Age-aware Scheduling for Asynchronous Arriving Jobs in Edge Applications

Jing Zhong*†, Wuyang Zhang*†, Roy D. Yates†, Andrey Garnaev‡ and Yanyong Zhang†§
†‡WINLAB, Rutgers University, USA, §University of Science and Technology of China (USTC), China
†{jzhong, wuyang, ryates, yyzhang}@winlab.rutgers.edu, ‡garnaev@yahoo.com

***Abstract*—Age of information has been proposed recently to measure information freshness, especially for a class of real-time video applications. These applications often demand timely updates with edge cloud computing to guarantee the user experience. However, the edge cloud is usually equipped with limited computation and network resources and therefore, resource contention among different video streams can contribute to making the updates stale. Aiming to minimize a penalty function of the weighted sum of the average age over multiple end users, this paper presents a greedy traffic scheduling policy for the processor to choose the next processing request with the maximum immediate penalty reduction. In this work, we formulate the service process when requests from multiple users arrive at edge cloud servers asynchronously and show that the proposed greedy scheduling algorithm is the optimal work-conserving policy for a class of age penalty functions.**

I. Introduction

The freshness of information is critical in real-time applications and systems, such as autonomous driving vehicles, virtual reality gaming, object tracking and facial recognition. These real-time applications share a common requirement, which is maintaining the freshness of data. Recently, an information freshness metric named the *age of information*, or simply age, has been proposed and applied to the evaluation of various status updating systems [1]–[5]. In [1], the real-time status updating system is modeled as a one-way communication between a source and destination pair over a communication channel. Such an "enqueue and forward" model assumes the source node receives randomly arriving information packets and selectively forwards them to the destination. Here, the channel capacity, and the channel busy/idle state, if it's available to the source, present as the impacting factors that the source node can refer to dynamically adjust the sending frequency.

Networking delay is only part of the story when the source node needs to further process the incoming information packets, especially when the computation overhead dominates the network transmission latency. Many real-time edge applications indeed demand an alternative "enqueue, process and forward" (EPF) model. For example, autonomous driving cars periodically, say every 20 ms, capture the front scenes with stereo cameras, and send them to a nearby edge cloud. The edge cloud is then required to perform heavy computer vision calculations [6]–[8] upon those received stereo images where the output involves the estimated depth of the surrounding objects appeared in the images or 3D point cloud. Those outputs will be delivered back to the autonomous driving cars (the destination nodes) for better understanding the traffic environment.

Timely environment updates are critical to guarantee the safety and efficiency of the driving experience. However, the age of those updates can grow substantially as edge clouds perform computer vision calculations. Importantly, unlike central clouds with nearly unlimited computing resources, edge clouds are typically constrained by their computing capabilities and might be over-utilized when the incoming traffic is heavy. Thus, resource contention among different video streams and the randomness of the processing time may significantly contribute to making information stale.

In this work, we examine the information freshness of an edge cloud computing system which supports real-time processing of multiple video streams. The edge cloud is simplified to a single processing unit that sequentially processes stereo video frames from multiple users. We view each video frame arriving at the edge cloud server as a job, and the monitor at the user itself is receiving the processed results as information updates. In this case, the source is self-updating itself through the closed-loop video frame processing at the edge cloud, and here we assume the processing times are i.i.d. across all jobs and all users. The age of an update is then defined as the difference between current time and the generation time of that particular job at the source. The objective of this work is to obtain the optimal scheduling policy for job processing so that the information freshness at each user is maintained.

There have been many relevant works on the scheduling of multiple users to minimize the age of information [9]–[13]. The scheduling of updates in an unreliable broadcast network with a base station and multiple receivers is considered in [9]. In this system, the base station accumulates updates from different sources but can only update at most one receiver at a time. A similar problem is considered in [11], in which an information update is discarded if it is not selected by the base station for transmission. Our work is motivated by the queueing model in [10], in which the job arrival times are synchronized among all sources. We note that the most relevant work to ours is [12]. It was shown by experiments that choosing the source with maximum age reduction leads to lower average age than several other schemes. A similar problem in cache updating in which the service facility can divide its capacity according to the update rate at different

* Co-primary authors.

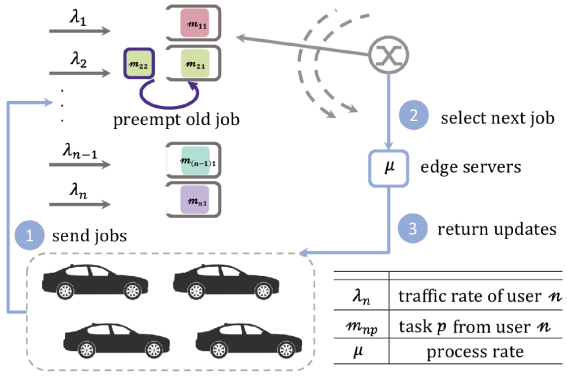

Fig. 1. Example of edge cloud traffic scheduling system.

sources is examined in [13].

In this work, we show that a greedy scheduling policy that chooses the user with maximum immediate penalty reduction is the optimal work-conserving policy for a class of penalty functions for the sum of the users' ages. If the penalty is simply the weighted sum of ages over all the users, we also prove that the optimal policy is to select the stream with maximum weighted age reduction[1]. We also validate our policy and demonstrate the fairness among different users by simulations.

II. System Overview

A. Scheduling Model

Here we assume the edge cloud server is shared by n self-driving vehicles, and we refer to each vehicle as a user. As illustrated in Fig. 1, each user i sequentially submits jobs to the edge cloud with rate λ_i, and each job is temporarily stored at a pre-processing buffer B_i that can hold a single user i job. In particular, an incoming job with newer generation time will replace an old job already stored in the buffer since the newer job always contains fresher information. Since each job is a video frame captured by the stereo camera, the job upload time from each user to the edge cloud is considered to be random. We denote m_{ij} as the j-th job from user i, and U_{ij} as the corresponding job upload time to the edge cloud. We assume the upload time U_{ij} is i.i.d. for all the jobs j corresponding to a user i. Once a job m_{ij} is generated at time $A_j^{(i)}$, it will be delivered to the buffer at time $A_j^{(i)} + U_{ij}$. At any time t, user i has submitted $N_i(t)$ jobs to the buffer, and the most recent job is generated at time $A_{N_i(t)}^{(i)}$, then the job stored at the buffer B_i has an instantaneous age

$$\delta_i(t) = t - A_{N_i(t)}^{(i)}. \tag{1}$$

We refer to $\delta_i(t)$ as the *buffer age* as it equals the age of an observer who views jobs arriving at buffer B_i as updates. Under this model, the age $\delta_i(t)$ is reset to the upload time U_{ij} of the most recent job when the buffered job is replaced by a new job.

[1]This policy is identical to MAD policy in [12] except that we consider non-preemptive system.

In this work, we assume the edge computing unit to be a single processor that can only handle one job at a time with processing rate μ identical across all users i and jobs j. We consider only *work-conserving* policies: the server is kept busy whenever the buffers $B_1, B_2, \ldots, B_n$ are non-empty; if the buffers are all empty, then the server stays idle and waits for the next arriving job. For a work-conserving policy, once the server finishes processing the previous job but the buffers are not empty, the scheduler selects the next job from the buffers as shown in Fig. 1.

Denote s_k as the processing time for job k and $\phi_k \in \{1, 2, \ldots, n\}$ as the indicator for the user corresponding to the k-th job processed by the server. We also assume that the processing time s_k is independent of the job arrival time. The processor starts the k-th job at time t_k and completes at time $t_k + s_k$. The processing result of job k, which is an information update, is then immediately sent back to the monitor at the corresponding user $\phi_k = i$. Since the processing result is much smaller than the original job size, and the downlink bandwidth is usually sufficiently large, we assume the download time is negligible. Here we define the age at the monitor of user i as the difference between the present time and user's knowledge about the environment. Since the received update at the monitor i contains the information generated at time $A_{N_i(t_k)}^{(i)}$, the instantaneous age at the monitor i is reset to

$$\Delta_i(t)|_{t=t_k+s_k} = t - A_{N_i(t_k)}^{(i)}. \tag{2}$$

After that, the age at the monitor increases linearly in time until the monitor receives another update.

We denote π as a scheduling policy that determines the job ϕ_k to be processed. In this model, the processor records the time stamp of each processed job, which is the generation time of these jobs. Thus, the processor also knows the set of instantaneous ages at each monitor $\Delta_i(t)$ at any time t. We let Π represent the set of causal work-conserving policies in which the scheduling decisions are made based on the history of the states of the system up to the present time. Here we only consider *non-preemptive* policies in which the processor must complete the processing of the current update before starting to serve another.

Fig. 2 demonstrates sample paths of both age processes $\delta_i(t)$ and $\Delta_i(t)$ for a particular user i. The first job is generated by user i at time $A_1^{(i)} = 0$, and arrives at the buffer B_i at time $A_1^{(i)} + U_{i1}$. Once its processing is completed, the age at the monitor Δ_i is reset to the age of the original job itself, which is also the buffer age δ_i. Both the third and fourth jobs are skipped by the scheduler since the processor is busy serving other users.

B. Sum Age Penalty Function

In this work, we consider the scenario in which vehicles are moving with different velocities, and thus have different age requirements. Let α_i to be the weighting factor associated

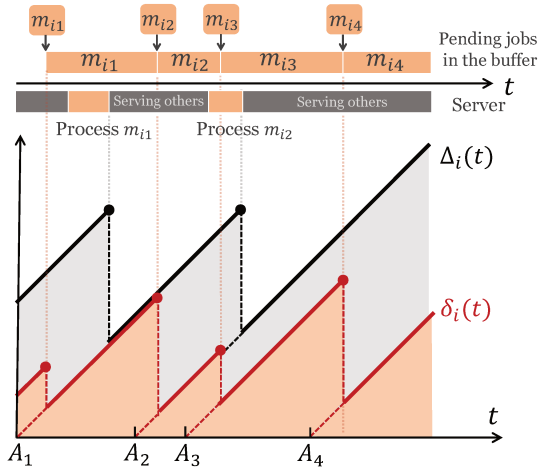

Fig. 2. Sample path of age process at the buffer δ_i and age at the user monitor Δ_i.

with the user i. Here we are interested in *sum age penalty functions* [10], which is a class of penalty functions defined as

$$\mathcal{P}_{\text{sum}}(t) = \sum_{i=1}^{n} \alpha_i f(\Delta_i(t)), \tag{3}$$

where $f : [0, \infty) \to \mathbb{R}$ is any non-decreasing penalty function for an individual user, which represents the dissatisfaction of the information staleness. Since the penalty should be zero if the information is timely, we usually impose the initial condition $f(0) = 0$. Some examples of penalty functions are:

1) The penalty is simply the age itself $f(\Delta) = \Delta$.
2) The exponential function of the age $f(\Delta) = e^{a\Delta} - 1$ where $a \geq 0$. This suits applications in which the need for information refresh is more desired as the information gets stale.
3) The fraction function $f(\Delta) = (a\Delta)/(a\Delta + b)$ that maps the age to the binary interval $[0, 1]$. This converts the age to a mission failure probability for many attack-defense problems and certain control applications [14].

The time-averaged penalty function is then defined as

$$\mathrm{E}[\mathcal{P}_{\text{sum}}(t)] = \lim_{\mathcal{T}\to\infty} \frac{1}{\mathcal{T}} \int_{t=0}^{\mathcal{T}} \sum_{i=1}^{n} \alpha_i f(\Delta_i(t)) \mathrm{d}t. \tag{4}$$

Our objective is to minimize time-averaged sum age penalty function over all n users by choosing the casual traffic scheduling policy π. We note that the scheduling policy π at any time t depend on the history of all prior states of the system, including:

1) the instant age at the monitors $\Delta_1(t), \Delta_2(t), \ldots, \Delta_n(t)$,
2) the generation time of j-th job $A_j^{(i)}$ from user i for all $j \leq N_i(t)$ up to the present time t.

III. Scheduling Policy

In order to describe the optimal scheduling policy, we first define the penalty reduction after the service by the processor. Assume that the processor becomes idle and there are at least one job waiting in the buffers $B_1, B_2, \ldots, B_n$ at time t_k, and it selects the job from user i for processing. Assuming this job is the k-th job at the processor and the service time is s_k time units, we observe in Fig. 2 that the age reduction for user i after the processing time s_k is

$$D_i(t_k + s_k) = \Delta_i(t_k) - \delta_i(t_k). \tag{5}$$

We remark that the processor obeys a non-preemptive scheduling policy, in which the server doesn't preempt any job being serviced by new incoming jobs. Thus, the age reduction at time $t_k + s_k$ depends on the difference between the age at the user monitor and buffer age, both at time t_k. At time t_k, the most recent served and delivered update for user i is denoted by $M_i(t_k)$, which is generated at time $A^{(i)}_{M_i(t_k)}$. On the other hand, the job in the buffer at time t_k is generated at time $A^{(i)}_{N_i(t_k)}$. From Fig. 2, we also observe that the age reduction $\Delta_i(t_k + s_k)$ after service is

$$D_i(t_k + s_k) = A^{(i)}_{N_i(t_k)} - A^{(i)}_{M_i(t_k)}. \tag{6}$$

That is, the reduction in age at user i after the service is exactly the inter-arrival time between the two most recent served jobs. In Fig. 2, the second job m_{i2} is selected for processing, and the third job m_{i3} arrives at the buffer during the processing period. After the service for m_{i2}, the reduction in age is $A_2 - A_1$ since the job m_{i2} relects the real-time information at time A_2. We also note that the job in the buffer is not necessarily the most recently generated one, since each job m_{ij} takes a random upload time U_{ij} to arrive at the buffer. Since the scheduler has direct access to the timestamp of the job $A_{N_i(t_k)}$ at each buffer i, the effect of the random upload time U_{ij} on the age reduction is not directly reflected in (6).

Since only one job can be processed at a time, only the age of the served user $\Delta_i(t_k+s_k)$ is reduced, and the age processes of all other users remain unchanged. Thus, the reduction of the sum age penalty function in (3) at time $t_k + s_k$ is

$$R_i(t_k + s_k) = \alpha_i \Big(f\big(\Delta_i(t_k + s_k)\big) - f\big(\Delta_i(t_k + s_k) - D_i(t_k + s_k)\big) \Big). \tag{7}$$

Although the scheduling decision is made at time t_k, the reduction occurs at time $t_k + s_k$ where the processor service time s_k is random and unknown to the scheduler. Thus, the scheduler knows the age reduction $D_i(t_k + s_k)$ in (6), but not the penalty reduction $R_i(t_k + s_k)$ in (7). We now consider the following greedy scheduling policy which is independent of the processor service time by assuming the processing can be completed instantaneously, i.e. $s_k = 0$, and the reduction occurs immediately.

Definition 1. Maximum Immediate Penalty Reduction (MIPR) Policy. *When the server becomes available and the buffers are not empty, the job from the user i with the maximum penalty reduction $R_i(t)$ at the present time t is served, with ties broken arbitrarily. That is, the scheduling indicator is*

$$\phi_k = \arg\max_i \alpha_i (f(\Delta_i(t_k)) - f(\Delta_i(t_k) - D_i(t_k))). \tag{8}$$

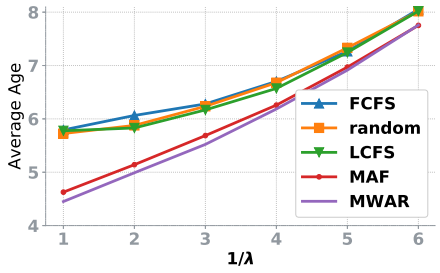

Fig. 3. Average age vs. average job arrival time with different scheduling policies.

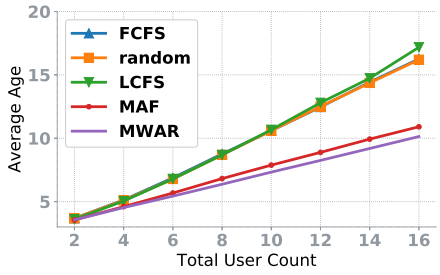

Fig. 4. Average age vs. number of users with different policies.

Intuitively. the greedy MIPR policy is optimal if the best decision made based on the current benefit still remains as the best decision in the future. This property brings some restrictions on the growth of the penalty function since the service time s_k is random.

Definition 2. *A penalty function f has **Base-Independent Growth (BIG)** if for any x and non-negative constant $s \geq 0$, there exists two penalty functions g_1 and g_2 such that*

$$f(x+s) = f(x)g_1(s) + g_2(s).$$

The definition of BIG states that the evolution of function f after s_k time units can be described by a multiplicative term $g_1(s)$ and a additive term $g_2(s)$, both depending on only s. Note that when the shift is $s = 0$, $f(x) = f(x)g_1(0) + g_2(0)$ holds for all x, which requires $g_1(0) = 1$ and $g_2(0) = 0$.

We note that $f(x) = e^{ax}$ is an example of the BIG penalty function since $f(x+s) = f(x)f(s)$. In step (13), the initial difference $f(x_1) - f(x_1 - d_1)$ is amplified by $f(s) = e^{as}$ as the time s increases. On the other hand, a linear function $f(x) = ax + b$ is another BIG penalty where $g_1(s) = 1$ and $g_2(s) = as$. In this example, $f(x_1 + s) - f(x_1 - d_1 + s) = f(x_1) - f(x_1 - d_1)$ only depends on the initial difference instead of the time difference s.

Theorem 1. *If the service times are identically distributed across all the jobs from all users, the MIPR policy is the optimal (1) causal, (2) work-conserving and (3) non-preemptive policy for BIG penalty function f, specifically*

$$\mathcal{P}_{\text{sum,MIPR}}(t) \leq_{st} \mathcal{P}_{\text{sum},\pi}(t), \tag{9}$$

for any $t \geq 0$ and any $\pi \in \Pi$, where $\leq_{st}$ is the stochastic ordering defined in [15].

It follows from Theorem 1 that

$$\mathrm{E}[\mathcal{P}_{\text{sum,MIPR}}] \leq \mathrm{E}[\mathcal{P}_{\text{sum},\pi}]. \tag{10}$$

The proof of Theorem 1 in the appendix follows the similar sample path technique used in [10]. One key idea used in the proof is the inductive comparison between two policies. By greedily choosing the user that gives the maximum penalty reduction, the instantaneous penalty after the service completion is always smaller than that in any other policy.

Definition 3. Maximum Weighted Age Reduction (MWAR) Policy. *When the server becomes available, the job from the user i with the maximum weighted age reduction $\alpha_i D_i(t_k)$ is served among all packets in the buffer, with ties broken arbitrarily.*

Corollary 1. *If the penalty function is $f(\Delta) = \Delta$, then MWAR is the age optimal MIPR policy.*

Corollary 1 follows directly from Theorem 1. In this special case, the scheduling policy is now independent of the current age of an individual user $\Delta_i(t_k)$. MWAR policy is also the maximum-age-first (MAF) policy in [10] if the job arrivals are synchronized among users.

IV. Performance Evaluation

In this section, we evaluate the effectiveness and the fairness of the proposed maximum weighted age reduction (MWAR) policy by considering the average age over users $1/n \sum_{i=1}^{n} \alpha_i \Delta_i$ as the penalty function. Here we let all the users to be equally weighted, $\alpha_i = 1$ for all i. We compare the MWAR policy with four other policies with dynamic system setup. The four reference policies are: (1) first-come-first-served (FCFS): the scheduler selects the job with earliest arrival time in the buffer; (2) last-come-first-served (LCFS): the scheduler selects the most recent arrived job in the buffer; (3) max age first (MAF): the scheduler compares the age of all users and selects the job corresponding to the user with maximum age; (4) random: the scheduler select on of the jobs in the buffer uniformly at random.

Fig. 3 compares the average age for each policy by fixing the processing rate $\mu = 1$ and varying the job arrival rate λ_i. The number of users is set to $n = 5$ and each user submits jobs according to Poisson process with average inter-update time $1/\lambda$. The service time is exponentially distributed and thus the average job processing time is $1/\mu = 1$. As the average job submission time $1/\lambda$ increases, all the curves increase and the gap between any two policies becomes smaller. This is mainly because the age becomes dominated by the idle time between updates instead of the processing delay, and the scheduling doesn't provide much performance gain. Among all five policies, MWAR policy gives the lowest average age. And the MAF policy, which is shown to be optimal when the job arrival times are synchronized in [10], provides slightly larger average age. On the other hand, the other three policies (FCFS, LCFS and random) lead to almost the same much larger average age regardless of the job arrival rate.

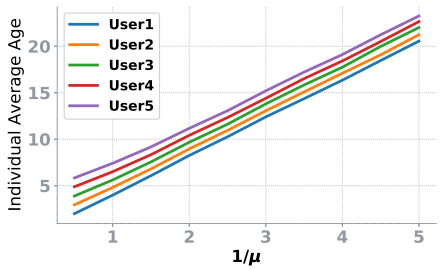

Fig. 5. Average age of each user in MWAR policy.

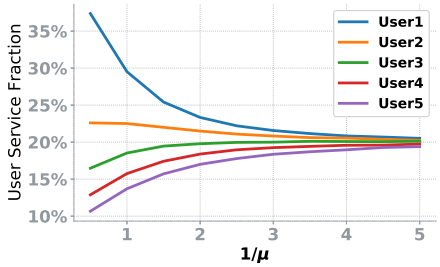

Fig. 6. The fraction of service corresponding to each user in MWAR policy.

Fig. 4 depicts the comparison with server processing rate $\mu = 1$ and user job submission rate $\lambda_i = 1/2$ by varying the total number of users n. As n increases, the processor becomes busier and thus MWAR provides larger performance gain. We also notice the average age grows almost linear as n increases.

While all the experiments in Fig. 3 sets all the user update rates λ_i identical for all i. In Fig. 5 and 6, we choose different job submission rate λ_i for each user i and evaluate how the scheduling policy treats users with different λ_i. Fig. 5 depicts the average age of each user Δ_i by varying the processing rate μ. The job submission rates for the $n = 5$ users are $\lambda_1 = 2, \lambda_2 = 1, \lambda_3 = 2/3, \lambda_4 = 1/2, \lambda_5 = 2/5$. As the average processing time $1/\mu$ increases, the individual average age increases almost linearly and the gap between any pair of users stays almost the same, which implies the MWAR policy keeps the difference between users regardless of the available resource of the service facility. Fig. 6 demonstrates the fraction of served job at the processor corresponding to each user. For example, when the average service processing time is $1/\mu = 1$, around 30% of jobs served by the processor are from user 1. When the processor is operating very fast, it can handle most of the jobs and thus the fraction of jobs is almost proportional to the rate of each user λ_i. As the processing time gets larger, we observe the scheduler starts to treat all users fairly and service in an equal way. Since every user experience long waiting time when the traffic load is high, the age for each user is almost equally large. As a result, the scheduler is busy serving every user one by one as soon as it finishes an old job.

V. Conclusion and Future Directions

In this paper, we proposed a greedy traffic scheduling policy that chooses the next processing request with maximum immediate penalty reduction, aiming to minimize the overall age penalty of multiple end users. The main contribution of our the proposed scheduling policy stems from its capability to provide minimal average age among multiple end users in multiple dynamic traffic environments, e.g., different server processing rates and incoming request frequencies. Importantly, this policy efficiently supports asynchronous job arrivals. Moving forward, we will continue to investigate how to perform traffic scheduling in a multi-server environment and with multi-stage jobs. We are also aware of other application scenarios where the transmission time of each job is not negligible because of the limited radio resources at the edge cloud. It's of our interest to study how to integrate scheduling policy with realistic distributed computing platforms, e.g., Apache Storm or Apache Spark.

References

[1] R. D. Yates and S. Kaul, "Real-time status updating: Multiple sources," in *Proc. IEEE Int. Symp. Inform. Theory*, Jul. 2012, pp. 2666–2670.
[2] C. Kam, S. Kompella, and A. Ephremides, "Age of information under random updates," in *Proc. IEEE Int. Symp. Inform. Theory*, Jul. 2013, pp. 66–70.
[3] Y. Sun, E. Uysal-Biyikoglu, R. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," in *Proc. INFOCOM*, 2016.
[4] E. Najm and R. Nasser, "Age of information: The gamma awakening," in *Proc. IEEE Int. Symp. Inform. Theory*, 2016, pp. 2574–2578.
[5] J. Zhong, E. Soljanin, and R. D. Yates, "Status updates through multicast networks," in *Proc. Allerton Conf. on Commun., Control and Computing*, 2017, pp. 463–469.
[6] W. Zhang, S. Li, L. Liu, Z. Jia, Y. Zhang, R. Yates, and D. Raychaudhuri, "Hetero-edge: Orchestration of real-time visionapplications on heterogeneous edge clouds," in *Proc. INFOCOM*, 2019.
[7] W. Zhang, J. Chen, Y. Zhang, and D. Raychaudhuri, "Towards efficient edge cloud augmentation for virtual reality mmogs," in *Proceedings of ACM/IEEE Symp. on Edge Computing (SEC)*, 2017.
[8] R. D. Yates, M. Tavan, Y. Hu, and D. Raychaudhuri, "Timely cloud gaming." *Proc. INFOCOM*, pp. 1–9, 2017.
[9] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Minimizing the Age of Information in broadcast wireless networks." *Proc. Allerton Conf. on Commun., Control and Computing*, pp. 844–851, 2016.
[10] Y. Sun, E. Uysal-Biyikoglu, and S. Kompella, "Age-optimal updates of multiple information flows," *arXiv preprint arXiv:1801.02394*, 2018.
[11] Y.-P. Hsu, "Age of Information - Whittle Index for Scheduling Stochastic Arrivals." in *Proc. IEEE Int. Symp. Inform. Theory*, 2018.
[12] H. B. Beytur and E. Uysal-Biyikoglu, "Minimizing Age of Information for Multiple Flows," *2018 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 1–5, Jun. 2018.
[13] J. Zhong, R. D. Yates, and E. Soljanin, "Two Freshness Metrics for Local Cache Refresh." in *Proc. IEEE Int. Symp. Inform. Theory*, 2018, pp. 1924–1928.
[14] A. Garnaev, M. Baykal-Gursoy, and H. V. Poor, "Security games with unknown adversarial strategies," *IEEE Trans. on Cybernetics*, vol. 46, no. 10, pp. 2291–2299, 2016.
[15] M. Shaked and J. G. Shanthikumar, *Stochastic orders*. Springer Science & Business Media, 2007.

Appendix A
Proof of Theorem 1.

We denote P as the MIPR policy and $\mathcal{P}_{\text{sum},\pi}(t)$ as the penalty function of policy π at time t. We will compare P and any other work-conserving policy $\pi \in \Pi$ on a sample path of $\mathcal{P}_{\text{sum}}(t)$.

For any sample path in policy P and π, we set the initial ages $\Delta_{i,P}(t=0) = \Delta_{i,\pi}(t=0)$ for users $i = 1, 2, \ldots, n$. The initial penalties are $\mathcal{P}_{\text{sum},P}(t=0) = \mathcal{P}_{\text{sum},\pi}(t=0)$. The system evolution is described by the following cases:

1) If no update completes in $[t', t'+s]$, the age process of every user $\Delta_i(t) = \Delta_i(t') + (t - t')$ for $t \in [t', t'+s]$.
2) If there is an update completion at time t, the age of the served user $\Delta_i(t)$ is reduced.

Now we define the following class of penalty functions.

Definition 4. *Function f is a **Present-Determines-Future (PDF)** function if f satisfies the following conditions:*

1) If a pair of n-tuple sequences $\{x_{1i}\}, \{x_{2i}\}$ and non-negative n-tuple constants $\{\alpha_i\}$ satisfy $\sum_{i=1}^n \alpha_i f(x_{1i}) \leq \sum_{i=1}^n \alpha_i f(x_{2i})$, then for any $s \geq 0$,

$$\sum_{i=1}^n \alpha_i f(x_{1i}+s) \leq \sum_{i=1}^n \alpha_i f(x_{2i}+s). \tag{11}$$

2) If x_1, x_2 and non-negative constants $\beta \geq 0, d_1 \geq 0, d_2 \geq 0$ satisfy

$$f(x_1) - f(x_1 - d_1) \geq \beta\big(f(x_2) - f(x_2 - d_2)\big),$$

then for any $s \geq 0$,

$$f(x_1+s) - f(x_1 - d_1 + s) \geq \beta\big(f(x_2+s) - f(x_2 - d_2 + s)\big).$$

Lemma 1. *If a penalty function f is BIG, then f is PDF.*

Proof. We need to show BIG f satisfies both conditions of PDF. For condition 1),

$$\begin{aligned}\sum_{i=1}^n \alpha_i f(x_{1i}+s) &= g_1(s)\sum_{i=1}^n \alpha_i f(x_{1i}) + \sum_{i=1}^n \alpha_i g_2(s) \\ &\leq g_1(s)\sum_{i=1}^n \alpha_i f(x_{2i}) + \sum_{i=1}^n \alpha_i g_2(s) \\ &= \sum_{i=1}^n \alpha_i f(x_{2i}+s). \end{aligned}\tag{12}$$

Similarly, condition 2) is met as follows

$$\begin{aligned} & f(x_1+s) - f(x_1 - d_1 + s) \\ =& \Big(f(x_1)g_1(s) + g_2(s)\Big) - \Big(f(x_1-d_1)g_1(s) + g_2(s)\Big) \\ =& \Big(f(x_1) - f(x_1-d_1)\Big)g_1(s) \\ \geq& \beta\Big(f(x_2) - f(x_2-d_2)\Big)g_1(s) \qquad (13) \\ =& \beta\Big(f(x_2+s) - f(x_2-d_2+s)\Big). \qquad (14) \end{aligned}$$

□

Now we can start the proof of Theorem 1 by the following lemma about the first case with no update completion.

Lemma 2. *For PDF f, if $\mathcal{P}_{\text{sum},P}(t) \leq \mathcal{P}_{\text{sum},\pi}(t)$ and there is no update completion between t and $t+s$, then*

$$\mathcal{P}_{\text{sum},P}(t+s) \leq \mathcal{P}_{\text{sum},\pi}(t+s). \tag{15}$$

The proof follows directly from the condition 1) in the definition of PDF by setting $x_{1i} = \Delta_{iP}(t)$ and $x_{2i} = \Delta_{i\pi}(t)$ for $i = 1, 2, \ldots, n$. Note that Lemma 2 guarantees that given the sum age penalty in policy P is smaller than that in policy π at some time t, the same ordering holds for any time beyond t if there is no update completion.

Now we move to the second case with an update completion at time $t_k + s_k$. Whether under policy P or policy π, an update is serviced from time t_k to $t_k + s_k$. The penalty of policy P is $\mathcal{P}_{\text{sum},P}$ before the completion and becomes $\mathcal{P}'_{\text{sum},P}$ after the completion. Similarly, the penalty of policy π is $\mathcal{P}_{\text{sum},\pi}$ before the completion and becomes $\mathcal{P}'_{\text{sum},\pi}$ after the completion. All policies have the same update arrival process and service process. We first prove the following lemma about inductive comparison between two sample paths.

Lemma 3. *For PDF function f, if $\mathcal{P}_{\text{sum},P} \leq \mathcal{P}_{\text{sum},\pi}$, then $\mathcal{P}'_{\text{sum},P} \leq \mathcal{P}'_{\text{sum},\pi}$.*

Proof. When job k is to go into service at time t_k, the MIPR policy P chooses the user $\phi_k = \arg\max_i R_i(t_k)$. At the service completion time $t_k + s_k$,

$$\mathcal{P}'_{\text{sum},P}(t_k+s_k) = \mathcal{P}_{\text{sum},P}(t_k+s_k) - \max_i R_i(t_k). \tag{16}$$

By the MIPR policy, choosing user i yields larger immediate penalty reduction than choosing user j, $R_i(t_k) \geq R_j(t_k)$ and

$$\begin{aligned} & f(\Delta_i(t_k)) - f(\Delta_i(t_k) - D_i(t_k)) \\ \geq& \frac{\alpha_j}{\alpha_i}\big[f(\Delta_j(t_k)) - f(\Delta_j(t_k) - D_j(t_k))\big] \end{aligned}\tag{17}$$

Since the age reductions D_i and D_j are independent of the service time s_k, by the definition of a PDF function we have

$$\begin{aligned} & f(\Delta_i(t_k+s_k)) - f(\Delta_i(t_k+s_k) - D_i(t_k+s_k)) \\ &= f(\Delta_i(t_k+s_k)) - f(\Delta_i(t_k+s_k) - D_i(t_k)) \\ &\geq \frac{\alpha_j}{\alpha_i}\big[f(\Delta_j(t_k+s_k)) - f(\Delta_j(t_k+s_k) - D_j(t_k))\big] \\ &= \frac{\alpha_j}{\alpha_i}\big[f(\Delta_j(t_k+s_k)) - f(\Delta_j(t_k+s_k) - D_j(t_k+s_k))\big]. \end{aligned}\tag{18}$$

Hence, the penalty reduction at time $t_k + s_k$ is $R_i(t_k+s_k) \geq R_j(t_k+s_k)$. Thus,

$$\arg\max_i R_i(t_k) = \arg\max_i R_i(t_k+s_k). \tag{19}$$

The penalty of any policy π after the service completion is

$$\begin{aligned} \mathcal{P}'_{\text{sum},\pi}(t_k+s_k) &= \mathcal{P}_{\text{sum},\pi}(t_k+s_k) - R_i(t_k+s_k) \\ &\geq \mathcal{P}_{\text{sum},\pi}(t_k+s_k) - \max_i R_i(t_k+s_k) \\ &\geq \mathcal{P}_{\text{sum},P}(t_k+s_k) - \max_i R_i(t_k+s_k) \\ &= \mathcal{P}'_{\text{sum},P}(t_k+s_k). \end{aligned}\tag{20}$$

□

Now given that the penalty function evolves under the condition of either Lemma 2 and 3, by induction over time, we have $\mathcal{P}_{\text{sum},P}(t) \leq \mathcal{P}_{\text{sum},\pi}(t)$, for all $t \geq 0$. And thus

$$\mathrm{E}[\mathcal{P}_{\text{sum},P}(t)] \leq \mathrm{E}[\mathcal{P}_{\text{sum},\pi}(t)]. \tag{21}$$

for any casual work-conserving policy π.