorithm searches for optimal policies and separators. With the separator, it makes a preserving or rejecting decision for the currently being transmitted packet.

In the predictive scheduling algorithm design, if the previous packet relative to the current update has been transmitted, the delivery time instant of previous update is used as the starting time $A_0^{(x)}$ for calculating the AoI of of the attempt epoch with partial prediction including current update. The age $A_0^{(y)}$ at this time instant will be used as the initial age of the next packet. On the other hand, if the previous update is discarded in a policy, the generation time t_i of the current update is regarded as the starting time for calculating the AoI of the attempt epoch and correspondingly The age $A_0^{(y)}$ at this time instant as initial age. Moreover, in the case of full prediction, when the optimal policy does not exist in the final epoch, for the reason of approximation, we compare the minimum AoI under exhaustive policy with N-prediction policy, and adopt the policy with smaller AoI out of the two policies.

C. Average-optimal policy and algorithm design

In algorithm design, there are two main loops for discovering global optimal, as shown in pseudo-code algorithm 1. The outer loop searches epochs, illustrated from lines 2 to 10. The inner loop is designed to explore and identify an epoch from long sequence and achieve the optimal AoI by exhausting all possible combinations from lines 3 to 9. The separator and minimum final age checks are executed at line 4. At last, a return vector S at line 11 delivers discovered optimal epochs.

Algorithm 1 Discovering global optimal with segmentation

1: $S_n^{(j=1)} \leftarrow \{X_{n+1}, X_{n+2}, ..., X_{n+k}\}\{I_{n+1}, I_{n+2}, ..., I_{n+k}\}$
2: **while** $n \leq m$ **do** ▷ n,m for epoch index and total number
3: **for** $i \leftarrow 1, j \leftarrow 1, S_i^{(j)} \leftarrow \text{traverse}(S_n^{(j)})$ **do**
4: **if** A_{ave}^* at idle of $S_i^{(j)}$, $A_F^{(0)}$ under π_n^* **then**
5: $S_n^{(j)} \leftarrow S_i^{(j)}$, $n_n^* \leftarrow n$ ▷ epoch found
6: **else** ▷ append next packet to epoch and continue
7: $S_i^{(j+1)} \leftarrow S_i^{(j)} + \{X_{n+k+1}\}\{I_{n+k+1}\}$
8: **end if**
9: **end for** ▷ i, j for combination and attempt times
10: **end while**
11: return vector $\{n^*, S^*\} \leftarrow \{n_1^*, ..., n_m^*\}, \{S_1^*, S_2^*, ..., S_m^*\}$

For approximation algorithm design, there are two main loops for discovering global optimal, as shown in pseudo-code algorithm 2. The symbols of N and L denote the limited number of predictable packets and maximum length limit. We use x, y, z represent index number of incoming packets, epoch number and intermediate variable for numbering packets within maximum length limit L. The U_y stands for the y^{th} epoch and n_i^* is the location of the first separator in the i^{th} epoch. Moreover, $\tilde{n}$ and $\tilde{S}$ respectively denote location of the last packet and near-optimal policy because the segmentation algorithm does not find optimal within maximum length limit.

Algorithm 2 Approximating optimal AoI under predictive scheduling policy with limited number of predictable packets

1: $\{N, L\} \leftarrow \{n_0, l_0\}$ ▷ initialize N, L with maximum limit
2: $U_y \leftarrow \{u_{x+1}, u_{x+2}, ..., u_{x+N}\}$
3: **for** x in range T **do**
4: $\{n^*, S_i^*\}$=ALGORITHM1(U_y)
5: **if** separator and optimal policy $n^*, \pi^* \in S_i^*$ **then**
6: $x \leftarrow x + n^*$, $y \leftarrow y + 1$
7: $U_y \leftarrow \{u_{x+1}, u_{x+2}, ..., u_{x+N}\}$
8: **else** ▷ either search or approximate optimal AoI
9: **if** $x \leq L$ **then** ▷ within maximum length limit
10: **for** z in range L **do**
11: $\{n^*, S_i^*\}$=ALGORITHM1(U_y)
12: **if** $n^*, \pi^* \in S_i^*$ **then**
13: $x \leftarrow x + n^*$, $y \leftarrow y + 1$
14: $U_y \leftarrow \{u_{x+1}, u_{x+2}, ..., u_{x+N}\}$
15: **else** ▷ append next packet within L
16: $U_y \leftarrow U_y + \{u_{x+N+z}\}$
17: **end if**
18: **end for**
19: **else** ▷ approximation near-optimal upon $x \geq L$
20: $\{\tilde{n}, \tilde{S}_i\}$=ALGORITHM1($U_y$)
21: $x \leftarrow x + \tilde{n}$, $y \leftarrow y + 1$
22: $U_y \leftarrow \{u_{x+1}, u_{x+2}, ..., u_{x+N}\}$
23: **end if**
24: **end if** ▷ discovering separator and optimal AoI
25: **end for**
26: return $\tilde{S} \leftarrow \{S_y^*, S_{y+1}^*, ..., \tilde{S}_{y+r}, ..., \tilde{S}_{y+r+v}, ..., S_{y+m}^*\}$

The outer loop consists of every independent epochs and the process of more update involvement, illustrated from lines 3 to 25. The inner loop is designed to explore and identify an epoch from long sequence and achieve the optimum AoI value by exhausting all possible combinations located from lines 10 to 18. At last line of 26, a return vector delivers discovered epochs from long sequence and outputs approximation to optimal AoI. Moreover, when algorithm finds separator in the first attempt epoch it stops searching and returning optimal local policy π^* and S_y^* at line 6. If the algorithm does not find a separator within maximum length limit L, it approximates and returns an near-optimal policy $\tilde{\pi}$ and $\tilde{S}_y$ at line 20 in order to improve efficiency of searching and reduce resource consumption.

VII. Numerical Results

This section presents numerical results to explore the optimization performance of the prediction-based segmentation policy and validate our theoretical results.

We provide a graphical example of a long sequence transmitting 18 packets from source to destination. The local optimal epochs split by implicit and explicit separators are colored

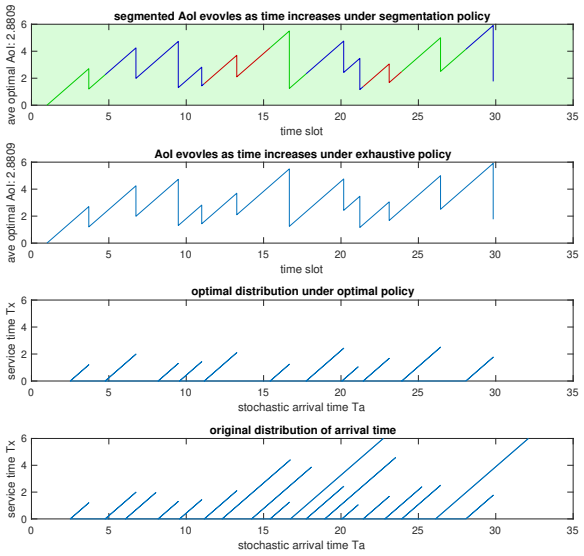

Fig. 6. *The predictive segmentation policy produces optimal AoI in sub-graph 1, while exhaustive policy produces sub-graph 2 with high intense resource consumption. Moreover, each epoch in various colors is identified in sub-graph 1 and is composed of the same global optimal policy as the one in sub-graph 2 with the same minimum AoI 2.8809. Finally, the optimal and original distribution are shown in sub-graphs 3 and 4, respectively.*

and differentiated by multiple ribbons. We use exhaustive algorithm to find optimal sub-policy for each epoch. As shown in the fourth sub-graph at the lowest part of Fig.6, the original distribution forms a sequence in which overlapping relationships between updates vary and range from independence to complete containment. For later comparison, the optimal distribution in the third sub-graph is obtained from the fourth.

Our segmentation policy generates optimal AoI shown in the sub-graph 1 of Fig. 6. By comparing AoI evolution on the sub-graphs 1 and 2, we observe that each segment corresponding to an epoch is the same as sub-graph 2. The explicit separators are found between update 1 and 2 as well as the implicit separators get identified at multiple locations (such as between update 6, 7 and 8). The Fig.6 also verifies that the minimum final age emerges at the end of each epoch, and the colored segment lines end at the exact instant when the first update of the next epoch is generated. Thus, the idle state between optimized updates in optimal distribution is at present and eliminates the delay accumulation resulting from the previous overlapped parts between original updates.

In the case of the exponential distribution, the simulated graph of explicit and implicit separators statistically appears within a number of packets arriving. A total of 10,000 independent experiments, as shown in Fig. 7 when arrival rate $\lambda = 1.0$, the cumulative probability of at least one separator presence within six updates exceeds 90 percentiles. We also observe that rare separators are discovered when packets grow greater than ten and chance decreases dramatically. Moreover, because at least two packets can compose an epoch, there is

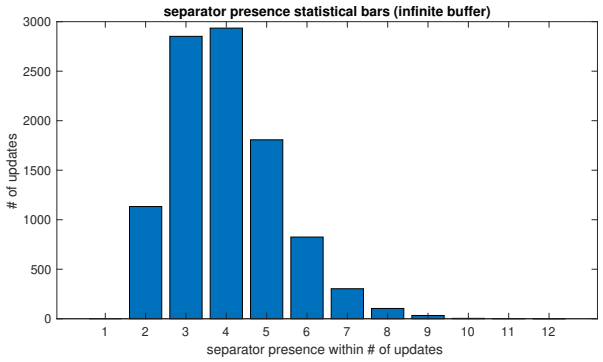

Fig. 7. *Statistical counts of separator presence for a given number N of updates (N = 12) for 10,000 independent experiments, and probability of separator presence increases in less than three updates and decreases from more updates.*

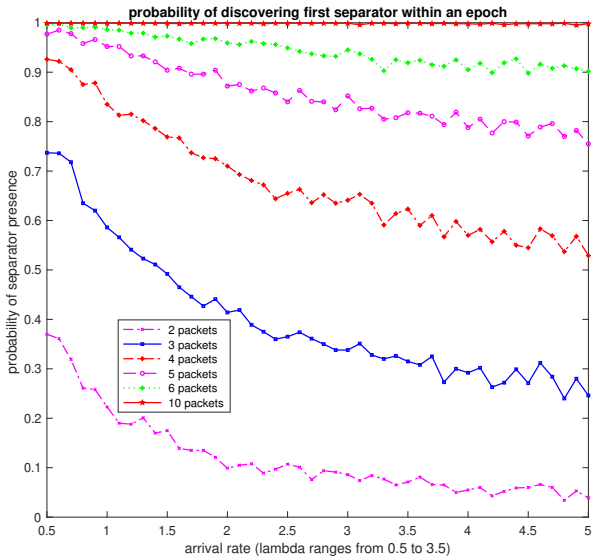

Fig. 8. *Probability distribution of separator presence within 2/3/4/5/6/10 packets when arrival rate λ varies from 0.50 to 5.00 under exponential distribution with $\mu = 1$.*

no statistical result for the one-packet case in Fig. 7.

With variation of arrival rate λ, the packet density of original distribution changes, and the probability of separator presence also varies. In Fig. 8, the curves illustrate that the probability distribution of separator presence within 2/3/4/5/6/10 packets changes as arrival rate increases from λ = 0.50 to 5.00. When the arrival rate is low, the presence probability within a given number of packets proves to be higher than the high arrival rate by comparing with the curves in Fig. 8. This also explains that each curve decreases as the arrival rate λ increases.

The curves in Fig.9 demonstrate that the performance of our prediction-based segmentation policy prevails over all of the other optimization policies as arrival rate λ increases. Thus, in comparison with other policies, segmentation policy can find theoretically optimal age of information for a long sequence.

The prediction errors of inter-arrival and service times could

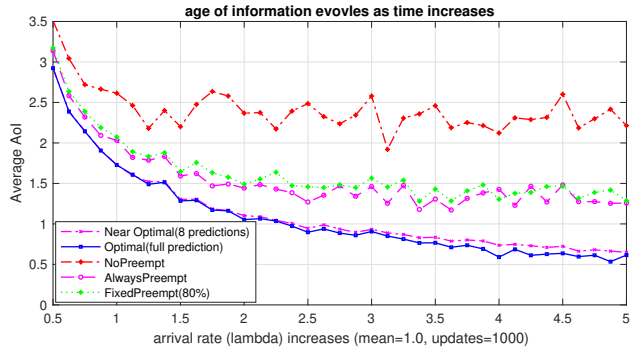

Fig. 9. *Arrival rate λ changes from $\lambda_L = 0.50$ to $\lambda_H = 5.00$ and the mean of service time $\mu = 1.00$. For each λ, 1000 updates are generated and composed of a long sequence. By comparing performance under the same groups of long sequences, this graph shows the difference between various policies.*

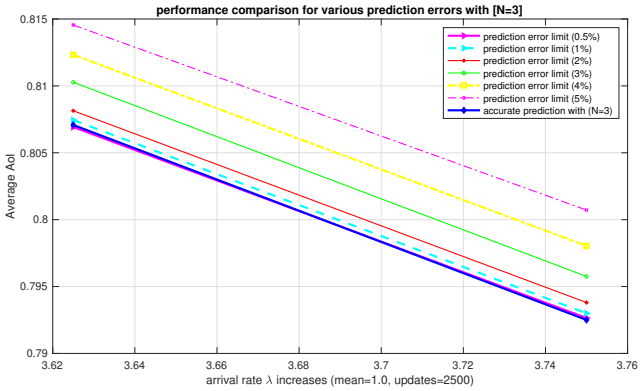

Fig. 10. *The maximum prediction errors of inter-arrival and service times could be possibly accepted by segmentation optimization algorithm, and it still can achieve optimal policy with fault tolerance. The curves show the deviations between different estimation errors 0.5%, 1%, 2%, 3%, 4%, 5% decrease as the arrival rates increase from 3.62 to 3.75.*

be accepted by predictive scheduling policy. The influence on average AoI is illustrated in the Fig.10 and the average AoI decreases as the arrival rate $\lambda \in [3.62, 3.75]$ increases. The bold blue line represents best near-optimal AoI under N-prediction scheduling policy with approximation algorithm when N=3 packets are predictable. Other lines in various colors reveal that average AoIs resulted from estimation errors 0.5%, 1%, 2%, 3%, 4%, 5% also increase as estimation errors become worse as expected.

The performance comparison between different number of predictable packets is shown in the Fig.11. When $N = 4$ number of future packets are predictable, the curve in red color is very close to the optimal curve in blue color which is generated under full prediction without estimation error. When there is no prediction for future packets, the predictive scheduling policy generates the yellow line in the most upper position. As the number $N = 1, 2, 3, 4$ of predictable future packets increases, the average AoI that it is able to achieve decreases rapidly.

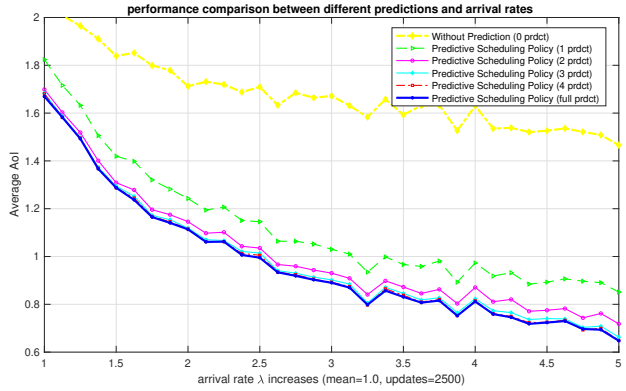

Fig. 11. *N-Prediction Algorithm generates near-optimal AoI under different number of packets $N = 0, 1, 2, 3, 4$. The lower curve in blue color represents AoI under full prediction without estimation error. As the number of predictable future packets increases, the average AoI that it is able to achieve decreases rapidly.*

VIII. Conclusion And Future Work

For a status update system with packet sequence predictable [10], this article introduced the segmentation approach and presented the mechanism of discovering epochs. Mathematical expressions for arbitrary epochs are derived, and numerical results are obtained to verify theoretical analysis. Fault tolerance of inaccurate prediction is also considered and the segmentation mechanism can effectively counteract the effect of prediction error on achieving optimal AoI. The proposed predictive scheduling policy with N number of predictable future packets efficiently reduces the requirement of predicting future packets. In future work, more queuing models [9] [12] get explored in the extensive analysis and this approach will be applied to the scheduling policies in multiple-source settings.

References

[1] S. Kaul, R. D. Yates and M. Gruteser. Real-time status: How often should one update? In *2012 Proceedings IEEE INFOCOM*, pages 2731–2735. IEEE, 2012.

[2] J. P. Champati, R. R. Avula, T. J. Oechtering and J. Gross. On the minimum achievable age of information for general service-time distributions. In *IEEE INFOCOM*, pages 456-465, 2020.

[3] Sun, Y., Uysal-Biyikoglu, E., Yates, R., Koksal, C. & Shroff, N. Update or wait: How to keep your data fresh. *IEEE Transactions On Information Theory*. **63**, 7492-7508 (2017)

[4] R. Devassy, G. Durisi, G. C. Ferrante, O. Simeone and E. Uysal-Biyikoglu. Delay and Peak-Age Violation Probability in Short-Packet Transmissions. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 2471-2475, 2018.

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

[6] Wang, Q., Chen, H., Gu, Y., Li, Y. & Vucetic, B. Minimizing the age of information of cognitive radio-based IoT systems under a collision constraint. *IEEE Transactions On Wireless Communications*. **19**, 8054-8067 (2020)

[7] Kadota, Igor and Modiano, Eytan Minimizing the Age of Information in Wireless Networks with Stochastic Arrivals. *IEEE Transactions on Mobile Computing*, 2019.

[8] Tong, P., Liu, J., Wang, X., Bai, B. & Dai, H. UAV-enabled age-optimal data collection in wireless sensor networks. *2019 IEEE International Conference On Communications Workshops (ICC Workshops)*. pp. 1-6 (2019)

[9] Champati, J., Al-Zubaidy, H. & Gross, J. On the distribution of AoI for the GI/GI/1/1 and GI/GI/1/2* systems: Exact expressions and bounds. *IEEE INFOCOM 2019-IEEE Conference On Computer Communications*. pp. 37-45 (2019)

[10] Phit, T. & Abe, K. Packet inter-arrival time estimation using neural network models. *Internet Conference, Tokyo*. (2006)

[11] R. D. Yates, Y. Sun, D. R. Brown, S. Kaul, E. Modiano, and S. Ulukus. Age of information: An introduction and survey. *IEEE Journal on Selected Areas in Communications*, 39(5):1183–1210, 2021.

[12] Zou, P. and Zhang, J. and Subramaniam, S. Performance Modeling of Scheduling Algorithms in a Multi-Source Status Update System. *IEEE International Symposium on Information Theory (ISIT)*. pp. 156-161, 2022.

[13] C. Li, S. Li, Y. Chen, Y. T. Hou and W. Lou. AoI scheduling with maximum thresholds. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 436–445. IEEE, 2020.

[14] Liu, J., Wang, X., Bai, B. & Dai, H. Age-optimal trajectory planning for UAV-assisted data collection. *IEEE INFOCOM 2018-IEEE Conference On Computer Communications Workshops (INFOCOM WKSHPS)*. pp. 553-558 (2018)

[15] O. T. Yavascan and E. Uysal. Analysis of slotted ALOHA with an age threshold. In *IEEE Journal on Selected Areas in Communications*, 39(5):1456–1470, 2021.

[16] Ayan, O., Vilgelm, M., Klügel, M., Hirche, S. & Kellerer, W. Age-of-information vs. value-of-information scheduling for cellular networked control systems. *Proceedings Of The 10th ACM/IEEE International Conference On Cyber-Physical Systems*. pp. 109-117 (2019)