Aequitas: A Uniformly Fair 5G Scheduler for Minimizing Outdated Information

Chengzhang Li[†] Qingyu Liu[†] Y. Thomas Hou[†] Wenjing Lou[†] Sastry Kompella[‡]
[†]Virginia Polytechnic Institute and State University, Blacksburg, VA
[‡]US Naval Research Laboratory, Washington, DC

Abstract—**Age of information (AoI) is a key metric to measure the freshness of information for IoT applications. This paper investigates a scheduling problem in a 5G network where there is an AoI threshold for each source node at the edge base station (BS). Our goal is to design a 5G scheduler to minimize the proportion of time when the information stored at the source is outdated, i.e., when the AoI is beyond its threshold. For performance benchmark, we develop an offline computational procedure to find a lower bound for the objective value. Then we derive a property called uniform fairness for an offline optimal scheduler and use this property as a guideline to develop an online 5G scheduler—Aequitas. To meet the sub-millisecond real-time requirement in 5G, we implement Aequitas on an off-the-shelf GPU. Through experiments, we show that the objective achieved by Aequitas is close to the lower bound and its running time is under 1 ms.**

I. Introduction

Age of Information (AoI) is an application-layer metric for latency that was first conceived by Kaul *et al.* [1], [2]. It is defined as the elapsed time for a sample (stored at a particular location, e.g., edge or cloud) between current time and the time when the sample was first generated at its source. AoI has been used as a measure of information freshness for the sample stored at a location.

There has been active research on designing schedulers for AoI related objectives (see an online bibliography in [3]). However, most of existing efforts share two common limitations. First, a majority of existing literature considers static channels or perfect channel state knowledge for tractability [4]–[7]. However, in most real-world scenarios, such an assumption does not hold. For example, wireless channel conditions are highly dynamic and unknown *a priori.* As such, an optimal scheduler with guaranteed performance is hardly possible. One must address issues with unknown future knowledge and design an online scheduler to cope with non-idealized scenarios in practice. Second, existing literature on AoI scheduler design employs overly simplified models, which can hardly be deployed for real systems [8]–[15]. As real-world communication systems must be standards compliant, it is important to design schedulers that conform to standards from the very beginning of the design phase (rather than an afterthought of theoretical exercise). As 5G NR has been widely adopted for wireless communications, including IoT applications, it's of utmost importance to design AoI schedulers that conform to 5G standards [16].

To address these limitations, we focus on designing *online* AoI schedulers that are 5G-compliant. We consider a canonical setting where a group of source nodes collect information and forward it to a base station (BS) through a shared wireless channel. At the BS, we assume that there is a unique AoI threshold for information (sample) from each source. Given the unknown nature of future channel conditions, it is unrealistic to demand a hard guarantee of AoI threshold for each source. Instead, it is more reasonable to minimize the proportion of time for each source when its information is outdated (i.e., AoI threshold is exceeded). In particular, it is plausible to ask for a scheduler that minimizes the maximum proportion of outdated information among all source nodes.

There are a number of challenges to design such a scheduler. First, an online scheduler is intrinsically difficult to design, due to the unknown knowledge of the future (e.g., channel conditions). Second, there are some unique considerations associated with 5G scheduling, such as frequency-time resource grids (RBs) and modulation and coding scheme (MCS) [17]. In each transmission time interval (TTI) the scheduler needs to allocate $\sim$100 RBs to $\sim$100 source nodes, as well as select one MCS (from 29 levels) for each source node, which presents an extremely large search space. Finally, a 5G scheduler has a stringent real-time requirement: the scheduler must make scheduling decisions within one TTI, which is in sub-millisecond time scale. Designing a scheduler that meets all the above three requirements is not a trivial task, especially for the AoI optimization objective that we wish to achieve in this paper. The main contributions of this paper are the following:

- We investigate a 5G IoT network for data collection with the objective of finding an online scheduler to minimize the maximum proportion of outdated information among all source nodes. For performance benchmark, we develop an offline computational procedure to find a lower bound for the objective value. The lower bound is obtained via a series of relaxation and reformulation, following which an effective technique can be applied to solve the reformulated optimization problem.
- We develop an important and interesting property, called *uniform fairness*, that is associated with an offline optimal scheduler. It says that there exists an optimal offline scheduler to our objective, under which the proportion of outdated information by all source nodes should converge to the same value as time becomes large, regardless of the difference in AoI thresholds and sample sizes among

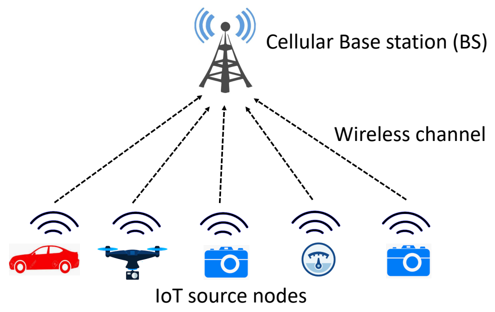


Figure 1: System model: N source nodes collect information and send it to a BS.

the source nodes. This interesting property offers us an important guideline in our design of an online scheduler where we only have knowledge of the past and present, not the future.

- We present Aequitas—an online 5G scheduler to optimize our objective. At the heart of Aequitas is a novel priority metric that takes the AoIs, AoI thresholds, channel conditions and historical performance behavior into consideration. By computing this priority metric iteratively in each time slot (TTI), Aequitas performs RB allocation and MCS selection for the source nodes that are selected for transmission.
- Although the time complexity of Aequitas is polynomial, its running time is still too long to meet the 5G timing requirement. We propose to exploit Aequitas' intrinsic property (amenable to parallel computation) and implementent it on an off-the-shelf GPU platform. Experimental results show that Aequitas can achieve excellent performance in terms of objective function (i.e., close to the lower bound) and its running time is under 1 ms.

II. System Model and Problem Statement

In this section we present the 5G data collection network model and state the min-max scheduling problem in this paper.

A. System Model

We consider a 5G-based IoT network where N source nodes collect information and forward it to a base station (BS) (see Fig. 1). Each source node takes a new sample at the beginning of each transmission time interval (TTI). For source node i, denote L_i as the sample size (in unit of bits), which is the amount of information carried in a sample. We assume the sample size (L_i) only depends on its source and is invariant over time. The BS maintains the most recent (freshest) sample from each source. Once a new sample from a source node is received completely by the BS, the BS deletes the previous sample and replaces it with this new one.

The source nodes transmit information (collected samples) to the BS through a 5G uplink channel. In 5G, uplink transmission resource is organized into 2-dimensional grids of *resource blocks* (RBs) that span both time and frequency domains [17]. In the time domain, time is equally slotted into TTIs, while in the frequency domain, bandwidth is equally slotted into a large number of sub-carriers, and 12 sub-carriers over a TTI is called an RB. RB is the smallest scheduling unit in 5G, and in each TTI there is a large number of RBs that can be allocated to the source nodes for uplink transmission. Due to channel fading (both time-selective and frequency-selective), the channel conditions on different RBs are different in general, even with respect to the same source node.

The BS employs a scheduler to allocate the uplink RBs to the source nodes based on the channel conditions in each TTI. Denote B as the total number of RBs that is available for uplink transmission in each TTI. We assume one RB can be allocated to at most one source node in each TTI. Denote $x_i^b(t)$ as a binary variable indicating whether RB $b \in \{1, 2, \cdots, B\}$ is allocated to source node i at TTI t. We have

$$x_i^b(t) = \begin{cases} 1 & \text{if RB } b \text{ is allocated to node } i \text{ at TTI } t, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\sum_{i \in \mathcal{N}} x_i^b(t) \le 1, \;\; b \in \{1, 2, \cdots, B\}. \tag{1}$$

In each TTI, the scheduler also needs to choose a modulation and coding scheme (MCS) for each source node [17]. The MCS of each source node determines the modulation and coding rate—how much information (in unit of bits) is modulated and coded within each RB for this source node. The higher the MCS is, the higher the modulation and coding rate is. On the other hand, the maximum amount of information that can be successfully transmitted by an RB depends on the channel condition. If the channel condition for this RB is poor and the source node uses a high MCS, information carried in the RB will not be successfully decoded by the BS.

Under 5G, there are $M = 29$ different levels of MCSs [17]. We assume $m = 1$ is the lowest MCS and $m = 29$ is the highest MCS. Denote $q_i^b(t)$ as the maximum MCS level that source node i's channel can support on RB b at TTI t. We have:

$$0 \le q_i^b(t) \le M.$$

In practice, $q_i^b(t)$ is determined by the channel quality indicator (CQI) report carried in the feedback from source node i at TTI $(t-1)$. Denote c^m as the modulation and coding rate under MCS level m, which can be found in Table 5.1.3.1-1 in [17]. Denote $r_i^{b,m}(t)$ as the achievable data rate by RB b w.r.t. source node i under MCS m. If $m \le q_i^b(t)$, the transmission is successful and the achievable data rate is c^m. Otherwise, i.e., $m > q_i^b(t)$, the transmission is unsuccessful and the achievable data rate is 0. We have:

$$r_i^{b,m}(t) = \begin{cases} c^m & \text{if } m \le q_i^b(t), \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Note that although each RB can only be allocated to at most one source node in a TTI, a source node may be allocated with multiple RBs. For a source node allocated with multiple RBs, it must choose and use one MCS $m \in \{1, 2, \cdots, M\}$ for all its allocated RBs [17]. Denote $y_i^m(t)$ as a binary variable indicating whether MCS $m \in \{1, 2, \cdots, M\}$ is chosen by source node i at TTI t, i.e.,

$$y_i^m(t) = \begin{cases} 1 & \text{if MCS } m \text{ is chosen for source } i \text{ at TTI } t, \\ 0 & \text{otherwise.} \end{cases}$$

We have

$$\sum_{m=1}^{M} y_i^m(t) \le 1, \ i \in \{1, 2, \cdots, N\}. \quad (3)$$

Denote $R_i(t)$ as the amount of information transmitted by source node i in TTI t across all RBs allocated to it. We have

$$R_i(t) = \sum_{b \in \mathcal{B}} \sum_{m \in \mathcal{M}} x_i^b(t) y_i^m(t) r_i^{b,m}(t), \ i \in \{1, \cdots, N\}. \quad (4)$$

Recall that the sample size for source i is L_i. In TTI t, if $R_i(t) \ge L_i$, then this sample can be transmitted completely within this TTI. Otherwise, part of the sample will be left to the following TTI(s) for transmission.

B. AoI Notation

AoI is location-dependent. AoI at the BS is defined as the elapsed time between the present and the time when the sample (stored at the BS) was generated at its source [1], [2]. For the most recent sample from source i that is currently maintained by the BS, denote $U_i(t)$ as its generation time. Then the AoI for source node i at the BS, denoted as $A_i(t)$, is:

$$A_i(t) = t - U_i(t). \quad (5)$$

The transmission of a sample under our 5G model may span multiple TTIs. Denote $\tau_i(n)$ (in unit of TTIs) as the time duration from the beginning of the TTI when the transmission starts to the end of the TTI when the transmission ends for the n-th sample ($n = 1, 2, \cdots$) from source node i. Clearly, $\tau_i(n)$ is an integer with $\tau_i(n) \ge 1$.

- At the end of TTI t, if no sample from source node i arrives at the BS, then at the beginning of TTI $(t+1)$ its AoI at the BS will increase by one.
- On the other hand, if the n-th sample arrives at the end of TTI t (i.e., TTI t is the last time slot of $\tau_i(n)$), then the new sample's generation time is the beginning of TTI $t-\tau_i(n)+1$.[1] Then at the beginning of TTI $(t+1)$, the BS has the complete new sample and $U_i(t+1) = t+1-\tau_i(n)$. We have:

$$A_i(t+1) = t + 1 - U_i(t+1) = \tau_i(n).$$

Combining the two cases, we have:

$$A_i(t+1) = \begin{cases} A_i(t) + 1, & \text{if no sample arrives in } t, \\ \tau_i(n), & \text{if } n\text{-th sample arrives in } t. \end{cases} \quad (6)$$

[1]We assume the time to collect a sample at a source is instant and does not take any TTIs.

C. Problem Statement

In this paper, we assume that each information source has an AoI threshold limit at the BS, denoted by d_i. Due to the heterogeneity of IoT applications, d_i may vary among different sources. For each source node i, when its AoI (of the stored sample) at the BS is no greater than d_i, we consider the information is *fresh*; when its AoI is greater than d_i, we consider the information as *outdated* (not fresh). Our goal is to minimize the proportion of time when the information at the BS is outdated across all source nodes.

More formally, denote v_i as the proportion of TTIs when the information from source i is outdated at the BS. We have:

$$v_i = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} [A_i(t) > d_i], \quad (7)$$

where "$[\cdot]$" is Iverson bracket, returning 1 if the inside statement is true and 0 otherwise. Denote $v_{\max}$ as the largest among all v_i's, i.e.,

$$v_{\max} = \max_{i \in \{1,2,\cdots,N\}} v_i. \quad (8)$$

So a plausible objective is to minimize $v_{\max}$. This is the objective of our optimization problem. Note that $v_{\max}$ is also a long-term metric, just as the v_i's.

To recap, we want to design a 5G scheduler that allocates B RBs to N source nodes in each TTI, and to choose one MCS from M levels for each source node so that $v_{\max}$ is minimized. We have the following optimization problem.

$$\begin{aligned} \text{OPT:} \ \min \ & v_{\max} \\ \text{s.t.} \ & \text{RB allocation constraint (1),} \\ & \text{MCS selection constraint (3),} \\ & \text{Calculation of data rates (2), (4),} \\ & \text{Calculation of AoI (6),} \\ & \text{Calculation of outdated proportion (7), (8).} \end{aligned}$$

In problem OPT, the decision variables are $x_i^b(t)$'s and $y_i^m(t)$'s for each TTI t.

There are a number of technical challenges to the design of a scheduler. First and foremost, the scheduler will be an online algorithm, without any knowledge of future information (including channel conditions) So it is impossible to obtain a provably optimal scheduler. Second, the search space of OPT is very large. In each TTI there are B^N possibilities for $x_i^b(t)$'s and N^M possibilities for $y_i^m(t)$'s (e.g., when $B = N = 100$ and $M = 29$, the search space consists of 1.7×10^{346} possibilities). Finding a near-optimal solution from this search space is challenging. Finally, a 5G scheduler has a stringent real-time requirement. The scheduler must make scheduling decisions within one TTI, which is in sub-millisecond time scale under 5G. It is very challenging to find a near-optimal scheduling solution in such a time scale.

III. Performance Bound

Since the scheduler that we will design addresses an online scheduling problem (with no knowledge of the future), it is impossible to find a provably optimal one. Nevertheless, in this section we will derive a lower bound for the objective $v_{\max}$, which can serve as a benchmark for the performance of any proposed scheduler.

A. A New Objective Function for Lower Bound

For any scheduler π, denote $\overline{R}_i$ as the long-term average data rate for source node i, i.e.,

$$\overline{R}_i = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} R_i(t). \tag{9}$$

Denote p_i as the fraction of TTIs when samples from source i arrive at the BS. Recall the sample size for source i is L_i. It is easy to see p_i is proportional to $\overline{R}_i$ with a factor L_i, i.e.,

$$\overline{R}_i = p_i \cdot L_i. \tag{10}$$

Note that a new sample from source i can ensure AoI at the BS for that source not to exceed its threshold limit for no more than d_i TTIs. So with p_i, the fraction of TTIs with AoI for source i being under its threshold limit is no greater than $p_i d_i$. Recall $(1-v_i)$ is the proportion of TTIs with source i's AoI being under d_i. Therefore, we have

$$1 - v_i \le p_i d_i = \frac{\overline{R}_i d_i}{L_i}, \tag{11}$$

which is equivalent to

$$v_i \ge 1 - \frac{\overline{R}_i d_i}{L_i}. \tag{12}$$

For $v_{\max}$ in (8), we have

$$v_{\max} \ge \max_{i\in\{1,2,\cdots,N\}} \{1 - \frac{\overline{R}_i d_i}{L_i}\}. \tag{13}$$

To find a lower bound to OPT, we can relax its objective $v_{\max}$ to the RHS of (13). So we have the following new optimization problem for the lower bound (LB):

$$\text{OPT-LB: } \min \max_{i\in\{1,2,\cdots,N\}} \{1 - \frac{\overline{R}_i d_i}{L_i}\}$$
$$\text{s.t.}\quad \text{Constraints (1), (2), (3), (4).}$$

Since optimal objective value of OPT-LB is always no greater than the objective of OPT, it can can serve as a lower bound for $v_{\max}$. Note that constraints (6), (7) and (8) are no longer included in OPT-LB because they do not affect either the new objective function or other constraints in OPT-LB.

B. Reformulation and Relaxation

Instead of working with $\{1-\frac{\overline{R}_i d_i}{L_i}\}$ in the $\min\max$ function, we can rewrite the objective function in OPT-LB as $\max\min$ function as follows:

$$\text{OPT-LB2: } \max \min_{i\in\{1,2,\cdots,N\}} \{\frac{\overline{R}_i d_i}{L_i}\}$$
$$\text{s.t.}\quad \text{Constraints (1), (2), (3), (4).}$$

Clearly, OPT-LB and OPT-LB2 are equivalent.

OPT-LB2 is a scheduling problem to maximize a utility function of $\overline{R}_i$. In [18] the authors studied a similar problem and showed that a gradient scheduling algorithm can achieve the optimal objective value asymptotically (when the number of TTIs goes to infinity). However, the utility function in [18] is required to be a concave smooth function. But the utility function in OPT-LB2 ($\min_i\{\frac{\overline{R}_i d_i}{L_i}\}$) isn't smooth. To address this issue, we will perform a relaxation for the utility function as follows.

Define a *smooth min* function S_α with parameter $\alpha > 0$ for $x_1, x_2, \cdots, x_N > 0$ as:

$$S_\alpha(x_1, x_2, \cdots, x_N) = \frac{\sum_{i=1}^{N} x_i e^{-\alpha x_i}}{\sum_{i=1}^{N} e^{-\alpha x_i}}. \tag{14}$$

Since $S_\alpha(x_1, x_2, \cdots, x_N) > \min\{x_1, x_2, \cdots, x_N\}$ for any $x_1, x_2, \cdots, x_N > 0$, we can perform a relaxation to OPT-LB2 as following, which we denote as OPT-LB-α:

$$\text{OPT-LB-}\alpha\text{: } \max S_\alpha(\frac{\overline{R}_1 d_1}{L_1}, \frac{\overline{R}_2 d_2}{L_2}, \cdots, \frac{\overline{R}_N d_N}{L_i})$$
$$\text{s.t.}\quad \text{Constraints (1), (2), (3), (4).}$$

Clearly, the optimal objective of OPT-LB-α, denoted by S_α^*, is always greater than the optimal objective of OPT-LB2. Then $(1-S_\alpha^*)$ is always smaller than the optimal objective of OPT-LB, so it can serve as a lower bound for $v_{\max}$.

Note that when $\alpha \to \infty$, $S_\alpha(x_1, x_2, \cdots, x_N) = \min\{x_1, x_2, \cdots, x_N\}$. So we always choose a large α to tighten the relaxation.

C. Solving OPT-LB-α

In OPT-LB-α, the objective $S_\alpha(x_1, x_2, \cdots, x_N)$ is smooth and concave. It has been shown in [18] that the so-called *gradient scheduling algorithm* can be used to solve it.

Gradient Scheduling Algorithm In a gradient scheduling algorithm, an empirical data rate $R_i^e(t)$ is defined for each TTI t and updated as an exponentially smoothed average as follows:

$$R_i^e(t+1) = (1-\beta)\cdot R_i^e(t) + \beta \cdot R_i(t), \tag{15}$$

where β is a small positive constant (e.g., 0.01) and $R_i(t)$ is the instantaneous data rate at TTI t and is given in (4).

Denote $\boldsymbol{R}^e(t) = [R_1^e(t)\ R_2^e(t)\ \cdots\ R_N^e(t)]^T$ and $\boldsymbol{r} = [r_1\ r_2\ \cdots\ r_N]^T$. The gradient scheduling algorithm solves OPT-LB-α by maximizing the following gradient-based objective function:

$$\sum_{i=1}^{N} \frac{\partial S_\alpha(\frac{r_1 d_1}{L_1}, \frac{r_2 d_2}{L_2}, \cdots, \frac{r_N d_N}{L_N})}{\partial r_i}\Big|_{\boldsymbol{r}=\boldsymbol{R}^e(t)} R_i(t)$$

in each TTI t. Note that

$$\frac{\partial S_\alpha}{\partial r_i}\Big|_{\boldsymbol{r}=\boldsymbol{R}^e(t)} = \frac{d_i e^{-\frac{\alpha d_i R_i^e(t)}{L_i}}\left(1+\alpha(S_\alpha - \frac{d_i R_i^e(t)}{L_i})\right)}{L_i \sum_{j=1}^{N} e^{-\frac{\alpha d_j R_j^e(t)}{L_j}}}. \tag{16}$$

Denote the RHS of (16) as $g_i(t)$. Then the gradient scheduling algorithm aims to maximize $\sum_{i=1}^{N} g_i(t) R_i(t)$ in every TTT t.

Algorithm 1

```
Input: $\alpha$, $\beta$, $T$, $q_i^b(t)$ for all $i$'s, $b$'s and $t$'s.
Output: A lower bound for $v_{\max}$.
1: Set $R_i^e(0) = 0$ for each $i = 1, 2, \cdots, N$.
2: for $t = 1, 2, \cdots, T$ do
3:     Solve OPT-LB-$t$ and get the optimal solution $x_i^b(t)$'s
       and $y_i^m(t)$'s.
4:     Use $x_i^b(t)$'s and $y_i^m(t)$'s to perform scheduling at TTI
       $t$ and get the data rate $R_i(t)$ for each $i = 1, 2, \cdots, N$.
5:     Use (15) to update $R_i^e(t)$ for $i = 1, 2, \cdots, N$.
6: end for
7: Let $\overline{R}_i = R_i^e(T)$ for each $i = 1, 2, \cdots, N$.
8: Substitute $\overline{R}_i$'s into the objective of OPT-LB-$\alpha$ to get $S_\alpha^*$,
   and use $(1 - S_\alpha^*)$ as a lower bound for $v_{\max}$. If $S_\alpha^* > 1$,
   then use 0 as the lower bound.
```

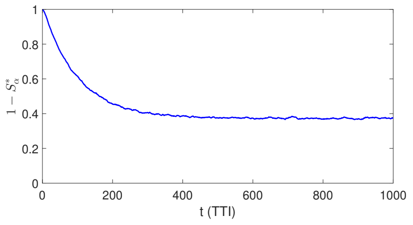


Figure 2: An illustration of convergence behavior of Algorithm 1 when $\alpha = 10$ and $\beta = 0.01$.

With $R_i(t)$ given in (4), we have the following optimization problem OPT-LB-t in every TTI t:

$$\text{OPT-LB-}t\text{: } \max \sum_{i\in\mathcal{N}}\sum_{b\in\mathcal{B}}\sum_{m\in\mathcal{M}} g_i(t) r_i^{b,m}(t) x_i^b(t) y_i^m(t)$$
$$\text{s.t. Constraints (1), (2), (3),}$$

where the decision variables are $x_i^b(t)$'s and $y_i^m(t)$'s. Problem OPT-LB-t is an integer quadratic program (IQP), which can be solved by a commercial solver such as CPLEX [19]. We need to solve OPT-LB-t for a large number (say T) of TTIs. Then we let $\overline{R}_i = R_i^e(T)$ and substitute $\overline{R}_i$'s into the objective of OPT-LB-α to get S_α^*—the optimal objective to OPT-LB-α. And we can use $(1 - S_\alpha^*)$ as a lower bound of $v_{\max}$.

Algorithm 1 presents a pseudocode for using the gradient scheduling algorithm to find a lower bound for $v_{\max}$. Note that $v_{\max}$ cannot be less than 0. So if the algorithm gives a value less than 0, we can instead use 0 as the lower bound.

Example Consider a network with $N = 100$ and $B = 100$ with the simulation settings in Section V-A. For $\alpha = 10$ and $\beta = 0.01$, we run Algorithm 1 for $T = 1,000$ TTIs and show the convergence behavior of $1 - S_\alpha^*$ (which we use as a lower bound for $v_{\max}$) in Fig. 2. We can see that setting $T = 1,000$ is adequate as the terminating time. At $T = 1,000$, the value of $(1 - S_\alpha^*)$ is 0.367, which is the lower bound for this example.

IV. Algorithm Design

In this section we present Aequitas[2]—a real-time scheduler to solve problem OPT.

[2]Aequitas is the Latin concept of justice, equality, and fairness.

A. Uniform Fairness

Denote π^* as an ideal offline optimal scheduler to OPT. By "ideal", we assume π^* has all future channel information when performing scheduling. Denote $v_1^*, v_2^*, \cdots, v_N^*$ as the outdated proportions for sources 1 to N under π^*. Recall that each source node i has its own sample size L_i and the channel conditions vary for different sources. So we ask the following question: Will v_i^*'s be different for different source nodes? The answer to this question is given in the following lemma.

Lemma 1 *There always exists an optimal offline scheduler π^* such that $v_1^* = v_2^* = \cdots = v_N^*$.*

A Proof Sketch The proof is based on contradiction. Suppose v_i^*'s were not equal. Then one could always re-allocate the transmission resources (RBs) from the source(s) with lower v_i^*'s to those source(s) with higher v_i^*'s and the objective value $v_{\max}$ will decrease. ∎

Now we give a definition of uniformly fair schedulers.

Definition 1 *A scheduler π is uniformly fair if $v_1 = v_2 = \cdots = v_N$.*

Lemma 1 says there exists an offline optimal scheduler π^* that is uniformly fair. Although an offline optimal scheduler cannot be designed without knowledge of future channel information, we will exploit this uniformly fair property as a guideline when we design Aequitas.

B. Main Ideas

In this section we outline the main ideas of Aequitas. Aequitas is a priority-based scheduler. Within a TTI, for each iteration, Aequitas computes the priorities for all eligible source nodes and selects the source node with the highest priority for resource (RB) allocation.

The most important question to address is how to define and calculate priority in scheduling. Aequitas addresses this question with the following considerations:

- First, Aequitas aims to achieve uniform fairness, i.e., $v_1 = v_2 = \cdots = v_N$ as t increases. Therefore, the source nodes with higher outdated proportion in the past will have higher priorities.
- Recall that $A_i(t)$ will keep increasing until a sample is received in its entirety. So Aequitas will try to make this duration (i.e., $\tau_i(n)$) as small as possible. Therefore, the source node with an unfinished sample carried from the previous TTI will have the highest priority.
- For a source i that already has $A_i(t) \geq d_i$ (i.e., either already outdated or about to be outdated), it should be assigned a higher priority.
- When transmitting a sample from a source, Aequitas tries to use a high MCS to obtain a high date rate per RB. Therefore, the source node that can use a higher MCS (based on its channel quality) will have a higher priority.

C. Design Details

Within a TTI, for each iteration, Aequitas computes a priority metric, denoted by w_i, for each eligible source node

i. Then it selects the source node with the largest w_i and then chooses an MCS and allocates RBs to it. After each iteration, the remaining available RBs will be fewer and Aequitas recomputes w_i's for all remaining eligible source nodes for the next iteration. To design w_i, we need to introduce some notations. When there is no ambiguity, we omit to include "t" in these notations in the rest of this section.

Notations Denote u_i as a binary indicator for whether or not a sample from source node $i = 1, 2, \cdots, N$ is being transmitted in the TTI t (1 for yes and 0 for no). We have

$$u_i = [\big(\sum_{b=1}^{B} x_i^b(t)\big) > 0]. \tag{17}$$

Recall that "$[\cdot]$" is Iverson bracket, returning 1 if the inside statement is true and 0 otherwise. Denote g^b ($b = 1, 2, \cdots B$) as a binary indicator for whether RB b is allocated to some source node (1 for yes, 0 for no). We have

$$g^b = [\big(\sum_{i=1}^{N} x_i^b(t)\big) > 0]. \tag{18}$$

Before the first iteration of Aequitas, we have $x_i^b(t) = 0$ for all i's and b's, so we have $u_i = 0$ and $g^b = 0$ for all i's and b's initially.

Denote s_i as a binary indicator (1 for yes, 0 for no) for whether source i has a sample that started its transmission in previous TTI but still has an unfinished part in the current TTI t. Denote L^{R} as the number of bits in the remaining unfinished part for the source with $s_i = 1$. Denote l_i^m as the minimum required number of RBs to transmit a sample from source node i under MCS m. We have:

$$l_i^m = \begin{cases} \lceil \frac{L_i}{c^m} \rceil & \text{if} \quad s_i = 0, \\ \lceil \frac{L^{\mathrm{R}}}{c^m} \rceil & \text{if} \quad s_i = 1. \end{cases} \tag{19}$$

where "$\lceil \cdot \rceil$" denotes the ceiling function. Denote n_i^m as the number of un-allocated RBs with $q_i^b(t) \geq m$. Then

$$n_i^m = \sum_{b=1}^{B} \big((1 - g^b) \cdot [q_i^b(t) \geq m]\big). \tag{20}$$

Clearly, if $n_i^m \geq l_i^m$, then a sample from source i can be transmitted in full under MCS m.

Denote z_i as a binary indicator for whether a sample from source i can be transmitted under some MCSs in full in this TTI (1 for yes, 0 for no). Clearly, if there is at least one m making $n_i^m \geq l_i^m$, then $z_i = 1$. We have

$$z_i = \big[(\sum_{m=1}^{M} [n_i^m \geq l_i^m]) > 0\big]. \tag{21}$$

In Aequitas, source nodes with $z_i = 1$ have a higher priority than those source nodes with $z_i = 0$.

When a source node is chosen for transmission, we need to select an MCS for it. Denote m_i^* as the optimal MCS level that source i should select.

1) If $z_i = 1$, Aequitas will selects an MCS as high as possible (as long as it can be transmitted in full). That is,

$$m_i^* = \max_m \{m : n_i^m \geq l_i^m\}, \text{ if } z_i = 1. \tag{22}$$

2) If $z_i = 0$, Aequitas will select an MCS that can transmit the maximum amount of information (in bits) with the remaining RBs. That is,

$$m_i^* = \arg\max_m \{c^m \cdot n_i^m\}, \text{ if } z_i = 0. \tag{23}$$

Denote λ_i as the amount of information that source i can transmit under MCS m_i^*, i.e.,

$$\lambda_i = c^{m_i^*} \cdot n_i^{m_i^*}. \tag{24}$$

Clearly, we have $\lambda_i < L_i$ if $z_i = 0$.

Priority Metric Now we are ready to present the priority metric w_i. Based on the design ideas discussed in the last section, w_i depends on s_i, $A_i(t)$, d_i, m_i^*, λ_i, and v_i. We propose the following construct for w_i:

$$w_i = f_1(s_i) \cdot f_2\big(A_i(t), d_i\big) \cdot f_3(z_i, m_i^*) \cdot e_i(t). \tag{25}$$

Some discussions are in order.

- $f_1(s_i)$: This component is about s_i, which indicates whether source node i has an unfinished sample from the previous TTI. In our design, the source node with $s_i = 1$ will have the highest priority. So we define:

$$f_1(s_i) = (1 - s_i) \cdot C_1 + 1 \ , \tag{26}$$

  where C_1 is a constant and $C_1 \gg 1$.
- $f_2\big(A_i(t), d_i\big)$: This component is concerned with the current AoI $A_i(t)$ and the AoI threshold d_i. Those source nodes with $A_i(t) \geq d_i$ should have a greater $f_2\big(A_i(t), d_i\big)$ than those source nodes with $A_i(t) < d_i$. When $A_i(t) < d_i$, the closer the $A_i(t)$ is to its d_i, the higher the $f_2\big(A_i(t), d_i\big)$ should be. When $A_i(t) \geq d_i$, from the objective's perspective, it doesn't matter how much $A_i(t)$ is greater than d_i. So we define:

$$f_2\big(A_i(t), d_i\big) = \begin{cases} \frac{A_i(t)}{d_i} & \text{if} \quad A_i(t) < d_i, \\ C_2 & \text{if} \quad A_i(t) \geq d_i, \end{cases} \tag{27}$$

  where C_2 is a constant and $C_2 > 1$.
- $f_3(z_i, m_i^*)$: This component is about z_i, whether a sample from source i can be transmitted in full, and its MCS level m_i^*. Since we prefer those samples that can be transmitted in full, we will make those sources with $z_i = 1$ have a higher priority than those sources with $z_i = 0$. Among the source nodes with $z_i = 1$, we prioritize those with higher date rates per RB, i.e., $c^{m_i^*}$. Among the nodes with $z_i = 0$, we prioritize those with a larger proportion that can be transmitted in this TTI, i.e., λ_i / L_i. So we define:

$$f_3(z_i, m_i^*) = \begin{cases} c^{m_i^*} & \text{if} \quad z_i = 1, \\ \frac{c^1}{C_3} \cdot \frac{\lambda_i}{L_i} & \text{if} \quad z_i = 0, \end{cases} \tag{28}$$

  where C_3 is constant and $C_3 > 1$.

Algorithm 2 Aequitas

```
Input: $A_i(t)$, $d_i$, $e_i(t)$ and $s_i$ for all $i$'s; $q_i^b(t)$ for all $i$'s and $b's$. Parameters: $C_1$, $C_2$, $C_3$, and $\gamma$.
Output: $x_i^b(t)$'s and $y_i^m(t)$'s.
1: Set $x_i^b(t) = 0$, $y_i^m(t) = 0$, $u_i = 0$ and $g^b = 0$ for all $i$'s, $b$'s and $m$'s.
2: For all $i$'s with $u_i = 0$, compute $z_i$ by (21), $m_i^*$ by (22) or (23), and $w_i$ by (25).
3: Set $k = \arg\max_i w_i$.
4: Set $u_k = 1$, $m = m_k^*$, and $y_k^m(t) = 1$.
5: if $z_k = 1$ then
6:     Choose any $n_k^m$ different $b$'s with $g^b = 0$ and $q_k^b(t) \geq m$, and set $x_k^b(t) = 1$ and $g^b = 1$ for these $b$'s.
7:     go to line 2
8: else
9:     Set $x_k^b(t) = 1$ for all $b$'s with $g^b = 0$.
10: end if
11: return $x_i^b(t)$'s and $y_i^m(t)$'s.
```

- $e_i(t)$: This component serves as an equalizer to ensure $v_1 = v_2 = \cdots = v_N$ as t increases. Denote $\bar{v}_i(t)$ as the outdated proportion for source i till TTI t, i.e.,
$$\bar{v}_i(t) = \frac{1}{t}\sum_{\tau=1}^{t}[A_i(\tau) > d_i]. \tag{29}$$
Clearly, $v_i = \lim_{t\to\infty} \bar{v}_i(t)$. We define $e_i(t)$ as:
$$e_i(t) = \Big(1-\gamma+\gamma\cdot\frac{N\bar{v}_i(t-1)}{\sum_{j=1}^{N}\bar{v}_j(t-1)}\Big)\cdot e_i(t-1), \quad \text{for } t \geq 2, \tag{30}$$
and $e_i(1) = 1$. Here γ is a constant and $0 < \gamma \ll 1$. We can see when $\bar{v}_i(t-1)$ is higher than the average outdated proportion of other sources, it will have a high priority $e_i(t)$ in the next TTI, and vice versa.

For parameter settings, since we want to ensure the unfinished sample from the previous TTI will have the highest priority, the component $f_1(s_i)$ should dominate over other component when $s_i = 1$. To ensure this is the case, we should let $C_1 \gg C_2$ and $C_1 \gg C_3$. After these two conditions are satisfied, the settings of C_2 and C_3 are rather open and do not affect the performance of Aequitas very much.

The complete Aequitas algorithm is summarized (in pseudocode) in Algorithm 2. It can be shown that the time complexity of Algorithm 2 in each TTI is $O(B^2NM)$.

Real-Time Implementation Although Aequitas' time complexity is polynomial, as we will see in Section V, its average running time cannot meet 5G's timing requirement. We propose to exploit parallel computation to speed up Aequitas' execution time. Recall that the computation of NM different n_i^m's (when computing w_i's in line 2, Algorithm 2) is the most time-costing task in Aequitas. We observe that computations of these NM different n_i^m's are independent from each other. In our implementation, we employ a commercial off-the-shelf (COTS) NVIDIA Tesla V100 GPU and CUDA programming platform to computer NM different n_i^m's in parallel.

Table I: Parameters for different types of source nodes.

Type	d_i (TTIs)	L_i (bits)	Type	d_i (TTIs)	L_i (bits)
1	3	6400	6	5	9800
2	5	5400	7	9	4500
3	4	7800	8	4	3900
4	3	4200	9	6	6600
5	6	5600	10	3	5800

V. Performance Evaluation

In this section we conduct experiments and evaluate the performance of our Aequitas implementation.

A. Experiment Setup

We implement Aequitas on an NVIDIA DGX Station with an Intel Xeon E5-2698 v4 CPU (2.20 GHz) and an NVIDIA Tesla V100 GPU (32 GB memory). We use Visual Studio 2019 and CUDA 10.2 for programming.

We consider a network with $B = 50{\sim}100$ RBs and $N = 50{\sim}100$ source nodes. We assume 10 different types of source nodes, each with the same AoI threshold d_i and sample size L_i. We assume small sample size, i.e., $L_i < 10$ Kbits, and a stringent AoI threshold, i.e., $d_i < 10$ TTIs. In this paper, to help readers reproduce the same results, we list a group of random-generated parameters in Table I for each type.

For each source node, we assume a Rician fading channel with factor $K = 1$ and randomly generate an average SNR between 10 and 20dB. We assume there is no correlation in time and frequency, i.e., for each TTI, we generate channel conditions independently and so as for each RB.

In Algorithm 1, we set $\alpha = 10$, $\beta = 0.01$, and $T = 1,000$. In Algorithm 2, we set $C_1 = 100,000$, $C_2 = 10$, $C_3 = 10$, $\gamma = 0.001$, and run the algorithm for 100,000 TTIs.

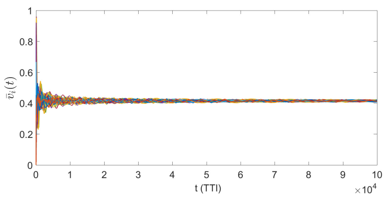
B. A Case Study

We first consider a network with $N = 100$ source nodes (10 nodes for each type in Table I) and $B = 100$ RBs. Define
$$\bar{v}_{\max}(t) = \max\{\bar{v}_1(t), \bar{v}_1(t), \cdots, \bar{v}_N(t)\}. \tag{31}$$
Clearly, we have $\lim_{t\to\infty} \bar{v}_{\max}(t) = v_{\max}$.

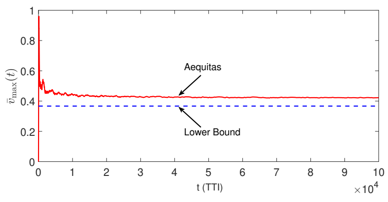
Fig. 3a shows the evolution of $\bar{v}_i(t)$ for all 100 nodes under Aequitas over 100,000 TTIs. As we can see, after a warm-up period, all 100 $\bar{v}_i(t)$'s converge roughly to the same value. This shows that the uniform fairness property for an offline optimal scheduler is also achieved by Aequitas.

Fig. 3b shows the evolution of $\bar{v}_{\max}(t)$ under Aequitas over 100,000 TTIs. The lower bound found by Algorithm 1 is also shown in the figure. We can see that $\bar{v}_{\max}(t)$ by Aequitas is close to the lower bound. When $t = 100,000$, we have $\bar{v}_{\max}(100,000) = 0.422$ while the lower bound is 0.367.

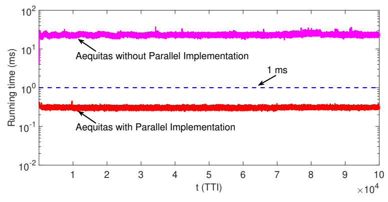
Fig. 3c shows the running time of Aequitas with and without parallel implementation over 100,000 TTIs. Under parallel implementation, the average running time of Aequitas is 0.31 ms, which is under 1 ms (as required by 5G NR); while the average running time is 23.48 ms when parallel implementation is not used.



(a) Evolution of $\bar{v}_i(t)$ for all 100 nodes.

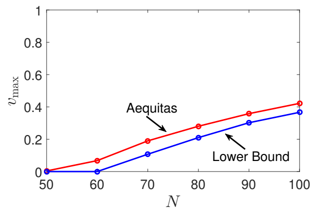


(b) Evolution of $\bar{v}_{\max}(t)$.

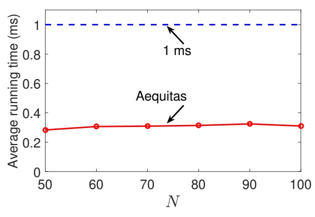


(c) Running time of Aequitas with and without parallel implementation

Figure 3: Results for a case study with $N = 100$, $B = 100$.



(a) $v_{\max}$



(b) Average running time

Figure 4: Aequitas under varying N when $B = 100$

C. Varying Numbers of Source Nodes

In this section, we investigate the performance of Aequitas under varying N. We set $B = 100$ RBs.

Fig. 4a shows the objective $v_{\max}$ under Aequitas for $N = 50, 60, 70, 80, 90$ and 100. Here $v_{\max}$ refers to $\bar{v}_{\max}(100,000)$. The lower bound found by Algorithm 1 is also shown in the figure. As we can see, when N increases $v_{\max}$ under Aequitas also increases, which is intuitive. For all settings of N, $v_{\max}$ under Aequitas is close to the lower bound.

Fig. 4b shows the average running time of Aequitas (with parallel implementation) for varying N's. As we can see, the average running time is always under 1 ms and remains nearly flat when $N = 50$ to 100. Specifically, when $N = 50$ the average running time is 0.283 ms while when $N = 100$ it is 0.310 ms. This is because the number of source nodes will not affect running time much as long as we have enough GPU cores to accommodate them.

VI. Conclusions

In this paper, we investigated online 5G scheduling to minimize outdated proportion of information among a group of source nodes. We first developed a computation procedure to find a lower bound for the objective, which can be used as a performance benchmark. Then we derived a uniform fairness property associated with an offline optimal scheduler, which indicates that each source may have the same proportion of outdated information when time becomes large. Using this property, we developed Aequitas—an online 5G scheduler that performs RB allocation and MCS selection in each TTI. By exploiting the intrinsic parallelism in Aequitas, we implemented it on a GPU platform. Through extensive experimental study, we found that Aequitas can achieve excellent performance in finding objective value while meeting the 5G timing requirement.

Acknowledgments

This research was supported in part by ONR under MURI Grant N00014-19-1-2621, Virginia Commonwealth Cyber Initiative (CCI), and Virginia Tech Institute for Critical Technology and Applied Science.

References

[1] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing Age of Information in Vehicular Networks," in *Proc. IEEE SECON,* 2011.

[2] S. Kaul, R. Yates, and M. Gruteser, "Real-Time Status: How Often Should One Update?" in *Proc. IEEE INFOCOM,* 2012.

[3] Y. Sun, "A Collection of Recent Papers on the Age of Information," available at http://www.auburn.edu/%7eyzs0078 [Online; accessed on 2022-7-15].

[4] R. Talak, S. Karaman, and E. Modiano, "Minimizing Age-of-Information in Multi-Hop Wireless Networks," in *Proc. Allerton Conference,* 2017.

[5] C. Li, S. Li, and Y.T. Hou, "A General Model for Minimizing Age of Information at Network Edge," in *Proc. IEEE INFOCOM,* 2019.

[6] 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 *Proc. IEEE INFOCOM,* 2020.

[7] Q. Liu, C. Li, Y.T. Hou, W. Lou, and S. Kompella, "Aion: A Bandwidth Optimized Scheduler with AoI Guarantee," in *Proc. IEEE INFOCOM,* 2021.

[8] L. Corneo, C. Rohner, and P. Gunningberg, "Age of Information-Aware Scheduling for Timely and Scalable Internet of Things Applications," in *Proc. IEEE INFOCOM,* 2019.

[9] Z. Qian, F. Wu, J. Pan, K. Srinivasan, and N.B. Shroff, "Minimizing Age of Information in Multi-channel Time-sensitive Information Update Systems," in *Proc. IEEE INFOCOM,* 2020.

[10] J. Lou, X. Yuan, S. Kompella, and N. Tzeng, "AoI and Throughput Tradeoffs in Routing-aware Multi-hop Wireless Networks," in *Proc. IEEE INFOCOM,* 2020.

[11] A.M. Bedewy, Y. Sun, R. Singh, and N.B. Shroff, "Optimizing Information Freshness Using Low-Power Status Updates via Sleep-Wake Scheduling," in *Proc. ACM MobiHoc,* 2020.

[12] A. Maatouk, S. Kriouile, M. Assaad, and A. Ephremides, "Asymptotically Optimal Scheduling Policy for Minimizing the Age of Information," in *Proc. IEEE ISIT,* 2020.

[13] C. Li, Q. Liu, S. Li, Y. Chen, Y.T. Hou, and W. Lou, "On Scheduling with AoI Violation Tolerance," in *Proc. IEEE INFOCOM,* 2021.

[14] Z. Jiang, "Analyzing Age of Information in Multiaccess Networks by Fluid Limits," in *Proc. IEEE INFOCOM,* 2021.

[15] X. Chen, X. Liao, and S.S. Bidokhti, "Real-time Sampling and Estimation on Random Access Channels: Age of Information and Beyond," in *Proc. IEEE INFOCOM,* 2021.

[16] 3GPP TS 38.211 version 16.0.0, "NR; Physical channels and modulation," available at https://portal.3gpp.org.

[17] 3GPP TS 38.214 version 16.0.0, "NR; Physical layer procedures for data," available at https://portal.3gpp.org.

[18] A.L. Stolyar, "On the Asymptotic Optimality of the Gradient Scheduling Algorithm for Multiuser Throughput Allocation," *Operations Research,* vol. 53, issue 1, 2005.

[19] IBM ILOG CPLEX Optimizer, available at https://www.ibm.com/analytics/cplex-optimizer [Online; accessed on 2022-7-15].