Minimizing AoI in a 5G-Based IoT Network Under Varying Channel Conditions

Chengzhang Li, *Graduate Student Member, IEEE*, Yan Huang, *Graduate Student Member, IEEE*, Shaoran Li, *Graduate Student Member, IEEE*, Yongce Chen, *Graduate Student Member, IEEE*, Brian A. Jalaian, *Member, IEEE*, Y. Thomas Hou, *Fellow, IEEE*, Wenjing Lou, *Fellow, IEEE*, Jeffrey H. Reed, *Fellow, IEEE*, and Sastry Kompella, *Senior Member, IEEE*

***Abstract*—The Age of Information (AoI) is a key metric to measure the freshness of information for IoT applications. Most of the existing analytical models for AoI are overly idealistic and do not capture state-of-the-art transmission technologies such as 5G as well as channel dynamics in both frequency and time domains. In this article, we present Kronos, a real-time 5G-compliant scheduler that minimizes AoI for IoT data collection. Kronos is designed to cope with highly dynamic channel conditions. Its main function is to perform RB allocation and to select the modulation and coding scheme for each source node based on channel conditions, with the objective of minimizing long-term AoI. To meet the stringent real-time requirement for 5G, we develop a GPU-based implementation of Kronos on commercial off-the-shelf Nvidia GPUs. Through extensive experimentation, we show that Kronos can find near-optimal solutions under sub-millisecond time scale. To the best of our knowledge, this is the first real-time AoI scheduler that is 5G compliant.**

***Index Terms*—5G, Age of Information (AoI), IoT, scheduling, varying channel conditions.**

I. Introduction

With the proliferation of IoT and its capability of massive information gathering and sharing through edge/cloud computing, users are no longer satisfied with merely obtaining the information they desire, but rather, care more about how fresh the information is when it is consumed. To address this need, the concept of "Age of Information" (AoI) was conceived in [1] and [2] and has since gained acceptance in the research community. AoI is associated with the latest (freshest) sample at a particular location (e.g., source, edge, or cloud) and is define as the elapsed time since its "birth" (generation). In contrast to network delay (either end-to-end or per link), AoI is an application-layer performance metric.

There has been active research on designing scheduling algorithms to minimize AoI (see [3]). However, existing research on AoI has been largely limited to information-theoretic exploration. Most notably, few existing efforts on AoI modeling and analysis have considered the characteristics or capabilities of state-of-the-art transmission technologies such as 5G NR [4]. A bulk of existing research has been predicated on extremely simple toy models (see related work Section II), which can hardly be applied to real-world systems. Furthermore, there has been limited research on AoI scheduling that addresses the impact of varying channel conditions, such as joint dynamics in both time and frequency. But in reality, channel condition can change rapidly (e.g., for each transmission time interval (TTI) in 5G) and must be taken into consideration in real-time scheduling.

In this article, we focus on the design of the 5G-compliant AoI scheduler and address the impact of both time and frequency-varying channel conditions. Our edge base station (BS) for an IoT network is designed to conform with state-of-the-art 5G cellular standard [4], which is supported by major carriers (e.g., AT&T [5] and Verizon [6]) for IoT deployment. Furthermore, our scheduler is designed to cope with highly varying channel conditions (e.g., time-selective fading and frequency-selective fading), which is a major challenge in the real-world environment. Finally, we want to ensure that our AoI scheduler can strictly meet the stringent timing requirement (i.e., submillisecond running time for computing scheduling solution) as specified in the 5G standard.

There are a number of technical challenges in this research. First, in a 5G setting, the AoI scheduling problem entails the allocation of resource blocks (RBs) and selection of modulation and coding scheme (MCS) for each source node in each TTI based on the channel conditions. This presents a much larger search space for an optimal solution than any of those problems considered to date in the AoI research literature. Second, the stringent timing requirement for real-world 5G systems (i.e., submillisecond time scale) sets a hard

Manuscript received July 24, 2020; revised November 5, 2020; accepted January 10, 2021. Date of publication January 25, 2021; date of current version September 23, 2021. This work was supported in part by the Office of Naval Research under MURI Grant N00014-19-1-2621; in part by the Virginia Commonwealth Cyber Initiative (CCI); in part by NVIDIA AI Lab (NVAIL) in Santa Clara, CA, USA, for its unrestricted gift and equipment donation. This article was presented in part at the IEEE ICDCS, Dallas, TX, USA, July 6–9, 2019. *(Corresponding author: Y. Thomas Hou.)*

Chengzhang Li, Shaoran Li, Yongce Chen, Y. Thomas Hou, Wenjing Lou, and Jeffrey H. Reed are with the Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA (e-mail: hou@vt.edu).

Yan Huang is with NVIDIA Corp., Santa Clara, CA 95051 USA.

Brian A. Jalaian is with Army Research Laboratory, Adelphi, MD, USA

Sastry Kompella is with the U.S. Naval Research Laboratory, Washington, DC 20375 , USA.

Digital Object Identifier 10.1109/JIOT.2021.3053914

performance benchmark against any design of AoI schedulers. As we shall see, it is extremely challenging to find a near-optimal solution for a problem of such size and complexity in a submillisecond time scale.

This article addresses the above technical challenges with the following contributions.

1) This article studies AoI scheduling in a 5G setting with considerations of varying channel conditions. Specifically, the uplink transmission resource is divided as grids of RBs that span both time and frequency domains with different channel conditions. The scheduling problem under 5G entails the RB allocation to each source node and the selection of MCS for each source node based on the channel condition on each RB, with the goal of minimizing long-term average AoI.
2) Since the channel condition for the future is unknown, we purse to design an online AoI scheduling algorithm. For the performance benchmark, we propose a novel computational procedure to find an asymptotic lower bound for the objective value. Specifically, we first relax the original AoI minimization problem to a data rate minimization problem. Then, we employ a gradient scheduling algorithm to find an asymptotic lower bound for the latter problem. The gradient scheduling minimizes an empirical data rate for each TTI, which can be formulated into an integer quadratic programming (IQP) problem and solved by the CPLEX solver.
3) For our AoI scheduling problem, we present Kronos, an online algorithm that conforms to 5G transmission standard and can cope with varying channel conditions. The essence of Kronos is to iteratively select a source node for RB allocation until all RBs in a TTI are allocated. We propose a novel metric that takes into consideration of AoI outage and the channel conditions for the source node. Based on this metric, we can identify the next source node for RB allocation and determine its MCS.
4) To ensure that Kronos can meet the stringent timing requirement in 5G, we propose to employ commercial off-the-shelf (COTS) GPUs to speed up Kronos' running time through parallel computation. Specifically, we exploit the massive number of GPU processing cores to compute and compare the scheduling metric for all possible combinations of source nodes and MCSs. For proof of concept, we implement Kronos on an Nvidia Tesla V100 GPU using the CUDA programming model. Through extensive performance evaluation, we find that Kronos can achieve near-optimal performance (when compared to our lower bound) in a submillisecond time scale.

The remainder of this article is organized as follows. In Section II, we review the related work on AoI scheduling. In Section III, we describe a 5G-based IoT network that we focus in this article. In Section IV, we formally state the 5G AoI scheduling problem and introduce its real-time requirement. In Section V, we present a novel computational procedure to find an asymptotic lower bound for the objective function. In Section VI, we present Kronos, a 5G-compliant real-time scheduler that minimizes AoI. In Section VII, we perform extensive experiments to study the performance of Kronos. Section VIII concludes this article.

II. Related Work

There has been a flourish of research on AoI in recent years [3]. The main line of research is to design schedulers that minimize AoI (e.g., [7]–[19]). Another line of research focuses on the modeling, analysis, and optimization of AoI (e.g., [20]–[26]). There are also some other branches on AoI research, e.g., the game theory for AoI (e.g., [27]–[29]), channel coding for AoI (e.g., [30]–[32]), and AoI applications (e.g., [33], [34]), to name a few. Since this article is on scheduler design to minimize AoI, we will limit our literature review in this area.

The authors in [7]–[10] considered the same transmission model in Fig. 1, where information sources share a common channel. A simplifying assumption in these works is that only one packet from at most one source can be transmitted to the BS in one time slot. Specifically, Hsu *et al.* [7] assumed a Bernoulli packet arrival model for the sources. Kadota *et al.* [8] considered an unreliable channel where there is a fixed packet loss probability for each transmission. Zhong *et al.* [9] studied a new metric called Age of Synchronization (AoS) along with AoI. Li *et al.* [10] studied scheduling under a given AoI deadline.

Extensions beyond the simple transmission model in [7]–[10] have also been explored in recent works. For example, Bedewy *et al.* [11] and Sun *et al.* [12] extended the one packet by only one source model to a multichannel system, where multiple sources can send packets to the BS through multiple independent channels. Li *et al.* [13] explored a more general problem with considerations of sampling periods, sample size, and link capacity. The authors in [14]–[17] considered a multilink *ad hoc* network where multiple information sources send information updates to multiple destinations. Finally, Bedewy *et al.* [18] and Talak *et al.* [19] considered a multihop network environment, where the sources update information to their destinations through relay nodes. Although these efforts have made important contribution to the theoretical foundation on AoI scheduling research, they fall short on addressing AoI scheduling problems in the real-world system, such as 5G.

There have been some works on exploring the impact of time-varying channel conditions on AoI. The authors in [8], [14]– [16] considered a 2-state time-varying channel model, where each packet can be either received successfully or lost with a fixed probability depending on the channel condition. Lu *et al.* [17] assumed that the channel coherence time could last an entire frame (consisting of a large number of time slots) and the maximum number of packets that can be delivered in a frame is time-varying (frame-varying). Tang *et al.* [25], [26] considered a multistate time-varying channel and the link capacity is modeled with multiple levels on each time slot. Although these efforts have explored different models for time-varying channel conditions, they are still considered overly simplified. For example, none of them has considered underlying physical-layer transmission technologies such as 5G, where channel-dependent 2-D frequency-time RBs are used for resource scheduling.

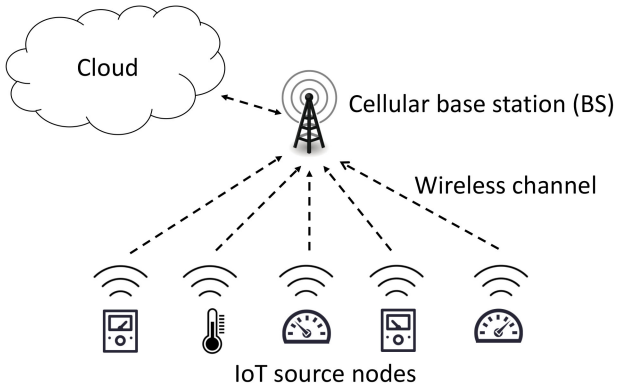


Fig. 1. System model: a set of source nodes collects information and sends it to a BS.

III. 5G-Based IoT Architecture

Consider a 5G-based IoT network, where a set $\mathcal{N}$ of source nodes collect information and forward it to a BS (see Fig. 1). Each source node periodically takes a sample of information from its environment. Denote T_i as the sampling period (in unit of time slots) at source node *i*. Due to the heterogeneity of IoT applications, the sampling periods are generally different among different source nodes. 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. Again, due to the heterogeneity of IoT devices, sample sizes are generally different among different source nodes.

Once a sample is produced at a source node *i*, it is stored in a local memory, which is of finite size. Due to the limited channel bandwidth, not every sample from each source node will be transmitted to the BS. When a source node starts a new transmission, it always selects the freshest sample (i.e., the most recently generated sample) for transmission. This is to ensure the cellular BS can receive the freshest information. As a result, only a fraction of samples generated at each source node will be transmitted while the rest will be eventually discarded at the source nodes (due to limited node memory). Once a source starts to transmit a sample, it may take multiple consecutive time slots (if necessary) to complete the transmission. In this case, any newly generated sample afterward cannot preempt an ongoing transmission of an older sample.

Since the uplink transmission from IoT source nodes to the BS conforms to the 5G standard [4], transmission resource is organized into grids of RBs that span both time and frequency domains. In the time domain, time is equally slotted into TTIs, while in the frequency domain, the bandwidth is equally slotted among a large number of subcarriers, with 12 subcarriers over a TTI being called an RB. That is, for each TTI, there is a large number of RBs that can be allocated to the source nodes for uplink transmission.

Due to varying channel conditions in time (across different TTIs, i.e., time-selective fading) and frequency (across different RBs, i.e., frequency-selective fading), channel feedback from each source node is necessary for the optimal scheduling of RBs at the BS. Given such channel variation over time and frequency, it is desirable to perform scheduling for each TTI, the smallest time resolution for 5G transmission.

At the BS, the collected information can be either processed and stored locally (edge computing) and/or be forwarded to a cloud, where the information can be further processed and accessed broadly by users from any location. Since many time-sensitive IoT applications need to access the latest sampled information from each source, it is desirable to maintain the freshest sample (from each source) at the edge BS. So, it is necessary to design a specialized scheduler to minimize AoI for the maintained samples at the BS. This is the problem we are going to study in this article.

TABLE I
Notation

Symbol	Definition
$A_i^{\mathrm{s}}(t)$	AoI of the most recent sample at source node i at TTI t
$A_i^{\mathrm{B}}(t)$	AoI of the sample from source node i at the BS at TTI t
$\bar{A}_i^{\mathrm{B}}$	Long term average of $A_i^{\mathrm{B}}(t)$
$\bar{A}^{\mathrm{B}}$	Weighted sum of all $\bar{A}_i^{\mathrm{B}}$'s
$\mathcal{B}$	A set of RBs available for scheduling at the 5G BS
c^m	Modulation and coding rate under MCS level m
$r_i^{b,m}(t)$	Achievable data rate by RB b w.r.t. source node i under MCS m at TTI t
L_i	Sample size at source node i
$\mathcal{M}$	A set of MCS levels that a source node can use
$\mathcal{N}$	A set of source nodes in the 5G-based IoT network
$q_i^b(t)$	The highest MCS level that source node i's channel can support on RB b at TTI t
$R_i(t)$	Amount of information transmitted by source node i in TTI t across all RBs allocated to it
T_i	Sampling period at source node i (in unit of TTIs)
$U_i^{\mathrm{B}}(t)$	Generation time of the most recent sample from source node i at the BS at TTI t
$U_i^{\mathrm{s}}(t)$	Generation time of the most recent sample at source node i at TTI t
w_i	An assigned weight for source node i at the BS

TABLE II
Abbreviations

Abbreviation	Meaning
AoI	Age of Information
AUS	Almost Uniform Scheduler
BS	Base Station
BTT	Begin-To-Transmit
COTS	Commercial Off-The-Shelf
IQP	Integer Quadratic Programming
MCS	Modulation and Coding Scheme
RB	Resource Block
TTI	Transmission Time Interval

IV. Modeling and Problem Statement

Table I lists key notations in this article and Table II lists the abbreviations. When there is no ambiguity, we use the term "TTI" and "time slot" interchangeably throughout this article. Also, wherever appropriate, we use the term "at TTI *t*" to refer to "at the beginning of TTI *t*" and use the term "in TTI *t*" to refer to the underlying action is completed "at the end of TTI *t*".

A. AoI Notation

Recall that for each source node i, it takes a sample every T_i time slots. Denote $U_i^s(t)$ as the generation time of the most recent sample at source node i. Clearly, $U_i^s(t) \leq t$. Then, the AoI of the most recent sample at source node i at TT t, denoted as $A_i^s(t)$, is defined as

$$A_i^s(t) = t - U_i^s(t). \tag{1}$$

Since sampling at source node i has a period T_i, function $A_i^s(t)$ exhibits a zigzag shape with slope 1 and period T_i. Clearly, we have $0 \leq A_i^s(t) \leq T_i - 1$.

On the other hand, the sample maintained at the BS may be older than the sample just produced at the source node. Denote $U_i^B(t)$ as the generation time of the most recently received sample from source node i at the BS. Then, the AoI for source i at the BS at time slot t, denoted as $A_i^B(t)$, is defined as

$$A_i^B(t) = t - U_i^B(t). \tag{2}$$

Function $A_i^B(t)$ also exhibits a zigzag shape with slope 1, and we have $A_i^B(t) \geq A_i^s(t)$.

It is instructive to make a connection between $A_i^B(t)$ (AoI at edge BS) and $A_i^s(t)$ (AoI at a source node). At source node i, for the kth sample that is successfully transmitted to the BS (excluding those that are not transmitted), denote its beginning transmission TTI as $b_i(k)$ and ending transmission TTI as $e_i(k)$. By the definition of $U_i^s(t)$, the generation time of this kth sample is $U_i^s(b_i(k))$. After the last unit of data of this sample is completely sent to the BS in TTI $e_i(k)$, in the next TTI $(e_i(k)+1)$, $U_i^B(t)$ is updated and we have $U_i^B(e_i(k)+1) = U_i^s(b_i(k))$. By the definitions of $A_i^B(t)$ and $A_i^s(t)$, it can be shown that AoI evolution at the BS follows the following expression:

$$A_i^B(t+1) = \begin{cases} A_i^s(b_i(k)) + e_i(k) - b_i(k) + 1, & \text{if } t = e_i(k) \\ A_i^B(t) + 1, & \text{otherwise.} \end{cases} \tag{3}$$

The long-term average of $A_i^B(t)$ for source node i at the BS is defined as

$$\bar{A}_i^B = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} A_i^B(t). \tag{4}$$

The BS assigns a weight for each source i, which is denoted by w_i. Then, the weighted sum of the long-term average of $A_i^B(t)$ over all source nodes $i \in \mathcal{N}$, denoted as $\bar{A}^B$, is

$$\bar{A}^B = \sum_{i\in\mathcal{N}} w_i \cdot \bar{A}_i^B. \tag{5}$$

Our objective is to minimize $\bar{A}^B$.

B. Uplink Transmission

As shown in Fig. 1, the set of IoT source nodes (users) $\mathcal{N}$ shares an uplink channel to transmit to the BS. Denote $\mathcal{B}$ as the set of RBs in one TTI for uplink transmission. The scheduler at the BS must allocate this set $\mathcal{B}$ of RBs to a subset of source nodes at each TTI to minimize $\bar{A}^B$.

Denote $x_i^b(t)$ as a binary variable indicating whether RB $b \in \mathcal{B}$ is allocated to source node i at TTI t, i.e.,

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

We assume that each RB can only be allocated to at most one source node.[1] We have

$$\sum_{i\in\mathcal{N}} x_i^b(t) \leq 1 \quad (b \in \mathcal{B}). \tag{6}$$

Besides RB allocation, for each TTI, the scheduler also needs to choose an MCS for each source node [4]. The MCS of each source node determines the modulation and coding rate—how much information (in unit of bits) is modulated and coded in each RB for this source node. The higher the MCS, the higher the modulation and coding rate. On the other hand, the maximum amount of information that can be successfully transmitted by an RB also depends on the channel condition. If the channel condition for this RB is poor and the source uses a high MCS, information carried in the RB cannot be successfully received and decoded by BS. Therefore, the achievable data rate by an RB $b \in \mathcal{B}$ depends on both the MCS used by the source node as well as the channel condition for this RB.

Under 5G, there are 29 levels of MCSs for transmission [4]. Denote $\mathcal{M}$ as the set of these available MCSs (i.e., $\mathcal{M} = \{1, 2, \ldots, 29\}$), where we have $m = 1$ corresponding to the lowest MCS and $m = 29$ corresponding to 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

$$0 \leq q_i^b(t) \leq |\mathcal{M}|.$$

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 [4, Table 5.1.3.1-1]. Denote $r_i^{b,m}(t)$ as the achievable data rate by RB b with respect to source node i under MCS m. If $m \leq q_i^b(t)$, 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

$$r_i^{b,m}(t) = \begin{cases} c^m, & \text{if } m \leq q_i^b(t) \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

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 $m \in \mathcal{M}$ for all its allocated RBs [4]. Denote $y_i^m(t)$ as a binary variable indicating whether MCS $m \in \mathcal{M}$ 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

$$\sum_{m\in\mathcal{M}} y_i^m(t) \leq 1 \quad (i \in \mathcal{N}). \tag{8}$$

[1]The case of multiuser MIMO where an RB can be allocated to multiple sources is much more complex and will be explored in future work.

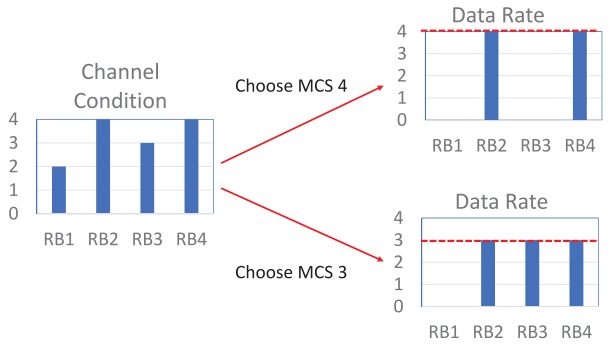


Fig. 2. Example illustrating the tradeoff between MCS selection and the number of RBs that can contribute to the total data rate.

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

$$R_i(t) = \sum_{b\in\mathcal{B}} \sum_{m\in\mathcal{M}} x_i^b(t) y_i^m(t) r_i^{b,m}(t) \quad (i \in \mathcal{N}). \tag{9}$$

Based on (7) and (8), there is a clear tradeoff between the choice of m and the number of RBs allocated to source node i that can contribute to $R_i(t)$, due to the differences in channel conditions on each RB allocated to the same source node. That is, the higher the MCS m is chosen, the fewer the number of RBs can help contribute to $R_i(t)$. In Fig. 2, we use an example to illustrate this tradeoff. For a source node, suppose it is allocated with four RBs. Suppose under the current channel conditions on the four RBs, the maximum allowed MCS on the four RBs are 2, 4, 3, and 4, respectively. If we choose MCS 4 for transmission, then only RBs 2 and 4 can contribute to $R_i(t)$, each with an MCS level 4. If we choose MCS 3 for transmission, then RBs 2–4 can contribute to $R_i(t)$, with each contributing to $R_i(t)$ with MCS level 3. Clearly, choosing a higher MCS level may not translate to the total aggregate (effective) data rate among the RBs. From this example, we can see that judicious choice of MCS is necessary to balance the achievable bit rates from each RB and the number of RBs that can actually contribute to achievable bit rates.

C. Problem Statement and Technical Challenges

In this article, we want to design a real-time 5G-compliant scheduler to minimize $\bar{A}^B$ at the BS. The scheduling algorithms entails to allocate $|\mathcal{B}|$ RBs to $|\mathcal{N}|$ source nodes in each TTI, and to choose an MCS for each user. That is, to determine the decision variables $x_i^v(t)$ and $y_i^m(t)$ for each TTI t so that $\bar{A}^B$ is minimized.

There are a number of challenges associated with this problem. First, the search space of the scheduling problem is enormous. Within each TTI, the BS needs to allocate $|\mathcal{B}|$ RBs (e.g., 100) among $|\mathcal{N}|$ source nodes (e.g., 100), and assign each source node an optimal MCS (among 29 possible levels). The solution space consists of $|\mathcal{B}|^{|\mathcal{N}|} \cdot |\mathcal{M}|^{|\mathcal{N}|}$ possibilities. None of the existing AoI research (see [3]) have studied problems of such size and complexity.

Second, the scheduling algorithm that we need should be an online algorithm. This is because the scheduler can only make a scheduling decision for the next TTI based on most recent channel feedback information. It does not have any knowledge of channel conditions for the future. Since we are minimizing a long-term average AoI, it is not possible for a schedule to make an optimal decision without knowledge of the future. Therefore, one can at best design a near-optimal scheduler.

Finally, the timing requirement to run our scheduler is very stringent. Under 5G [4], our scheduler must find its scheduling decision within a TTI, which is in a submillisecond time scale. To date, none of the existing AoI research has considered such a real-time requirement in the design of a scheduling solution.

V. Performance Bound

In this section, we develop a lower bound for the objective $\bar{A}^B$. This lower bound (if tight) can be used as a benchmark to measure the performance of a scheduling algorithm that we will design later in Section VI.

Denote $\overline{R}_i$ as the long-term average data rate for source node $i \in \mathcal{N}$, i.e.,

$$\overline{R}_i = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} R_i(t). \tag{10}$$

In Section V-A, we develop a lower bound for $\bar{A}^B$ (applicable to all scheduling algorithms) assuming that $\overline{R}_i$'s are given *a priori*. In Section V-B, we remove this assumption (i.e., knowledge of $\overline{R}_i$'s) and present a novel computational procedure to find a lower bound for $\bar{A}^B$.

A. Lower Bound for Known $\overline{R}_i$'s

For given $\overline{R}_i$'s, there may exist different scheduling algorithms. Among these different scheduling algorithms, how do we find a scheduling algorithm that offers the minimum $\bar{A}^B$? This is the question we want to answer in this section.

Note that in (5), $\bar{A}^B$ is a weighted sum of all $\bar{A}_i^B$'s. A lower bound of $\bar{A}^B$ under the given $\overline{R}_i$'s can be found by the following relaxation. If we can find a lower bound for each $\bar{A}_i^B$ (for $i \in \mathcal{N}$) under the given $\overline{R}_i$ that is independent of the other $\bar{A}_j^B$'s or $\overline{R}_j$'s (for $j \neq i$), then we can multiple each with its corresponding weight and sum them up. This sum of weighted lower bound is clearly a lower bound of $\bar{A}^B$.

Following this idea, we now show how to find a lower bound for $\bar{A}_i^B$ under a given $\overline{R}_i$ that is independent of other $\overline{R}_j$ (for $j \neq i$). For ease of exposition, denote p_i as the fraction (in percentage) of successfully transmitted samples over all generated samples in the long term. Clearly, $p_i \leq 1$. With p_i, we can rewrite $\overline{R}_i$ as

$$\overline{R}_i = \frac{p_i L_i}{T_i}. \tag{11}$$

Note that $\overline{R}_i$ is proportional to p_i via a constant factor. Therefore, minimizing $\bar{A}_i^B$ under a given $\overline{R}_i$ is equivalent to minimizing $\bar{A}_i^B$ under a given p_i.

Since we want to find a lower bound of $\bar{A}_i^B$ under a given p_i, let us artificially move ahead the update time for $A_i^B(t)$ as follows. Instead of updating $A_i^B(t)$ at the end of TTI $e_i(k)$, let us move the update time to the end of TTI $b_i(k)$. Although this is infeasible in reality, it serves our purpose of finding a lower bound. Clearly, $\bar{A}_i^B$ at the BS under such a fictitious

updating mechanism is smaller than that when the update is made at the end of TTI $e_i(k)$ (since the update is performed earlier than it should be). Therefore, we will use $\bar{A}_i^B$ obtained under such a fictitious update mechanism as a lower bound.

Ideally, to minimize this new lower bound of $\bar{A}_i^B$ under a given p_i, we would like to have each of those samples be transmitted *immediately* after they are generated. Clearly, such a hypothesized (ideal) scheduler would offer a new lower bound for $\bar{A}_i^B$ under a given p_i.

Denote $T_i^{BTT}(k)$ as the "begin-to-transmit" (BTT) interval for the kth and $(k+1)$th sample at source node i. That is, $T_i^{BTT}(k)$ is the interval between the starting (beginning) time of transmitting the kth sample and the starting time of transmitting the $(k+1)$th sample at source node i. Based on this definition, we have

$$T_i^{BTT}(k) = b_i(k+1) - b_i(k). \tag{12}$$

Then, under a hypothesized scheduler, $T_i^{BTT}(k)$ is an integral multiple of the sampling period T_i. Clearly, such a hypothesized scheduler is not unique and many (each with different behavior of $T_i^{BTT}(k)$'s) may offer the same p_i. Among this group of hypothesized schedulers, we want to identify a scheduler that minimizes $\bar{A}_i^B$.

More formally, we define almost uniform scheduler (AUS) as a hypothesized scheduler with

$$T_i^{BTT}(k) = \text{either } h_iT_i \text{ or } (h_i+1)T_i \text{ for } k > 0$$

where h_i is defined as

$$h_i = \left\lfloor \frac{1}{p_i} \right\rfloor \tag{13}$$

and $\lfloor \cdot \rfloor$ is the floor function. Then, we have the following lemma for the scheduler that minimize $\bar{A}_i^B$ among the hypothesized schedulers.

Lemma 1: AUS minimizes $\bar{A}_i^B$ among all hypothesized schedulers with

$$\bar{A}_i^B = \frac{T_i}{2} f(p_i) + \frac{1}{2} \tag{14}$$

where

$$f(p_i) = 2\left\lfloor \frac{1}{p_i} \right\rfloor + 1 - \left(\left\lfloor \frac{1}{p_i} \right\rfloor^2 + \left\lfloor \frac{1}{p_i} \right\rfloor \right) \cdot p_i. \tag{15}$$

Lemma 1 says that the hypothesized scheduler that employs almost uniform BTT intervals (or exactly uniform in the case when $1/p_i$ is an integer) minimizes $\bar{A}_i^B$. This result is very intuitive. It can be shown (as in the proof below) that for any other scheduler with a larger variance in BTT intervals, one can always find a scheduler with a smaller variance in BTT intervals that reduces $\bar{A}_i^B$.

Proof: To prove Lemma 1, we need to prove: 1) AUS is the hypothesized scheduler that minimizes $\bar{A}_i^B$ and 2) under AUS, $\bar{A}_i^B$ is given in (14).

We first prove AUS is the hypothesized scheduler that minimizes $\bar{A}_i^B$. Our proof is based on contradiction. Assume that AUS is not the hypothesized scheduler that minimizes $\bar{A}_i^B$. Then, denote π^* (which is not AUS) as the hypothesized scheduler that minimizes $\bar{A}_i^B$. Recall the length of any BTT interval under a hypothesized scheduler is an integral multiple of T_i. Therefore, we can find two BTT intervals under π^* with length n_1T_i and n_2T_i such that $n_1 \geq n_2 + 2$, with n_1 and n_2 being integers. We can change the lengths of the two BTT intervals to $(n_1-1)T_i$ and $(n_2+1)T_i$, and get a new hypothesized scheduler π_0. Clearly, π_0 and π^* has the same p_i. However, the average AoI within the two BTT intervals in π^* is

$$A^* = \frac{(n_1T_i)^2 + (n_2T_i)^2 + (n_1+n_2)T_i}{2(n_1+n_2)T_i} \tag{16}$$

while the average AoI within the two BTT intervals in π_0 is

$$A_0 = \frac{((n_1-1)T_i)^2 + ((n_2+1)T_i)^2 + (n_1+n_2)T_i}{2(n_1+n_2)T_i}. \tag{17}$$

We have

$$A^* - A_0 = \frac{(n_1 - n_2 - 1)T_i}{n_1+n_2} > 0 \tag{18}$$

which means π_0 has a smaller $\bar{A}_i^B$ than what π^* has. This contradicts to that π^* is the hypothesized scheduler that minimizes $\bar{A}_i^B$.

Now, we have proved that AUS is the hypothesized scheduler that minimizes $\bar{A}_i^B$. We want to find $\bar{A}_i^B$ under AUS. Recall p_i is the fraction of all generated samples that are successfully transmitted. So, the average of all BTT intervals at the BS is T_i/p_i. Clearly, the length of the BTT interval for source node i under AUS is either h_iT_i or $(h_i+1)T_i$. Denote a as the percentage of those BTT intervals with length h_iT_i, then $(1-a)$ is the percentage of those BTT intervals with length $(h_i+1)T_i$. When the time interval $[0, T)$ is large, i.e., $T \to \infty$, the fraction of successfully transmitted samples approaches p_i, and the occurrence rates (the number of occurrences over the number of TTIs) of those two types of BTT intervals are p_ia/T_i and $p_i(1-a)/T_i$, respectively. We have

$$\lim_{T\to\infty} \frac{T \cdot \frac{p_ia}{T_i} \cdot h_iT_i + T \cdot \frac{p_i(1-a)}{T_i} \cdot (h_i+1)T_i}{T} = 1 \tag{19}$$

which gives us

$$h_ia + (h_i+1)(1-a) = \frac{1}{p_i}. \tag{20}$$

We have

$$a = h_i + 1 - \frac{1}{p_i}. \tag{21}$$

Based on a, we can calculate $\bar{A}_i^B$ under AUS

$$\begin{aligned} \bar{A}_i^B &= \frac{a \cdot \frac{h_iT_i(1+h_iT_i)}{2} + (1-a) \cdot \frac{(h_i+1)T_i(1+(h_i+1)T_i)}{2}}{ah_iT_i + (1-a)(h_i+1)T_i} \\ &= \frac{T_i}{2}\left(2h_i + 1 - \left(h_i^2 + h_i\right)p_i\right) + \frac{1}{2} \\ &= \frac{T_i}{2} f(p_i) + \frac{1}{2}. \end{aligned}$$

This completes our proof. ∎

Combining (5), (11), and (14), we have the following lower bound for $\bar{A}^B$ as a function of $\overline{R}_i$:

$$\bar{A}^B \geq \sum_{i\in\mathcal{N}} w_i \cdot \left(\frac{T_i}{2} f\left(\frac{\overline{R}_iT_i}{L_i} \right) + \frac{1}{2} \right). \tag{22}$$

In the next section, we will remove the assumption of the prior knowledge of $\overline{R}_i$.

B. Finding Lower Bound of $\bar{A}^B$

Based on (22), a lower bound for $\bar{A}^B$ can be found by minimizing the RHS of (22), i.e.,

$$\text{OPT-LB: } \min \sum_{i\in\mathcal{N}} w_i\left(\frac{T_i}{2}f\left(\frac{\overline{R}_i T_i}{L_i}\right)+\frac{1}{2}\right)$$
$$\text{s.t. Constraints (6), (7), (8), (9), (10)}$$

where the optimization variables are $x_i^b(t)$'s and $y_i^m(t)$'s. In the rest of this section, we show how to solve OPT-LB. We emphasize that the optimal solution to OPT-LB only serves as a lower bound of $\bar{A}^B$, while its own AoI performance is not of our concern (and is likely much higher than this lower bound).

For the ease of exposition, we define

$$J_i\big(\overline{R}_i\big) = w_i\left(\frac{T_i}{2}f\left(\frac{\overline{R}_i T_i}{L_i}\right)+\frac{1}{2}\right). \tag{23}$$

Then, the objective of OPT-LB becomes

$$\min \quad \sum_{i\in\mathcal{N}} J_i\big(\overline{R}_i\big). \tag{24}$$

We will design an optimal scheduling algorithm to OPT-LB and obtain a lower bound for $\bar{A}^B$.

OPT-LB is a scheduling problem to minimize a function of $\overline{R}_i$. Similar problems have appeared in the information theory community (see [35] and [36]), where it has been shown that a gradient scheduling algorithm can achieve the same optimal objective value asymptotically (when the number of TTIs goes to infinity). Specifically, in a gradient scheduling algorithm, we define an empirical data rate $R_i^e(t)$ for each TTI t and it is updated as a moving average as follows:

$$R_i^e(t+1) = (1-\beta)R_i^e(t) + \beta R_i(t) \tag{25}$$

where β is a small positive constant (e.g., 0.01) and $R_i(t)$ is the instant data rate at TTI t. It can be easily shown that under the moving average updating algorithm in (25), when $\beta \to 0$

$$\lim_{t\to\infty} R_i^e(t) = \overline{R}_i. \tag{26}$$

That is, $R_i^e(t)$ asymptotically approaches $\overline{R}_i$ when $t \to \infty$. In practice, t does not need to be very large to achieve this approximation.

Based on (26), OPT-LB becomes

$$\text{OPT-1: } \min \lim_{t\to\infty} \sum_{i\in\mathcal{N}} J_i\big(R_i^e(t)\big)$$
$$\text{s.t. Constraints (6), (7), (8), (9), (25)}$$

where the optimization variables are $x_i^b(t)$'s and $y_i^m(t)$'s.

The idea of the gradient scheduling algorithm is to minimize $\sum_{i\in\mathcal{N}} J_i(R_i^e(t+1))$ at every t. It has been shown that by performing such a minimization for every TTI, $\lim_{t\to\infty}\sum_{i\in\mathcal{N}} J_i(R_i^e(t))$ is also minimized when $\beta \to 0$ [35], [36].

We now show how to minimize $\sum_{i\in\mathcal{N}} J_i(R_i^e(t+1))$ at TTI t. When $\beta \to 0$, using (25), we have

$$\begin{aligned}\sum_{i\in\mathcal{N}} J_i\big(R_i^e(t+1)\big) &= \sum_{i\in\mathcal{N}} J_i\big((1-\beta)R_i^e(t)+\beta R_i(t)\big)\\ &= \sum_{i\in\mathcal{N}} J_i\big(R_i^e(t)+\beta\big(R_i(t)-R_i^e(t)\big)\big)\\ &= \sum_{i\in\mathcal{N}} J_i\big(R_i^e(t)\big) + \sum_{i\in\mathcal{N}} \frac{dJ_i(R)}{dR}\Big|_{R=R_i^e(t)}\\ &\quad\times \beta\big(R_i(t)-R_i^e(t)\big)\end{aligned} \tag{27}$$

where the last equality follows from the definition of derivative of $J_i(\cdot)$. Since $R_i^e(t)$ can be computed at TTI t, $J(R_i^e(t))$ and $[(dJ_i(R))/dR]|_{R=R_i^e(t)}R_i^e(t)$ can also be computed. Therefore, to minimize $\sum_{i\in\mathcal{N}} J_i(R_i^e(t+1))$ at TTI t, we only need to solve the following problem:

$$\text{OPT-2: } \min \sum_{i\in\mathcal{N}} \frac{dJ_i(R)}{dR}\Big|_{R=R_i^e(t)} R_i(t)$$
$$\text{s.t. Constraints (6), (7), (8), (9), (28)}$$

where the derivative of $J_i(\cdot)$ is computed as

$$\begin{aligned}\frac{dJ_i(R)}{dR}\Big|_{R=R_i^e(t)} &= \frac{w_i T_i}{2}\frac{df\left(\frac{RT_i}{L_i}\right)}{dR}\Big|_{R=R_i^e(t)}\\ &= -\frac{w_i T_i^2}{2L_i}\left(\left\lfloor\frac{L_i}{R_i^e(t)T_i}\right\rfloor^2 + \left\lfloor\frac{L_i}{R_i^e(t)T_i}\right\rfloor\right).\end{aligned} \tag{28}$$

To ensure $f(\cdot)$ is continuously differentiable at every point, we need to define how to perform derivative at certain points. Note that $f(\cdot)$ is a piecewise linear function. When $L_i/(R_i^e(t)T_i)$ is exactly an integer, the function $f(RT_i/L_i)$ is continuous but not differentiable at $R = R_i^e(t)$ (i.e., the left derivative does not equal to the right derivative). For these points, we use the right derivative as the derivative at $R = R_i^e(t)$, as shown in the RHS of (28).

Since each term in the RHS of (28) is either a constant or a known value at TTI t, let us denote the RHS of (28) as $W_i(t)$. Using (9), OPT-3 can be written as

$$\text{OPT-}t\text{: } \min \sum_{i\in\mathcal{N}}\sum_{b\in\mathcal{B}}\sum_{m\in\mathcal{M}} W_i(t) r_i^{b,m}(t) x_i^b(t) y_i^m(t)$$
$$\text{s.t. Constraints (6), (7), (8)}$$

where the optimization variables are $x_i^b(t)$'s and $y_i^m(t)$'s. The minimization problem OPT-t is an IQP, which can be solved by a commercial solver such as CPLEX [37].

Based on OPT-t, we propose a procedure to solve OPT-LB and use it as a lower bound for $\bar{A}^B$, as shown in Fig. 3. Although in theory, it requires $\beta \to 0$ and $T \to \infty$ for the procedure to converge asymptotically; in practice, we can get an excellent approximation even for a small T. We use the following example to illustrate this point.

Example 1: Consider a sizable IoT-network with $|\mathcal{N}| = 100$. Suppose that we have ten types of source nodes as specified in Table III, we assume that our 100 nodes consist of all ten types, with ten nodes in each type. More details on our experimental setup and parameter settings are given in Section VII-A. We assume a Rayleigh fading channel

An Approximate Solution to OPT-LB

Input: β, T.

1: Initialization: $R_i^e(0) = 0$ for each $i \in \mathcal{N}$.
2: **for** $t = 1, 2, \cdots, T$ **do**
3: Solve OPT-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 instant data rate $R_i(t)$.
5: Use (25) to update $R_i^e(t)$.
6: **end for**
7: Let $\overline{R}_i = R_i^e(t)$ for each $i \in \mathcal{N}$.
8: Substitute $\overline{R}_i$'s into the objective function of OPT-LB and use this value as a lower bound for $\bar{A}^{\mathrm{B}}$.

Fig. 3. Procedure to find a lower bound for $\bar{A}^B$.

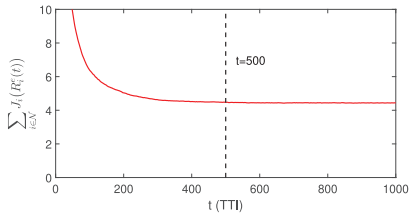


Fig. 4. Convergence time of our proposed approximate solution to OPT-LB when $\beta = 0.01$.

with no frequency nor time correlation. For $\beta = 0.01$, we run the approximate solution for 1000 TTIs. The relationship between $\sum_{i\in\mathcal{N}} J_i(R_i^e(t))$ (which is used as the lower bound for $\bar{A}^B$) and TTI t is shown in Fig. 4. When $t = 500$, we have $\sum_{i\in\mathcal{N}} J_i(R_i^e(t)) = 4.48$ while for $t = 1000$, we have $\sum_{i\in\mathcal{N}} J_i(R_i^e(t)) = 4.44$. Given the negligible difference between the two, we can choose $t = 500$ as our terminating time.

It turns out for all of our experiments in Section VII-A, $t = 500$ is sufficient for our procedure (in Fig. 3) to terminate and obtain an excellent approximate solution.

Note that although the scheduler in Fig. 3 can find an approximate objective value of OPT-LB and use it as a lower bound for $\bar{A}^B$, its own AoI performance is much higher than this lower bound. This is because under this scheduler, the RBs allocated to a particular source node tend to be uniformly distributed across all TTIs (e.g., under a random channel with small coherence time), thus leading to large AoI. On the other hand, an obviously better scheduler would allocate RBs to a source node right after a new sample is generated and use as few TTIs as possible. We use the following example to illustrate this point.

Example 2: Following the same network settings as in Example 1, Fig. 5 shows the AoI performance of the scheduler in Fig. 3 across 1000 TTIs. Also shown in the same figure are the lower bound that we developed and AoI performance of Kronos (the scheduler that we will design in the next section). As we can see, the AoI performance of the scheduler in Fig. 3 is much worse (larger) than Kronos, which confirms our argument that it cannot be directly used as a scheduler to minimize AoI.

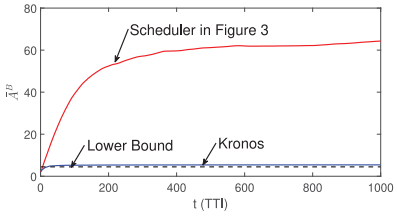


Fig. 5. AoI performance of the scheduler in Fig. 3, Kronos, and the lower bound.

VI. Kronos: Real-Time Scheduler

The goal of this section is twofold. First, we want to design a scheduler that minimizes $\bar{A}^B$. Second, we want to ensure the scheduler can meet the stringent timing requirement in 5G.

We present Kronos,[2] a 5G-compliant real time AoI scheduler that offers near-optimal performance. We organize this section as follows. In Section VI-A, we present the basic design ideas of Kronos. In Section VI-B, we elaborate the details of a key step of Kronos. In Section VI-C, we present a GPU-based implementation for Kronos that can meet the stringent timing requirement in 5G.

A. Basic Idea

The design of Kronos is based on the following key ideas.

1) For the objective function in (5), it is obvious that we need to minimize $\bar{A}_i^B$ from each source node $i \in \mathcal{N}$. For A_i^B, its value at the BS is not reduced until a new sample is received by the BS in its entirety. That is, a partially transmitted sample will not reduce (update) A_i^B at the BS. Based on this observation, we should minimize the number of partially (incomplete) transmission of samples at the end of each TTI. As an extreme, we can limit the number of samples that are partially (incompletely) transmitted at the end of a TTI to at most one. This can be done by devoting all the remaining RBs to one sample, rather than spreading out to multiple samples.
2) Following the last idea, at the beginning of a new (the next) TTI, we will inherit at most one partially (incomplete) transmission of a sample from the previous TTI. Recall that we cannot preempt a sample once it starts transmission, even if there is a newly generated sample from the same source node. Furthermore, for our IoT applications, a sample size is relatively small. So, the remaining portion of the partially transmitted sample is not large (in most cases) and it makes sense to complete its transmission before starting to transmit any other samples.
3) After we complete the transmission of the remaining (incomplete) sample (carried from the last TTI), we need to decide which sample to transmit next in the current TTI. To do this, we need a metric to compare among the samples from different source nodes and decide which subset of samples that we will allocate the remaining RBs. Clearly, this metric should consist of the weight and the "outage" (difference between AoI at the BS

[2]Kronos is the god of time in Greek mythology.

and the source, i.e., $A_i^B(t) - A_i^s(t)$) for each source node $i \in \mathcal{N}$. In our previous work [13], in the absence of considering channel conditions, we use the metric $w_i\Delta_i^2(t)$ for scheduling, where $\Delta_i(t)$ is defined as

$$\Delta_i(t) = A_i^B(t) - A_i^s(t). \tag{29}$$

It was shown in [13] that a scheduler based on this metric can offer near-optimal performance (under simplified channel conditions). Therefore, it would be wise to have Kronos to inherit this basic trait before we add additional features to cope with varying channel conditions.

4) To incorporate channel conditions into the scheduling decision metric, we must consider the impact of the MCS setting on RBs. As shown in the example in Fig. 2, the higher the MCS m is chosen, the fewer number of RBs (with a higher rate) can be used for transmission. Intuitively, we prefer to use as few RBs as possible to transmit a sample. Therefore, the scheduling metric should also include the number of RBs required to transmit a sample, i.e., the more RBs required, the lower the priority (or smaller the metric) associated with a source node. We will elaborate the details of how to incorporating channel conditions into the scheduling metric in the next section.
5) Once we have a scheduling metric (see next section), we can compare samples and perform scheduling, i.e., RB allocation. Clearly, RB allocation is an iterative process, where in each iteration, we will consider how to allocate a subset of RBs among the *remaining* unallocated RBs to a sample in the *remaining* unscheduled samples. Eventually (after a number of iterations), all RBs are allocated and the algorithm terminates.

B. Algorithm Details: Design of Scheduling Metric

We devote this section to the discussion of how channel conditions are incorporated into the scheduling metric, which is at the heart of our design. Recall that the choice of MCS value m at a source node will set the corresponding coding rate c^m, which will, in turn, determine two parameters.

1) The set of RBs in the remaining un-allocated RBs that can contribute at this bit rate c^m. We denote the number in this set as $n_i^m(t)$.
2) The number of RBs that is needed to transmit a sample for source node i, which we denote as s_i^m.

That is

$$n_i^m(t) = \sum_{\text{un-allocated } b} \Big[q_i^b(t) \geq m\Big] \tag{30}$$

where $q_i^b(t)$ (see Section IV) is the maximum MCS that can be used for RB b and source node i for transmission (which is determined by the channel condition on RB b), and "$[\cdot]$" is the notation for the Iverson bracket, returning 1 if the inside statement is true and 0 otherwise [38]. We have

$$s_i^m = \left\lceil \frac{L_i}{c^m} \right\rceil \tag{31}$$

where "$\lceil\cdot\rceil$" is the ceiling function.

Clearly, the scheduling metric for a sample is dependent on m and is a function of $n_i^m(t)$ and s_i^m, in addition to $w_i\Delta^2(t)$ (as discussed in the last section). As a start, denote $V_i^m(t)$ as the scheduling metric under MCS m with the following general form:

$$V_i^m(t) = g\Big(w_i\Delta_i^2(t), n_i^m(t), s_i^m\Big) \tag{32}$$

where "g" denotes a function of $w_i\Delta^2(t)$, $n_i^m(t)$, and s_i^m.

For each sample from source node $i \in \mathcal{N}$, we have the pair $(n_i^m(t), s_i^m)$ under each $m \in \mathcal{M}$. If $n_i^m(t) \geq s_i^m$, it means that this sample can possibly be transmitted in its entirety in this TTI. Otherwise (i.e., $n_i^m(t) < s_i^m$), this sample can only be partially transmitted even if we allocate all the remaining RBs to it. Now, we have a dilemma: shall we transmit a partial sample (while holding back one or more other samples that can otherwise be transmitted in their entirety) or shall we transmit one or more complete samples first?

Since our goal is to minimize (5), based on the *shortest-job-first* principle from queuing theory [39], we should first schedule one or more samples that can be fully transmitted. Therefore, we purposely design the function $g(w_i\Delta_i^2(t), n_i^m(t), s_i^m) > 0$ when $n_i^m(t) \geq s_i^m$ and $g(w_i\Delta_i^2(t), n_i^m(t), s_i^m) < 0$ when $n_i^m(t) < s_i^m$. Under such definition, the priority for a sample that can be fully transmitted within this TTI is always higher than that for a sample that can only be transmitted partially. After RBs have been allocated to those samples that can be fully transmitted, we move on to consider how to allocate the remaining RBs to those samples that cannot be fully transmitted (i.e., samples with $V_i^m(t) < 0$). Recall that in each TTI, we only schedule at most one partially transmitted sample. So, when $V_i^m(t) < 0$ for all remaining source nodes i and MCS m, we will choose one with the largest value of $V_i^m(t) < 0$ for transmission.

Based on the above discussion, we show how to design function $g(w_i\Delta_i^2(t), n_i^m(t), s_i^m)$ as follows.

1) When $n_i^m(t) \geq s_i^m$, sample i can be fully transmitted with un-allocated RBs under MCS m in this TTI. In this case, the fewer RBs required for transmission (i.e., s_i^m), the higher the priority it should have. Therefore, we define function g as

$$g\Big(w_i\Delta_i^2(t), n_i^m(t), s_i^m\Big) = w_i\Delta_i^2(t) \cdot \frac{1}{s_i^m}. \tag{33}$$

2) When $n_i^m(t) < s_i^m$, sample i cannot be fully transmitted under MCS m. In this case, the greater the fraction of the sample that can be transmitted (i.e., $n_i^m(t)/s_i^m$), the higher the priority it should have. Based on this idea, the function g should be proportional to the term $n_i^m(t)/s_i^m$. On the other hand, as discussed earlier, $g(w_i\Delta_i^2(t), n_i^m(t), s_i^m)$ should be negative when $n_i^m(t) < s_i^m$. To ensure this is the case, we can add a negative offset constant and define function g as

$$g\Big(w_i\Delta_i^2(t), n_i^m(t), s_i^m\Big) = w_i\Delta_i^2(t) \cdot \frac{n_i^m(t)}{s_i^m} - C \tag{34}$$

where C is a large (offset) constant that can ensure $g(n_i^m(t), s_i^m) < 0$ for all i and m when $n_i^m(t) < s_i^m$.

Kronos:

At TTI t:

1: If there is a partially transmitted sample carried from the last TTI, allocate minimum number of RBs (based on channel conditions) to complete its transmission.
2: Compute $n_i^m(t)$ by (30). Then compute $V_i^m(t)$ by (35) for all $i \in \mathcal{N}$ that have not been scheduled at t and for all $m \in \mathcal{M}$.
3: Choose i^* and m^* with the largest $V_{i^*}^{m^*}(t)$ among all $V_i^m(t)$'s.
4: **if** $V_{i^*}^{m^*}(t) > 0$ **then**
5: Allocates RBs to source i^* (using MCS m^*) to complete its transmission. Goto Step 2.
6: **else**
7: Allocate all remaining RBs to source i^* (using MCS m^*).
8: **end if**

Fig. 6. Pseudocode for Kronos.

For example, we can set $C = \sum_{i\in\mathcal{N}} w_i \Delta_i^2(t)$ or $C = \max_{i\in\mathcal{N}} w_i \Delta_i^2(t)$.

Combining (32)–(34), the scheduling metric $V_i^m(t)$ is given as

$$V_i^m(t) = \begin{cases} \frac{w_i \Delta_i^2(t)}{s_i^m}, & \text{if } n_i^m(t) \geq s_i^m \\ \frac{w_i \Delta_i^2(t) n_i^m(t)}{s_i^m} - C, & \text{otherwise.} \end{cases} \quad (35)$$

Based on the previous discussions, a pseudocode for Kronos is given in Fig. 6.

We now discuss the complexity of Kronos. To allocate RBs to complete the transmission of the incomplete sample from the last TTI, the time complexity is $O(|\mathcal{B}||\mathcal{M}|)$. After that, if Kronos is implemented sequentially (e.g., in a CPU), then in each iteration, Kronos needs to compute $|\mathcal{N}||\mathcal{M}|$ different $V_i^m(t)$'s, which has a time complexity $O(|\mathcal{N}||\mathcal{M}||\mathcal{B}|)$. After that, Kronos selects i^* and m^* with the largest $V_{i^*}^{m^*}(t)$, which has a time complexity $O(|\mathcal{N}||\mathcal{M}|)$. Then, Kronos allocates RBs to the selected source node i^*, which has a time complexity of $O(|\mathcal{B}|)$. Therefore, the time complexity for each iteration is $O(|\mathcal{N}||\mathcal{M}||\mathcal{B}|) + O(|\mathcal{N}||\mathcal{M}|) + O(|\mathcal{B}|) = O(|\mathcal{N}||\mathcal{M}||\mathcal{B}|)$. Since there are at most $|\mathcal{N}|$ iterations in each TTI, the time complexity for scheduling new samples is $O(|\mathcal{N}|^2|\mathcal{M}||\mathcal{B}|)$. Thus, the total time complexity in each TTI is $O(|\mathcal{B}||\mathcal{M}|) + O(|\mathcal{N}|^2|\mathcal{M}||\mathcal{B}|) = O(|\mathcal{N}|^2|\mathcal{M}||\mathcal{B}|)$.

In one of our experiments in Section VII, we find that for a typical 5G-based IoT network with $|\mathcal{N}| = 100$, $|\mathcal{B}| = 100$ and, $|\mathcal{M}| = 29$, the average running time for Kronos is about ~10 ms, which cannot meet the 5G timing requirement (submillisecond time scale). To address this real-time problem in 5G, we will incorporate parallel computation to speed up its running timing. This will be discussed in the next section.

C. Running Time Speedup: GPU-Based Implementation

Motivation and Basic Idea: We observe that in each iteration of Kronos, the computation of $V_i^m(t)$'s for each i and m is independent from each other. This motivates us to compute them in parallel rather than in sequence. We propose to employ a COTS GPU to compute $V_i^m(t)$'s in parallel. Today's COTS GPUs consist of a large number (1000 s) of processing cores and are highly optimized for massive parallel computations. However, unlike a CPU core, each GPU core processor has very limited computational capability and is designed to handle very simple computations (and thus has a low cost).

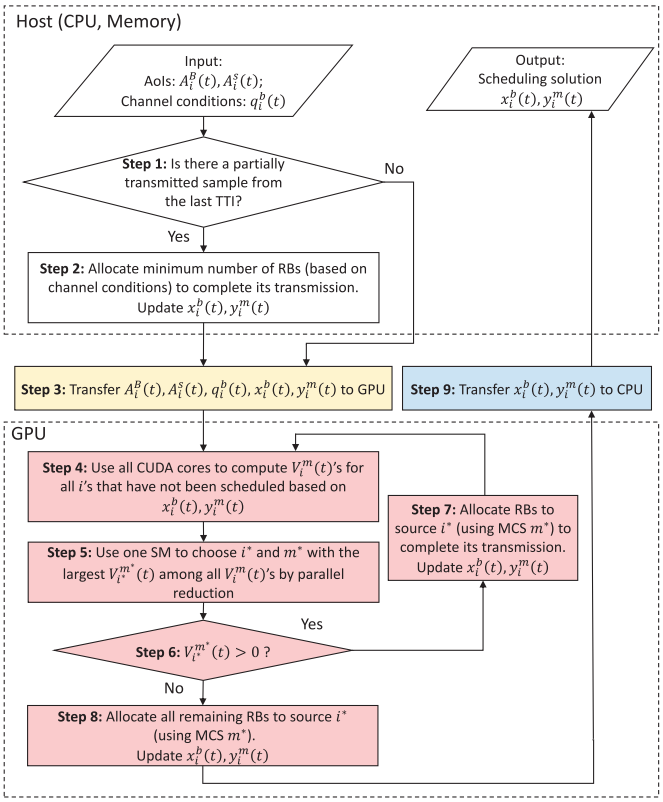


Fig. 7. Flowchart for a GPU-based implementation of Kronos.

To best utilize a GPU's capability, it is utmost important to ensure that each subproblem handled by a GPU core processing is of extremely low complexity and requires a very few iterations to find a solution. Specifically, to calculate $V_i^m(t)$'s for all i's and m's in an iteration, we can decompose this problem into $|\mathcal{N}||\mathcal{M}|$ independent subproblems, each of which is to calculate $V_i^m(t)$ under a specific value of i and m. Recall that the computational complexity of this subproblem is only $O(|\mathcal{B}|)$, which can be done very quickly by GPU cores.

Implementation Details: In our implementation, we employ an NVIDIA Tesla V100 GPU and the CUDA programming platform. This GPU consists of 80 streaming multiprocessors (SMs), with each SM consisting of 64 small processing cores (CUDA cores), i.e., 5120 cores in total. These cores are capable of performing concurrent computation tasks involving arithmetic and logic operations.

Fig. 7 shows the flowchart of our GPU-based implementation of Kronos. Note that steps 1 and 2 are still performed in CPU because GPU cannot help much in these steps.

After step 2, we transfer the AoI information (i.e., $A_i^B(t)$'s and $A_i^s(t)$'s), the channel conditions information (i.e., $q_i^b(t)$'s),

and the scheduling variables (i.e., $x_i^b(t)$'s and $y_i^m(t)$'s) from CPU memory to GPU. This is shown as step 3 in Fig. 7. In our CUDA implementation, to save time, we use the asynchronous mode for this data transfer.

In GPU, we need to perform the following steps.

1) In step 4, we compute $V_i^m(t)$ for all $i \in \mathcal{N}$ that have not been scheduled at t and for all $m \in \mathcal{M}$. In this step, we can use all CUDA cores to compute $V_i^m(t)$'s in parallel. Under CUDA, the subtasks to compute $V_i^m(t)$'s are handled by a grid of thread blocks, each with a certain number of threads. We limit each SM to handle at most one thread block to avoid sequential execution of thread blocks on the same SM.
2) In step 5, we choose i^* and m^* based on the largest $V_{i^*}^{m^*}(t)$. This involves a comparison of $|\mathcal{N}||\mathcal{M}|$ values. We can use parallel reduction [40] to reduce complexity. For example, when $|\mathcal{N}||\mathcal{M}|$ is a power of 2, we can construct an elimination tournament, where only half of $V_i^m(t)$'s survive after each round. After $\log_2(|\mathcal{N}||\mathcal{M}|)$ rounds, the champion (with the largest $V_{i^*}^{m^*}(t)$ among all $|\mathcal{N}||\mathcal{M}|$ $V_i^m(t)$'s) will be found. This parallel reduction technique helps reduce the time complexity to $\log_2(|\mathcal{N}||\mathcal{M}|)$. When $|\mathcal{N}||\mathcal{M}|$ is not a power of 2, we can add some dummy elements at the beginning to increase the number of elements to a power of 2 and then apply parallel reduction. Note that parallel reduction is done in the same (one) SM in our implementation.
3) Steps 6–8 follow the design of Kronos, which have been discussed in detail in the previous section.

Finally, in step 9, we transfer the scheduling solutions for $x_i^b(t)$'s and $y_i^m(t)$'s from GPU back to CPU memory. Again, we use the asynchronous mode in CUDA.

Complexity Analysis: We now examine the computational complexity of our GPU-based implementation. As discussed in Section VI-B, the time complexity for steps 1 and 2 is $O(|\mathcal{B}||\mathcal{M}|)$. Note that these two steps are executed in CPU as they do not involve any parallelism. Time (latency) for data transfer (i.e., steps 3 and 9) between CPU memory and GPU mainly depends on hardware.

The time complexity for step 4, i.e., computing $V_i^m(t)$'s, is $O(|\mathcal{B}|)$. The time complexity for step 5, i.e., choosing the largest $V_i^m(t)$ (with parallel reduction), is $O(\log(|\mathcal{N}||\mathcal{M}|))$. The time complexity of step 7, i.e., RB allocation, is $O(|\mathcal{B}|)$. Therefore, the time complexity of each iteration of steps 4–7 is $O(|\mathcal{B}|) + O(\log(|\mathcal{N}||\mathcal{M}|))$. Recall that there are at most $|\mathcal{N}|$ iterations, so the time complexity in GPU before step 8 is $O(|\mathcal{B}||N|) + O(|N| \cdot \log(|\mathcal{N}||\mathcal{M}|))$. As the last step, the time complexity of step 8 is $O(|\mathcal{B}|)$. Then, the total computation complexity in each TTI is

$$O(|\mathcal{B}||\mathcal{M}|) + O(|\mathcal{B}||\mathcal{N}|) + O(|N| \cdot \log(|\mathcal{N}||\mathcal{M}|)) + O(|\mathcal{B}|)$$
$$= O(|\mathcal{B}| \cdot (|\mathcal{N}| + |\mathcal{M}|)) + O(|\mathcal{N}| \cdot (\log|\mathcal{N}| + \log|\mathcal{M}|)).$$

We emphasize that the above $O(\cdot)$ analysis is only of theoretical interest. What we are really interested in is the actual "wall clock" computational time of our Kronos implementation. In Section VII, we use experiments to show that for a typical 5G IoT network with $|\mathcal{N}| = 100$, $|\mathcal{B}| = 100$, and $|\mathcal{M}| = 29$, the average running time for GPU-based implementation of Kronos is about ∼0.3 ms, which meets 5G requirement (e.g., numerology 0 and 1). This timing performance is more than an order of magnitude faster than that when GPU is not used.

TABLE III
WEIGHTS AND SAMPLING PARAMETERS FOR DIFFERENT TYPES OF SOURCE NODES

Type	w_i	L_i (bits)	T_i (TTIs)
1	8	5400	2
2	2	7200	5
3	10	6800	3
4	6	6200	6
5	5	7600	1
6	2	8200	11
7	9	6000	4
8	1	7100	5
9	4	9600	6
10	3	8400	3

VII. PERFORMANCE EVALUATION

The objective of this section is twofold. First, we will examine the timing performance of Kronos and see if it can meet the real-time requirement under 5G. Second, we will evaluate Kronos in terms of its ability to achieve our objective function. The primary benchmark for this purpose is the lower bound that we developed in Section V.

A. Experiment Setup and Parameter Settings

Our GPU implementation is done on an NVIDIA DGX station with an Intel Xeon E5-2698 v4 CPU (2.20 GHz, 256-GB memory) and an NVIDIA Tesla V100 GPU (32-GB memory). Data communication between CPU and GPU goes through a PCIe 3.0 X16 slot with default configuration. We use CUDA 10.2 to program Kronos in our GPU.

For the source nodes, we assume that there are ten different types, each with different weight, sample size, and sampling period, as shown in Table III.

We will consider different channel fading models, e.g., the Rayleigh fading and Rician fading with different time and frequency correlations. The specific channel fading model will be given in each experiment. Under each fading model, the average channel condition for each source node depends on its physical location. In this article, to help the readers reproduce the same results, we artificially assign an average channel condition for each type of source nodes. In particular, we set the MCS levels corresponding to the average channel conditions of types 1–10 source nodes in Table III to 26, 28, 24, 23, 20, 22, 25, 18, 24, and 21, respectively. For each MCS m, we get the corresponding modulation and coding rate c^m from [4, Table 5.1.3.1-1].

We assume the uplink transmission consists of 100 RBs, i.e., $|\mathcal{B}| = 100$. In each network setting, we run Kronos over 500 TTIs and then calculate the average AoI $\bar{A}^B$. For initialization, $A_i^s(0)$ for each i is set to a random number.

To compute the lower bound (used for benchmark), we use IBM ILOG CPLEX Optimizer (version 12.10.0).

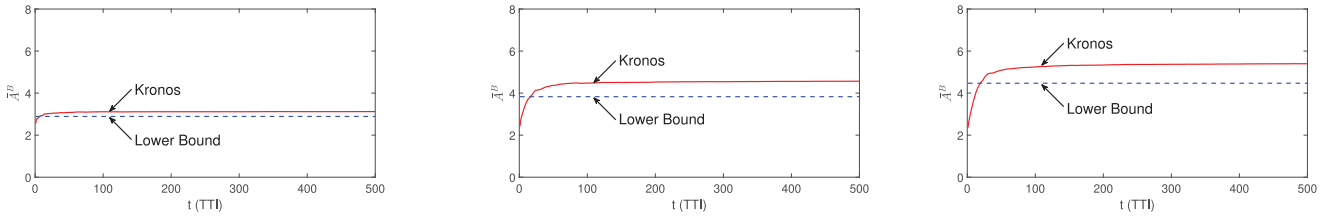


Fig. 8. $\bar{A}^{\mathrm{B}}$ under different numbers of source nodes. (a) $|\mathcal{N}| = 50$. (b) $|\mathcal{N}| = 80$. (c) $|\mathcal{N}| = 100$.

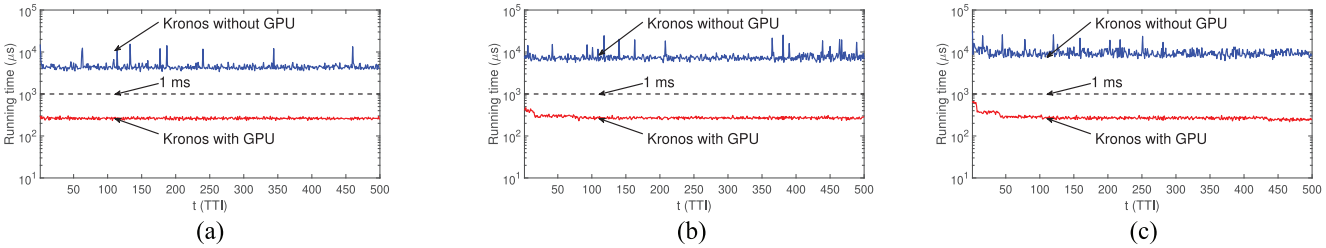


Fig. 9. Running time under different numbers of source nodes. (a) $|\mathcal{N}| = 50$. (b) $|\mathcal{N}| = 80$. (c) $|\mathcal{N}| = 100$.

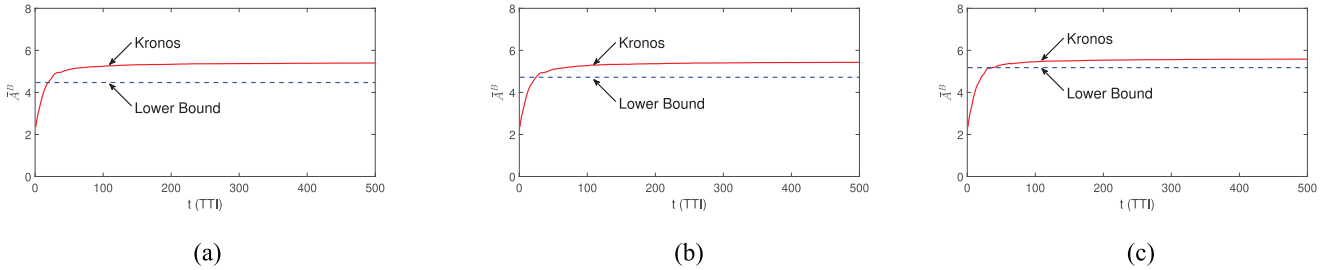


Fig. 10. $\bar{A}^{\mathrm{B}}$ under different Rician factors. (a) $K = 0$. (b) $K = 2$. (c) $K = 10$.

B. Results

Varying Numbers of Source Nodes: We first evaluate Kronos under channels with different number of source nodes. We consider the Rayleigh fading channel with no frequency or time correlation. We consider $|\mathcal{N}| = 50, 80, 100$ source nodes (equally distributed in each source type in Table III). For ease of presentation, we normalize the weight of each source node with respect to $\sum_{i\in\mathcal{U}} w_i$.

Fig. 8 shows the evolution of $\bar{A}^{B}$ under Kronos across 500 TTIs for different $|\mathcal{N}|$'s. In terms of the objective value, both Kronos' implementations (with and without GPU) offer the same values. Also shown in each subfigure is the lower bound by the procedure in Fig. 3. Clearly, we see Kronos can achieve near-optimal performance. In particular, when $|\mathcal{N}| = 50$, 80 and 100, the objectives of Kronos are 8.0%, 19.2%, and 20.8% within their respective lower bounds. Since the optimal objective values lie between Kronos and the lower bound, the gap between Kronos and the optimal is even closer.

Fig. 9 shows the running time for Kronos in each TTI with and without GPU implementation. As a benchmark, we also show the 5G timing requirement for numerology 0 (1 ms) in each subfigure. We can see under all K, the running time of Kronos falls below 1 ms when it is implemented with GPU. When it is implemented only with CPU, it is about $\sim$10 ms. In particular, when $K = 0$, 2, and 10, the average running times of Kronos (with GPU implementation) are 263, 277, and 279 μs, respectively.

Varying Channel Propagation: We now evaluate Kronos under channels with different LOS signal strength. We consider the Rician fading channel with no frequency or time correlation. We consider 100 source nodes (ten from each type in Table III). The weight of each source node is also normalized.

Fig. 10 shows the evolution of $\bar{A}^{B}$ under Kronos across 500 TTIs for different Rician factor K. Also shown in each subfigure is the lower bound by the procedure in Fig. 3. Clearly, we see Kronos can achieve near-optimal performance. In particular, when the Rician factor $K = 0$ (i.e., the Rayleigh fading), 2, and 10, the objectives of Kronos are 20.8%, 15.0%, and 7.7% within their respective lower bounds.

Fig. 11 shows the running time for Kronos in each TTI with and without GPU implementation. As a benchmark, we also show the 5G timing requirement for numerology 0 (1 ms) in each subfigure. We can see under all K, the running time of Kronos falls below 1 ms when it is implemented with GPU. When it is implemented only with CPU, it is about $\sim$10 ms. In particular, when $K = 0$, 2, and 10, the average running times of Kronos (with GPU implementation) are 275, 278, and 260 μs, respectively.

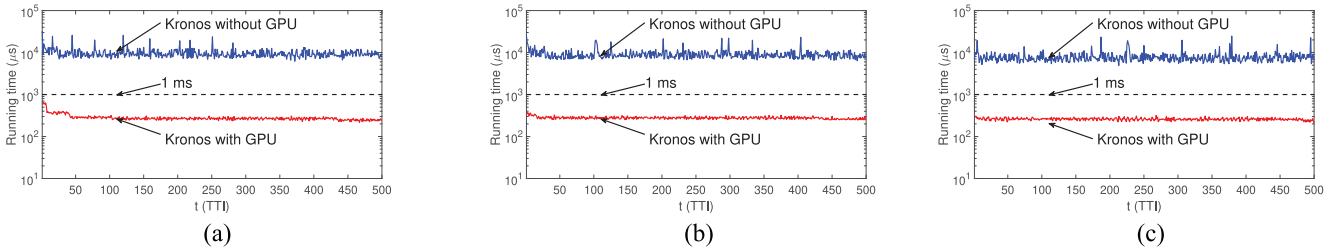


Fig. 11. Running time under different Rician factors. (a) $K = 0$. (b) $K = 2$. (c) $K = 10$.

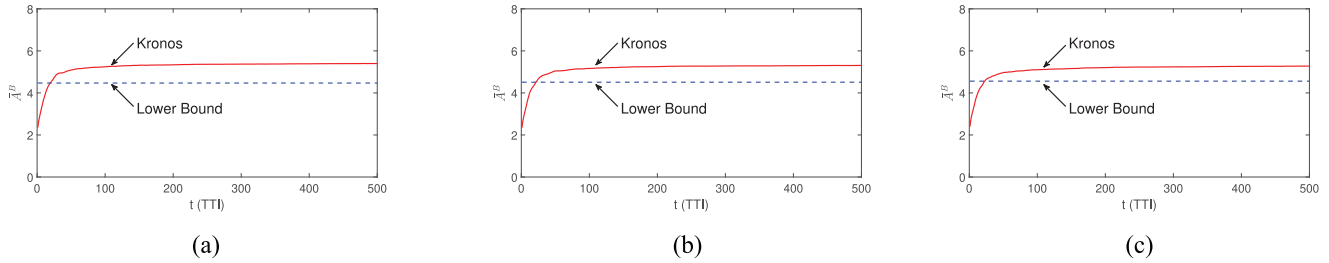


Fig. 12. $\bar{A}^{\mathrm{B}}$ under different coherence bandwidths. (a) $B_c = 1$. (b) $B_c = 4$. (c) $B_c = 10$.

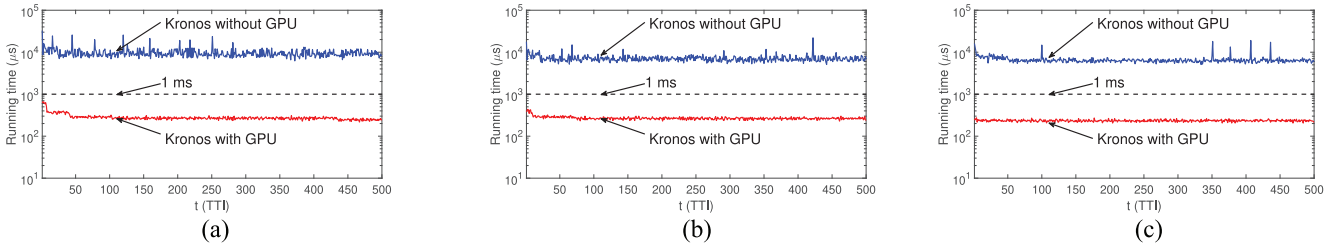


Fig. 13. Running time under different coherence bandwidths. (a) $B_c = 1$. (b) $B_c = 4$. (c) $B_c = 10$.

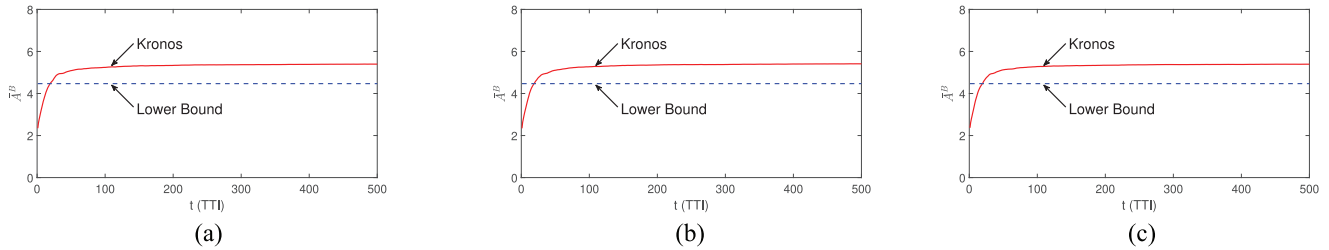


Fig. 14. $\bar{A}^{B}$ under different coherence times. (a) $T_c = 1$. (b) $T_c = 2$. (c) $T_c = 5$.

Varying Frequency Correlation: We now evaluate Kronos under channels with different frequency correlations. We assume the Rayleigh fading channels with no time correlation, and the coherence bandwidth is B_c, i.e., the channel conditions on adjacent B_c RBs are identical for each source node. We consider 100 source nodes (ten from each type in Table III). The weight of each source node is also normalized.

Fig. 12 shows the evolution of $\bar{A}^{B}$ under Kronos across 500 TTIs for different coherence bandwidth B_c. Also shown in each subfigure is the lower bound by the procedure in Fig. 3. Clearly, we see Kronos can achieve near-optimal performance. In particular, when $B_c = 1$ (i.e., no frequency correlation), 4, and 10, the objectives of Kronos are, respectively, 20.8%, 17.7%, and 15.8% within their respective lower bounds.

Fig. 13 shows the running time for Kronos in each TTI with and without GPU implementation. As a benchmark, we also show the 5G timing requirement for numerology 0 (1 ms) in each subfigure. We can see under all coherence bandwidth B_c, the running time of Kronos falls below 1 ms when it is implemented with GPU. When it is implemented only with CPU, it is about $\sim$10 ms. In particular, when $B_c = 1$, 4, and 10, the average running times of Kronos (with GPU implementation) are, respectively, 275, 270, and 234 μs.

Varying Time Correlation: Finally, we evaluate Kronos under channels with different time correlation. We consider the Rayleigh fading channels with no frequency correlation, and the coherence bandwidth is T_c, i.e., the channel conditions on adjacent T_c TTIs are identical for each source node. We consider 100 source nodes (ten from each type in Table III). The weight of each source node is also normalized.

Fig. 14 shows the evolution of $\bar{A}^{B}$ under Kronos across 500 TTIs for different coherence bandwidth T_c. Also shown

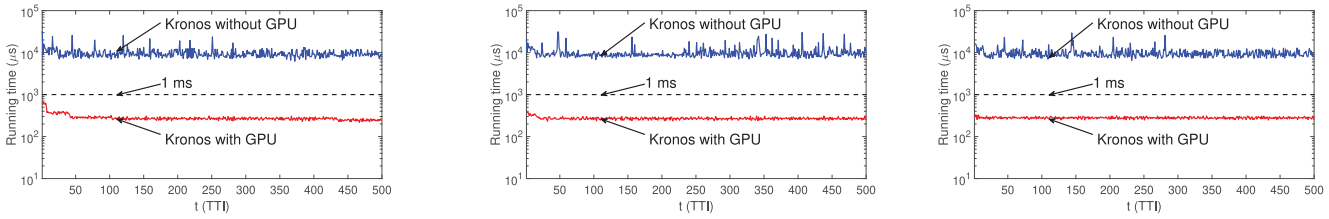


Fig. 15. Running time under different coherence times. (a) $T_c = 1$. (b) $T_c = 2$. (c) $T_c = 5$.

in each subfigure is the lower bound by the procedure in Fig. 3. Clearly, we see Kronos can achieve the near-optimal performance. In particular, when $T_c = 1$ (i.e., no time correlation), 2, and 5, the objectives of Kronos are, respectively, 20.8%, 20.6%, and 21.0% within their respective lower bounds.

Fig. 15 shows the running time for Kronos in each TTI with and without GPU implementation. As a benchmark, we also show the 5G timing requirement for numerology 0 (1 ms) in each subfigure. We can see under all coherence bandwidth T_c, the running time of Kronos falls below 1 ms when it is implemented with GPU. When it is implemented only with CPU, it is about ∼10 ms. In particular, when $T_c = 1$, 2, and 5, the average running times of Kronos (with GPU implementation) are, respectively, 275, 271, and 280 μs.

Summary of Results: In summary, under all network settings (e.g., varying number of source nodes, channel propagation, time, or frequency correlation), we find that the objective value achievement by our GPU-based implementation of Kronos is no more than 21% of the lower bounds. Since the optimal value of the objective (i.e., $\bar{A}^B$) lies between Kronos and the lower bound, the gap between Kronos and the optimal value is even closer. Furthermore, under all network settings, the average running time of Kronos (with GPU implementation) is under 0.3 ms, which meets the 5G timing requirement (e.g., numerology 0 and 1).

VIII. Conclusion

This article presented Kronos—the first successful design of a 5G-compliant real-time AoI scheduler. Kronos is capable of performing RB allocation and MCS selection for each source node based on channel conditions within each TTI (in sub-millisecond time scale). To cope with the enormous search space for the optimal solution and the unknown nature of channel conditions, we developed a novel computation procedure to find an asymptotic lower bound for the objective as a performance benchmark. We further developed a novel metric, which is used by Kronos to select a source node, allocate RBs, and determine MCS in each iteration. To meet the stringent timing requirement in 5G, we proposed to implement Kronos on COTS GPU by exploiting its massive parallel computing capability. For demonstration, we implemented Kronos on an NVIDIA Tesla V100 GPU. Through extensive simulation experiments, we showed that Kronos can find a near-optimal AoI scheduling solution while complying the 5G standard and its stringent timing requirement.

References

[1] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *Proc. IEEE SECON*, Salt Lake City, UT, USA, Jun. 2011, pp. 350–358.

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

[3] Y. Sun. *A Collection of Recent Papers on the Age of Information*. Accessed: Jul. 12, 2020. [Online]. Available: http://www.auburn.edu/%7eyzs0078

[4] *NR; NR and NG-RAN Overall Description, Version 15.0.0*, 3GPP Standard TS 38.214. Accessed: Jul. 12, 2020. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3216

[5] "AT&T edge cloud (AEC)," AT&T Labs & AT&T Foundry, Dallas, TX, USA, White Paper. Accessed: Jul. 12, 2020. [Online]. Available: https://about.att.com/content/dam/innovationdocs/Edge_Compute_White_Paper%20FINAL2.pdf

[6] *The Internet of Things Will Thrive on 5G Technology*, Verizon, Basking Ridge, NJ, USA. Accessed: Jul. 12, 2020. [Online]. Available: https://www.verizon.com/about/our-company/5g/internet-things-will-thrive-5g-technology

[7] Y.-P. Hsu, E. Modiano, and L. Duan, "Age of information: Design and analysis of optimal scheduling algorithms," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Archen, Germany, Jun. 2017, pp. 561–565.

[8] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Minimizing the age of information in broadcast wireless networks," in *Proc. 54th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, Monticello, IL, USA, Sep. 2016, pp. 844–851.

[9] J. Zhong, R. D. Yates, and E. Soljanin, "Two freshness metrics for local cache refresh," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 1924–1928.

[10] C. Li, S. Li, Y. Chen, Y. T. Hou, and W. Lou, "AoI scheduling with maximum thresholds," in *Proc. IEEE INFOCOM*, Jul. 2020, pp. 436–445.

[11] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Optimizing data freshness, throughput, and delay in multi-server information-update systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 2569–2573.

[12] Y. Sun, E. Uysal-Biyikoglu, and S. Kompella, "Age-optimal updates of multiple information flows," in *Proc. IEEE INFOCOM Workshop - Age of Inf. Workshop*, Honolulu, HI, USA, Apr. 2018, pp. 136–141.

[13] C. Li, S. Li, and Y. T. Hou, "A general model for minimizing age of information at network edge," in *Proc. IEEE INFOCOM*, Paris, France, 2019, pp. 118–126.

[14] Q. He, D. Yuan, and A. Ephremides, "Optimal link scheduling for age minimization in wireless systems," *IEEE Trans. Inf. Theory*, vol. 64, no. 7, pp. 5381–5394, Jul. 2018.

[15] C. Joo and A. Eryilmaz, "Wireless scheduling for information freshness and synchrony: Drift-based design and heavy-traffic analysis," in *Proc. 15th Int. Symp. Model. Optim. Mobile Ad Hoc Wireless Netw. (WiOpt)*, Paris, France, May 2017, pp. 1–8.

[16] R. Talak, S. Karaman, and E. Modiano, "Improving age of information in wireless networks with perfect channel state information," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1765–1778, Aug. 2020.

[17] N. Lu, B. Ji, and B. Li, "Age-based scheduling: Improving data freshness for wireless real-time traffic," in *in Proc. ACM MobiHoc*, Los Angeles, CA, USA, Jun. 2018. pp. 191–200,

[18] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Age-optimal information updates in multihop networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Archen, Germany, Jun. 2017. pp. 576–580.

[19] R. Talak, S. Karaman, and E. Modiano, "Minimizing age-of-information in multi-hop wireless networks," in *Proc. 55th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, Monticello, IL, USA, Oct. 2017, pp. 486–493.

[20] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Modeling the age of information in emulated ad hoc networks," in *Proc. IEEE IEEE Mil. Commun. Conf. (MILCOM)*, Baltimore, MD, USA, Oct. 2017, pp. 436–441.

[21] X. Zheng, S. Zhou, Z. Jiang, and Z. Niu, "Closed-form analysis of non-linear age of information in status updates with an energy harvesting transmitter," *IEEE Trans. Wireless Commun.*, vol. 18, no. 8, pp. 4129–4142, Aug. 2019.

[22] V. Tripathi and E. Modiano, "A whittle index approach to minimizing functions of age of information," in *Proc. 57th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, Monticello, IL, USA, Sep. 2019, pp. 1160–1167.

[23] B. Li, A. Eryilmaz, and R. Srikant, "On the universality of age-based scheduling in wireless networks," in *in Proc. IEEE INFOCOM*, Kowloon, Hong Kong, 2015, pp. 1302–1310.

[24] A. Maatouk, M. Assaad, and A. Ephremides, "On the age of information in a CSMA environment," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 818–831, Apr. 2020.

[25] H. Tang, J. Wang, L. Song, and J. Song, "Scheduling to minimize age of information in multi-state time-varying networks with power constraints," in *Proc. 57th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, Monticello, IL, USA, Sep. 2019, pp. 1198–1205.

[26] H. Tang, J. Wang, L. Song, and J. Song, "Minimizing age of information with power constraints: Multi-user opportunistic scheduling in multi-state time-varying channels," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 854–868, May 2020.

[27] Y. Xiao and Y. Sun, "A dynamic jamming game for real-time status updates," in *Proc. IEEE INFOCOM Workshop - Age of Inf. Workshop*, Honolulu, HI, USA, Apr. 2018, pp. 354–360.

[28] G. D. Nguyen, S. Kompella, C. Kam, J. E. Wieselthier, and A. Ephremides, "Information freshness over an interference channel: A game theoretic view," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, Apr. 2018, pp. 908–916.

[29] Z. Ning *et al.*, "Mobile edge computing enabled 5G health monitoring for Internet of Medical Things: A decentralized game theoretic approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 463–478, Feb. 2021.

[30] J. Zhong, R. D. Yates, and E. Soljanin, "Backlog-adaptive compression: Age of information," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 566–570.

[31] R. Devassy, G. Durisi, G. C. Ferrante, O. Simeone, and E. Uysal-Biyikoglu, "Delay and peak-age violation probability in short-packet transmissions," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 2471–2475.

[32] P. Mayekar, P. Parag, and H. Tyagi, "Optimal source codes for timely updates," *IEEE Trans. Inf. Theory*, vol. 66, no. 6, pp. 3714–3731, Jun. 2020.

[33] C. Kam, S. Kompella, and A. Ephremides, "Experimental evaluation of the age of information via emulation," in *Proc. IEEE IEEE Mil. Commun. Conf. (MILCOM)*, Tampa, FL, USA, Oct. 2015, pp. 1070–1075.

[34] V. Marbukh, "Towards managing age of network state information in challenged networks," in *Proc. IEEE INFOCOM Workshop - Age of Inf. Workshop*, Honolulu, HI, USA, Apr. 2018, pp. 1–2.

[35] R. Agrawal and V. Subramanian, "Optimality of certain channel aware scheduling policies," in *Proc. 40th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, Monticello, IL, USA, Oct. 2002, pp. 1533–1542.

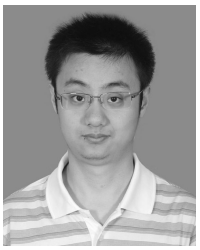
[36] A. L. Stolyar, "On the asymptotic optimality of the gradient scheduling algorithm for multiuser throughput allocation," *Oper. Res.*, vol. 53, no. 1, pp. 12–25, 2005.

[37] *IBM ILOG CPLEX Optimizer*. Accessed: Jul. 12, 2020. [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer

[38] R. L. Garham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics*. Reading, MA, USA: Addison-Wesley, 1989, ch. 2.

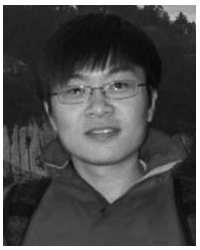
[39] A. S. Tanenbaum and A. S. Woodhaul, *Operating Systems: Design and Implementation*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1987, ch. 4.

[40] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Upper Saddle River, NJ, USA: Addison-Wesley, 2010, ch. 5.



Chengzhang Li (Graduate Student Member, IEEE) received the B.S. degree in electronics engineering from Tsinghua University, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree with the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA.

His research interests are modeling, analysis and algorithm design for wireless networks, with a focus on age of information, 5G, and ultralow-latency research.



Yan Huang (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2012 and 2015, respectively, and the Ph.D. degree in electrical engineering from Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, in 2020.

He is a Senior System Software Engineer with NVIDIA, Santa Clara, CA, USA. His research interests include GPU-based real-time optimizations and machine learning for wireless communications and networking.

Dr. Huang was awarded the Pratt Scholarship in 2019 and the Bindi Prasad Scholarship in 2020 during his Ph.D. study at Virginia Polytechnic Institute and State University.

Shaoran Li (Graduate Student Member, IEEE) received the B.S. degree from Southeast University, Nanjing, China, in 2014, and the M.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree with Virginia Polytechnic Institute and State University, Blacksburg, VA, USA.

His research interests include algorithm design and implementation for wireless networks.

Mr. Li won the Fred W. Ellersick MILCOM Award for the Best Paper in the Unclassified Technical Program in 2019.

Yongce Chen (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree with the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA.

His current research interests include optimization, MIMO techniques, and real-time implementation for wireless networks.

Brian A. Jalaian (Member, IEEE) received the M.S. and Ph.D. degrees in electrical engineering and the M.S. degree in industrial and systems engineering (operation research) from Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, in 2013, 2016, and 2014, respectively.

He is a Research Scientist and a Research Lead with Army Research Laboratory, Adelphi, MD, USA, and an Adjunct Research Assistant Professor with the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University. His research interests are optimization, machine learning, and network science.

Dr. Jalaian was a recipient of the 2019 Fred W. Ellersick MILCOM Award for the Best Paper in the unclassified technical program.

Y. Thomas Hou (Fellow, IEEE) received his Ph.D. degree from NYU Tandon School of Engineering in 1998. He is the Bradley Distinguished Professor of Electrical and Computer Engineering at Virginia Polytechnic Institute and State University, Blacksburg, VA, USA. He has published over 300 papers in IEEE/ACM journals and conferences. He holds six U.S. patents. He has authored/coauthored two textbooks: *Applied Optimization Methods for Wireless Networks* (Cambridge University Press, 2014) and *Cognitive Radio Communications and Networks: Principles and Practices* (Academic Press/Elsevier, 2009). He is also interested in wireless security. His current research focuses on developing innovative solutions to complex science and engineering problems arising from wireless and mobile networks.

Dr. Hou's papers were recognized by eight best paper awards from IEEE and ACM. He was/is on the editorial boards of a number of IEEE and ACM transactions and journals. He served as Steering Committee Chair for IEEE INFOCOM conference and was a member of the IEEE Communications Society Board of Governors. He was also a Distinguished Lecturer of the IEEE Communications Society.

Wenjing Lou (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2003.

She is the W. C. English Endowed Professor of Computer Science with Virginia Polytechnic Institute and State University, Arlington, VA, USA. Her research interests cover many topics in the cybersecurity field, with her current research interests focusing on wireless networks, privacy protection in machine learning systems, and security and privacy problems in the Internet-of-Things systems.

Prof. Lou received the Virginia Tech Alumni Award for Research Excellence in 2018, the highest university level faculty reseach award, and the IEEE INFOCOM Test-of-Time Paper Award in 2020. She is a Highly Cited Researcher by the Web of Science Group. She served as a TPC Chair for IEEE INFOCOM 2019 and ACM WiSec 2020. She was the Steering Committee Chair for IEEE CNS conference from 2013 to 2020. She is currently a Steering Committee Member of IEEE INFOCOM and IEEE TRANSACTIONS ON MOBILE COMPUTING. She served as a Program Director at U.S. National Science Foundation from 2014 to 2017.

Jeffrey H. Reed (Fellow, IEEE) received the Ph.D. degree in Electrical and Computer Engineering from University of California, Davis, in 1987. He is the Willis G. Worcester Professor with the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, where he is the Founder of *Wireless@VT* and served as its Director until 2014. He is the Founding Faculty Member of the Ted and Karyn Hume Center for National Security and Technology and served as its Interim Director when founded in 2010. He served as an Interim Director for the Commonwealth Cyber Initiative during its establishment and now serves as its CTO. He has published the book *Software Radio: A Modern Approach to Radio Design* (Prentice Hall, 2022), *Cellular Communications: A Comprehensive and Practical Guide* (Wiley–IEEE, in 2014), and his new eBook *5G Cellular Communications: Journey and Destination* (An ebook, 2019).

Dr. Reed was awarded the International Achievement Award by the Wireless Innovations Forum in 2013. In 2012, he served on the President's Council of Advisors of Science and Technology (PCAST) advisory group that examined ways to transition federal spectrum for commercial use. He is a Past Member of CSMAC, a group that provides advice to NTIA on spectrum issues. In 2005, he was named an IEEE Fellow for contributions to software radio and communications signal processing and for leadership in engineering education.

Sastry Kompella (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, in 2006.

He is currently the Section Head of the Wireless Network Research Section, Information Technology Division, U.S. Naval Research Laboratory, Washington, DC, USA. His research interests include various aspects of wireless networks, including cognitive radio, dynamic spectrum access, and age of information.