Aion: A Bandwidth Optimized Scheduler with AoI Guarantee

Qingyu Liu* Chengzhang Li* Y. Thomas Hou* Wenjing Lou* Sastry Kompella†
*Virginia Tech, Blacksburg, VA, USA
†U.S. Naval Research Laboratory, Washington, DC, USA

IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488781

***Abstract*—This paper investigates bandwidth minimization under AoI constraints – a fundamental problem that has not been studied in AoI research. The problem is of critical importance in bandwidth-limited IoT environment when AoI is used as a constraint. We present a novel fast algorithm called Aion that can construct a scheduler to satisfy AoI constraints with strong theoretical guarantee in terms of minimizing required bandwidth. Specifically, we prove that the bandwidth required by Aion is minimum if the AoI constraint vector meets a special mathematical structure called *Fractional Consecutively Divisible* (FCD). In the general case when the given AoI constraint vector is not FCD, we prove that the bandwidth required by Aion is tightly upper bounded by a factor of the minimum. The results from this paper lay the foundation for future research on bandwidth minimization with AoI guarantee.**

***Index Terms*—Age of Information, AoI Constraint, Bandwidth Minimization, Scheduling, Wireless Communication.**

I. Introduction

Age of Information (AoI) is an application layer metric used to quantify the freshness of information [1], [2]. It measures the elapsed time between the present and the generation time of the information. Since its inception, AoI has attracted active research efforts in the research community (see [3] for a comprehensive bibliography). A main line of research has been focused on minimizing AoI under certain resource constraints, with resource ranges from bandwidth (e.g., [4]–[9]), to energy (e.g. [10]–[14]), and to throughput (e.g., [15]–[18]). Another line of research has been focused on modeling, analysis, and optimization of AoI (e.g., [19]–[24]). There are also some other branches on AoI research, e.g., game theory for AoI (e.g., [25]–[28]), channel coding for AoI (e.g. [29]–[32],), and AoI applications (e.g., [33]–[35]), to name a few.

Recently, there has been a new line of research on problems under AoI constraints, with the objective of either determining the feasibility of AoI constraints (e.g., [36]) or optimizing energy subject to feasible AoI constraints (e.g., [37]).

Indeed, AoI constraints arise from many important IoT applications such as autonomous vehicles, industrial automation, telemedicine, smart building management, to say a few. To guarantee AoI constraints, one of the most important network resources that needs to be consumed is bandwidth. Bandwidth resource is especially of concern when an IoT system operates in a bandwidth-limited environment. A well-known example is the wireless channel in an access network, which is typically the first hop for collecting information from the IoT source nodes to the edge Base Station (BS). In this environment, a scheduler that uses the minimum bandwidth to guarantee AoI is of critical importance. Unfortunately, to date, there is hardly any research on bandwidth minimization under AoI constraints in the community. This may be attributed to the significant technical challenge that the underlying problem entails.

In this paper, we study the problem of bandwidth minimization under AoI constraints – a long overdue, but critically important problem in understanding and harnessing AoI. We employ the most widely accepted model for edge IoT data collection, which consists of a BS and a set of source nodes. We focus on uplink data collection where information collected at the sources should be sent to the BS. The communication between all source nodes and the BS is through a wireless channel. Clearly, if one had infinite bandwidth on the wireless channel, then the AoI of all information maintained at the BS (from the source nodes) will always be fresh and meet the AoI constraints. But for a bandwidth-limited scenario, which is what one would encounter in practice, minimizing the required bandwidth of the wireless channel while guaranteeing the AoI constraints for all sources is a challenging problem. The main contributions of this paper are summarized as follows:

- This is the first paper to investigate bandwidth minimization with AoI guarantee. For this problem, we present a novel fast algorithm called Aion[1] that can always construct a scheduler that satisfies $\boldsymbol{d}$ for any AoI constraint vector $\boldsymbol{d}$ for N source nodes.
- We show that Aion has strong theoretical performance guarantee for minimizing the required channel bandwidth. Our theoretical results for Aion are based on a mathematical structure of a constraint vector $\boldsymbol{d}$ that we call *Fractional Consecutively Divisible* (FCD). Specifically, we show that if the AoI constraint vector $\boldsymbol{d}$ is FCD, then Aion can always construct a scheduler ϕ that uses the minimum bandwidth K^*. In the general case when $\boldsymbol{d}$ is not FCD, we show that the bandwidth required by ϕ is tightly upper bounded by a factor $\left(\left\lceil \sum_{i=1}^{N} \frac{1}{l_i^*} \right\rceil / \left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil\right)$ of K^*, where l_i^* is the i-th element of a FCD vector $\boldsymbol{l}^*$.
- The performance of Aion hinges upon how to map a general AoI constraint vector $\boldsymbol{d}$ to a FCD vector $\boldsymbol{l}^*$ so that $\left\lceil \sum_{i=1}^{N} \frac{1}{l_i^*} \right\rceil$ is minimized. This is the first problem that our Aion will address. By exploiting an intrinsic property that relates $\boldsymbol{d}$ and $\boldsymbol{l}^*$, we develop an algorithm (RCS)

[1]Aion is a Hellenistic deity associated with time, the orb or circle encompassing the universe, and the zodiac.

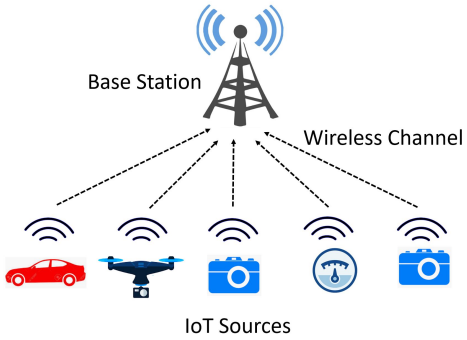

Fig. 1. System model: N IoT source nodes collect data samples from environment and forward them to a BS through a shared wireless channel.

TABLE I
SUMMARY OF KEY NOTATIONS.

Notations	Definitions
N	Number of sources to be scheduled
K	Bandwidth achieved by a scheduler
$A_i(t)$	AoI for source i at the BS at time t
$G_i(t)$	Generation time of the sample which is from source i and stored at the BS at time t
$\phi^K(t)$	Set of scheduling decisions at time t under a bandwidth K
$S_j(t)$	Source scheduled for transmission using the j-th bandwidth unit at time t
$\boldsymbol{x}$	A general vector $[x_1\ x_2\ \cdots\ x_N]$
d_i	AoI threshold for source i
$\boldsymbol{d}$	A vector of AoI constraints $[d_1\ d_2\ \cdots\ d_N]$
K^*	Minimum bandwidth that can be achieved by any scheduler under AoI constraints $\boldsymbol{d}$

that can substantially reduce the search space for $\boldsymbol{l}^*$. Subsequently, we show that the optimal mapping problem (from $\boldsymbol{d}$ to $\boldsymbol{l}^*$) can be transformed into a shortest path problem, which is a very interesting result on its own.

- Once a general AoI constraint vector $\boldsymbol{d}$ is mapped optimally to a FCD vector $\boldsymbol{l}^*$, our Aion needs to have an algorithm to find a scheduler ϕ that can satisfy $\boldsymbol{l}^*$ with a bandwidth $K_{\boldsymbol{l}^*} = \left\lceil \sum_{i=1}^{N} \frac{1}{l_i^*} \right\rceil$. We present an algorithm (SUD) that does this job. This is the second key component in Aion. The key idea of SUD is to use a novel technique to scale up and scale down cycle length to handle fractional $\boldsymbol{l}^*$. A time slot grouping method (and a corresponding dynamic resource accounting) is embedded in SUD to ensure that the final cyclic scheduler (ϕ) retains its discrete nature (in integral time slots).
- We evaluate the performance of Aion using a large number of simulations for the general case of $\boldsymbol{d}$. For all simulated instances, Aion satisfies the AoI constraint vector $\boldsymbol{d}$, which confirms Aion's theoretical guarantee. Further, for all simulated instances, the bandwidth $K_{\boldsymbol{l}^*}$ required by Aion is close to a lower bound of K^*.

II. Problem Statement

Consider a data collection scenario where there are N sources and one BS (see Fig. 1). Each source collects data samples from environment and forwards them to BS through a wireless channel that is shared by all N sources. Assume uplink transmission time is slotted. For clarity of terminology, we use the term "at time t" to refer to "at the beginning of time slot t" and use the term "in time slot t" to refer to the underlying action is completed "at the end of time slot t".

When a source is chosen for transmission at time t, it will transmit its most recent (freshest) sample to the BS. This will help minimize the AoI of this source at the BS. In this research, we assume (i) the transmission of a sample takes exactly one time slot; and (ii) at most $K \in \mathbb{Z}^+$ samples can be transmitted in each time slot, where $\mathbb{Z}^+$ denotes the set of positive integers. Assumption (ii) is based on the state-of-the-art transmission technologies used in cellular networks (e.g., 5G, 4G LTE) where there is a large number of Resource Blocks (RBs) in each time slot that can be used for transmission by multiple users. For ease of reference, we call K the *bandwidth* of the shared wireless channel. Naturally, K is a precious commodity and the objective of this research is to minimize K while meeting AoI constraints from the sources.

Following some widely used AoI models (see, e.g., [8], [9], [36], [38], [39]), we assume each source collects a data sample in each time slot, and the BS only stores the most recently received sample from each source. For the sample from source i that is maintained by the BS at time t, denote $G_i(t)$ as the generation time of this sample at source i. Then the AoI for this sample at the BS at time t, denoted by $A_i(t)$, can be defined as the elapsed time between t and $G_i(t)$, i.e., $A_i(t) = t - G_i(t)$ for $i \in \{1, 2, \cdots, N\}$ and $t \in \mathbb{Z}^+$. $A_i(t+1)$ is reset to 1 whenever there is a new sample coming from source node i in time slot t, or it is incremented by 1. That is,

$$A_i(t+1) = \begin{cases} 1, & \text{if a sample from source } i \text{ is transmitted to BS in } t; \\ A_i(t) + 1, & \text{otherwise,} \end{cases}$$

for $i \in \{1, 2, \cdots, N\}$. We assume the BS does not have any samples in the beginning. Therefore, we can define $A_i(1) = +\infty$ for each source $i \in \{1, 2, \cdots, N\}$.

We assume that each source i has a pre-assigned AoI threshold (deadline) $d_i \in \mathbb{Z}^+$ at the BS. That is, the AoI of source i must not be more than d_i at the BS. We denote $\boldsymbol{d} = [d_1\ d_2\ \cdots\ d_N]$ as the vector of AoI constraints for all N sources at the BS. Without loss of generality, we assign the node index among the source nodes so that the elements in $\boldsymbol{d}$ are in *non-decreasing* order, i.e., $d_1 \le d_2 \le \cdots \le d_N$. To satisfy this AoI constraint vector $\boldsymbol{d}$, a properly designed scheduler at the BS (to determine which subset of source nodes to transmit their samples in each time slot) is critical to conserve bandwidth K. The goal of this paper is to find such an optimal scheduler that consumes the smallest K (denoted as K^*) while meeting the AoI constraint vector $\boldsymbol{d}$.

For a given K, denote $\phi^K(t) = \{S_1(t), \cdots, S_K(t)\}$ as the set of sources chosen for transmission at time $t \in \mathbb{Z}^+$, where $S_j(t) = i$ denotes that the j-th bandwidth unit is to be used to transmit the sample from source i at time t. Given that

we assume $d_1 \leq d_2 \leq \cdots \leq d_N$, we will follow the same non-decreasing order for the elements in $\phi^K(t)$, i.e., $S_1(t) \leq S_2(t) \leq \cdots \leq S_K(t)$. Such notation will offer us much clarity in identifying each unique scheduling solution. Note that it is possible that not all K bandwidth units are allocated for transmission. In this case, we assume the lower indexed bandwidth units are used consecutively for transmission and the higher indexed bandwidth units are idle. In particular, when the j-th bandwidth unit is not scheduled to transmit any sample from the sources, we denote $S_j(t) = \text{NULL}$.

More formally, we want to find a scheduler $\{\phi^{K^*}(t) : t \in \mathbb{Z}^+\}$ that minimizes the required bandwidth K^* while the AoI constraint vector $\boldsymbol{d}$ can be satisfied at the BS after the BS receives at least one sample from all N sources.

In the end of this section, we present an important lemma which gives both lower and upper bounds for K^*. This lemma lays the foundation for our algorithm design.

Lemma 1: For any AoI constraint vector $\boldsymbol{d} = [d_1\ d_2\ \cdots\ d_N]$,

$$\left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil \ \leq\ K^* \ \leq\ N.$$

Proof: The upper bound is trivial. We prove the lower bound. For each source i, to satisfy d_i, there must be at least one transmission over consecutive d_i time slots, and hence

$$\frac{1}{d_i} \ \leq\ \lim_{T\to\infty} \frac{1}{T} \cdot \sum_{t=1}^{T} \mathbf{1}_A(i,t), \quad \text{for } i = 1, 2, \cdots, N,$$

where $\mathbf{1}_A(i,t)$ is the indicator function that is equal to 1 if the source i is chosen to transmit its sample at time t, and is equal to 0 otherwise. Overall, if K is the bandwidth of a feasible scheduler that satisfies AoI constraints, we shall have

$$\begin{aligned} \sum_{i=1}^{N} \frac{1}{d_i} \quad &\leq\ \lim_{T\to\infty} \frac{1}{T} \cdot \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbf{1}_A(i,t) \\ &=\ \lim_{T\to\infty} \frac{1}{T} \cdot \sum_{t=1}^{T} \sum_{i=1}^{N} \mathbf{1}_A(i,t) \\ &\leq\ \lim_{T\to\infty} \frac{1}{T} \cdot \sum_{t=1}^{T} K \ =\ K. \end{aligned}$$

Hence the lower bound holds, considering that $K \in \mathbb{Z}^+$. ∎

III. When $\boldsymbol{d}$ is Consecutively Divisible

For the special case when $K = 1$, Li *et al.* [36] designed a fast procedure called PSC (Polynomial Scheduler Construction) that can always construct a scheduler satisfying AoI constraints $\boldsymbol{d}$ if $\boldsymbol{d}$ is a *polynomial* vector. Specifically, they define that a vector $\boldsymbol{x} = [x_1\ x_2\ \cdots\ x_N]$ is a polynomial vector if each element in $\boldsymbol{x}$ can be expressed as a polynomial term with base 2, i.e., $x_i = b \cdot 2^{n_i}$ for each i, where b is a positive integer and n_i is a non-negative integer. When $\boldsymbol{d}$ is polynomial, their proposed PSC can construct a cyclic scheduler satisfying $\boldsymbol{d}$ by directly setting the cycle length to d_N and assigning every d_i time slots in a cycle to source i for all $i \in \{1, 2, \cdots, N\}$.

Algorithm 1 Consecutively Divisible Scheduler (CDS)

1: **input**: $\boldsymbol{d} = [d_1\ d_2\ \cdots\ d_N]$ that is consecutively divisible
2: **output**: A cyclic scheduler ϕ
3: **procedure**
4: Set the cycle length of ϕ to d_N
5: Set $K = \left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil$ as the bandwidth for ϕ
6: **for** $i = 1, 2, \cdots, N$ **do**
7: Obtain the first available slot in the cycle of ϕ that can be used to transmit one sample and assign it to source i
8: Assign every d_i slots following the slot obtained in Line 7 in the cycle of ϕ to source i

In this section, we extend this result to a more general form beyond base 2. First, we introduce *consecutively divisible*:

Definition 1: A vector $\boldsymbol{x} = [x_1\ x_2\ \cdots\ x_N]$ *is consecutively divisible if* $x_i \in \mathbb{Z}^+$ *for each* $i \in \{1, 2, \cdots, N\}$ *and* $x_i/x_{i-1} \in \mathbb{Z}^+$ *for each* $i \in \{2, 3, \cdots, N\}$.

Comparing a polynomial vector with a consecutively divisible vector, they both require each element in the vector to be a multiple of any element prior to it. However, each element in the polynomial vector is required to be a power of 2, while each element in the consecutive divisible vector does not have this requirement and can offer much larger design space.

Theorem 1: For any vector $\boldsymbol{d} = [d_1\ d_2\ \cdots\ d_N]$ *that is consecutively divisible, there exists a cyclic scheduler that requires the minimum bandwidth* K^* *to satisfy the AoI constraint vector* $\boldsymbol{d}$.

A proof of Theorem 1 is based on construction. Specifically, in Algorithm 1, we present a procedure called *Consecutively Divisible Scheduler* (CDS)[2]. CDS first sets the bandwidth to be the lower bound of K^* (see Lemma 1), i.e., $K = \left\lceil \sum_{i=1}^{N} (1/d_i) \right\rceil$, and sets the cycle length of the cyclic scheduler to be d_N. Then it assigns time slots to sources $1, 2, \cdots, N$ iteratively. In the i-th iteration, CDS finds the first available slot that can be used to transmit one sample and assigns it to source i. Subsequently, CDS assigns every d_i slots following the first assigned slot to source i. A proof (based on contradiction) of the correctness of CDS can be easily constructed. Due to space limitation, we omit it here. It is easy to show that CDS has a time complexity of $O(N \cdot d_N)$.

IV. Aion – An Optimal Scheduler

The results for CDS are inspiring but still suffer from a serious limitation – it only works for consecutively divisible AoI constraint vector $\boldsymbol{d}$. In this section, we address this limitation by presenting a novel fast algorithm (with codename Aion) that can be applied to any AoI constraint vector $\boldsymbol{d}$ and provide strong theoretical guarantee to minimize bandwidth.

A. Roadmap

Before we outline the roadmap for the design of Aion, we introduce a notion called *average transmission interval vector.*

[2]CDS generalizes PSC in [36] (for $K = 1$) to the general case of $K \geq 1$.

Definition 2: A vector $\boldsymbol{l} = [l_1\ l_1\ \cdots\ l_N]$ *is an average transmission interval vector if* l_i *is the average transmission interval for source* i, $i \in \{1, 2, \cdots, N\}$.

In other words, l_i represents the average length of the interval between two consecutive transmissions from source i. Obviously a scheduler that satisfies AoI constraints vector $\boldsymbol{d}$ will render a vector $\boldsymbol{l}$ that is no greater than $\boldsymbol{d}$.

Property 1: If a scheduler can meet the AoI constraint vector $\boldsymbol{d}$*, then the average transmission interval vector* $\boldsymbol{l}$ *under this scheduler must satisfy* $1 \le l_i \le d_i$ *for all* $i \in \{1, 2, \cdots, N\}$.

For example, when we use CDS to construct a scheduler to satisfy $\boldsymbol{d}$ that is consecutively divisible, the corresponding average transmission interval vector $\boldsymbol{l}$ is exactly $\boldsymbol{d}$, where each source i transmits exactly every d_i time slots. Note that the scheduler by CDS will have each l_i being an integer. But in general a scheduler that satisfies $\boldsymbol{d}$ may render an $\boldsymbol{l}$ with a fractional l_i, as illustrated in the following example.

Example 1. Consider four sources $\{A, B, C, D\}$ and $\boldsymbol{d} = [3\ 5\ 5\ 5]$, which is not consecutively divisible. So CDS cannot be applied here. But it is easy to verify that the following cyclic scheduler with a cycle length of 5 is optimal:

$$ABACD$$

which requires the minimum bandwidth of 1 to satisfy $\boldsymbol{d}$. The corresponding vector $\boldsymbol{l}$ achieved for this scheduler is $\boldsymbol{l} = [2.5\ 5\ 5\ 5]$. Here $l_A = 2.5$ is not an integer. ■

By allowing $\boldsymbol{l}$ to be fractional, we are ready to describe our design roadmap for Aion. Given an arbitrary AoI constraint vector $\boldsymbol{d}$, we can find a consecutively divisible vector $\boldsymbol{l}$ (likely fractional), with $l_i \le d_i$ for $i \in \{1, 2, \cdots, N\}$, whose scheduler (found by CDS), will satisfy the AoI constraint vector $\boldsymbol{d}$. But in Definition 1, for $\boldsymbol{l}$ to be consecutively divisible, each element in $\boldsymbol{l}$ must be an integer. This definition appears to be overly restrictive (as shown in Example 1). Therefore, in Section IV-B, we first generalize the definition of consecutively divisible to include fractional values. Then we present an algorithm (SUD) that can construct a scheduler to satisfy $\boldsymbol{l}$, which in turn also satisfies any $\boldsymbol{d}$ with $\boldsymbol{l} \le \boldsymbol{d}$.[3]

One question that we need to address is for an arbitrary AoI constraint vector $\boldsymbol{d}$, how to find an $\boldsymbol{l}$ (with $\boldsymbol{l} \le \boldsymbol{d}$) that requires the minimum (or a close-to-minimum) bandwidth K. We propose a novel mapping procedure to address this question in Section IV-C. Our proposed algorithm Aion combines the mapping procedure and SUD together. In Section IV-D, we present the theoretical performance guarantee of Aion.

B. When $\boldsymbol{l}$ is Fractional Consecutively Divisible

Definition 3: A vector $\boldsymbol{x} = [x_1\ x_2\ \cdots\ x_N]$ *is Fractional Consecutively Divisible (FCD) if* $x_i \ge 1$ *for each* $i \in \{1, 2, \cdots, N\}$ *and* $x_i / x_{i-1} \in \mathbb{Z}^+$ *for each* $i \in \{2, 3, \cdots, N\}$.

Comparing Definition 1 to Definition 3, each element in a consecutively divisible vector is required to be an integer, while that in a FCD vector can be fractional. For example, the vector $[2.5\ 5\ 5\ 5]$ is FCD, but not consecutively divisible.

[3]In this paper $\boldsymbol{l} \le \boldsymbol{d}$ denotes that $l_i \le d_i$ for all $i \in \{1, 2, \cdots, N\}$.

For an average transmission interval vector $\boldsymbol{l}$ that is FCD, we need an algorithm to construct a scheduler ϕ to satisfy $\boldsymbol{l}$. A natural question is whether we can simply extend CDS for this purpose. The answer is, unfortunately, no. The main challenge comes from CDS's intrinsic property – CDS is designed to have each source i transmit one sample exactly every l_i time slots to satisfy $\boldsymbol{l}$ when l_i's are all integers. But for a fractional l_i, it is not clear which time slot should be assigned to source i for its first transmission. Neither is clear how CDS should schedule subsequent transmissions. So there does not appear to exist a simple extension of CDS to satisfy a FCD $\boldsymbol{l}$.

Before we show how to design a scheduler ϕ to satisfy FCD $\boldsymbol{l}$, let's first find what bandwidth (denoted as $K_{\boldsymbol{l}}$) ϕ would use. The following lemma addresses this question.

Lemma 2: For any average transmission interval vector $\boldsymbol{l} = [l_1\ l_2\ \cdots\ l_N]$ *that is FCD, there exists a cyclic scheduler using a bandwidth of* $K_{\boldsymbol{l}} = \left\lceil \sum_{i=1}^{N} \frac{1}{l_i} \right\rceil$ *to satisfy* $\boldsymbol{l}$.

A proof of Lemma 2 is based on construction of such a scheduler that uses a bandwidth of $K_{\boldsymbol{l}}$ to satisfy $\boldsymbol{l}$. Algorithm 2 (SUD) is such an algorithm. In Lemma 2, it does not say whether $K_{\boldsymbol{l}}$ is the minimum required bandwidth. This question, or the general question of how much the gap is (between $K_{\boldsymbol{l}}$ and optimal K^*) will be discussed in Lemma 3.

SUD – Key Idea For a FCD vector $\boldsymbol{l}$, we can always scale it up by a factor, say a such that $a \cdot l_1$ is an integer. By doing so, the scaled-up vector, denoted as $\hat{\boldsymbol{l}} = [a \cdot l_1,\ a \cdot l_2,\ \cdots,\ a \cdot l_N]$, has all its elements being integers and is consecutively divisible. Hence it is possible to construct a cyclic scheduler $\hat{\phi}$ with a cycle length of $(a \cdot l_N)$ to satisfy $\hat{\boldsymbol{l}}$. Although it is tempting to use CDS to construct this scheduler $\hat{\phi}$ that satisfies $\hat{\boldsymbol{l}}$, this is not our ultimate goal. Our ultimate goal is to use $\hat{\phi}$ as a step stone to design a scheduler ϕ that satisfies $\boldsymbol{l}$. To ensure this *Scaling Up and Down* (SUD) idea indeed works, we need to take some extra care in our design of scheduler $\hat{\phi}$. Specifically, we must ensure that $\hat{\phi}$ meets the following three criteria:

(i) $\hat{\phi}$ must satisfy $\hat{\boldsymbol{l}}$;

(ii) When scaling down $\hat{\phi}$ to construct ϕ, ϕ must satisfy $\boldsymbol{l}$;

(iii) The bandwidth used by ϕ cannot exceed $K_{\boldsymbol{l}}$.

In this section, we present SUD that meets all criteria. First, we use the following example to illustrate how SUD works.

Example 2. Consider 6 sources $\{A, B, C, D, E, F\}$ and $\boldsymbol{l} = [1.5\ 1.5\ 1.5\ 4.5\ 4.5\ 9]$ that is FCD. The bandwidth of the cyclic scheduler ϕ that we want to construct is:

$$K_{\boldsymbol{l}} = \left\lceil \frac{1}{1.5} + \frac{1}{1.5} + \frac{1}{1.5} + \frac{1}{4.5} + \frac{1}{4.5} + \frac{1}{9} \right\rceil = 3.$$

Now we show how to construct such a scheduler ϕ with bandwidth $K_{\boldsymbol{l}}$ to satisfy $\boldsymbol{l}$.

First, we scale up $\boldsymbol{l}$ by a factor of $a = 2$ and get $\hat{\boldsymbol{l}} = [3\ 3\ 3\ 9\ 9\ 18]$. Now we construct a cyclic scheduler $\hat{\phi}$ to achieve $\hat{\boldsymbol{l}}$ with a cycle length of 18. Later, we will construct ϕ by scaling down the cycle length of $\hat{\phi}$ by a factor of $a = 2$ (i.e., with a cycle length of 9) while satisfying $\boldsymbol{l}$.

The iterations of SUD procedure is given in Fig. 2. In the initialization step, we lay out 18 columns (because the cycle length of $\hat{\phi}$ is 18), with each column corresponds to a time

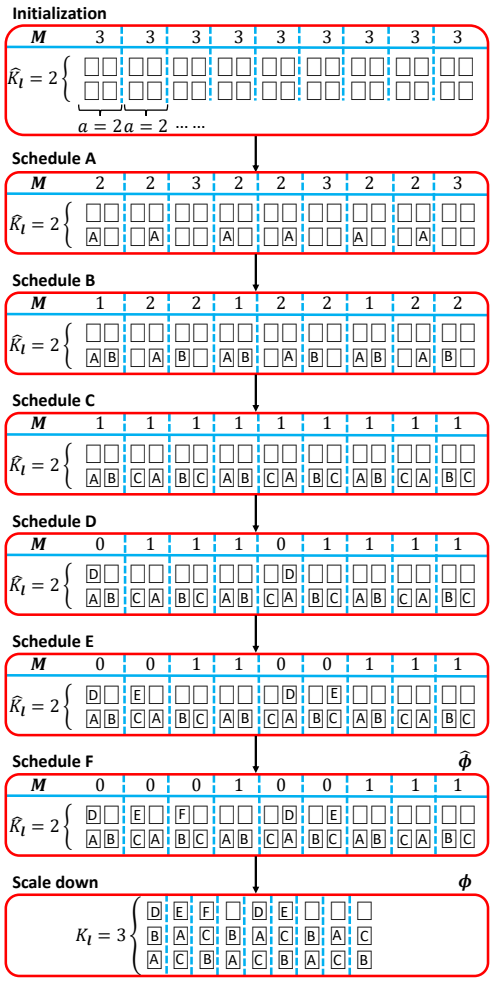

Fig. 2. Iterations of SUD to construct a cyclic scheduler ϕ to satisfy $\boldsymbol{l}$ that is FCD. We have 6 sources $\{A, B, C, D, E, F\}$ and $\boldsymbol{l} = [1.5\ 1.5\ 1.5\ 4.5\ 4.5\ 9]$.

slot for which we can make scheduling decisions in $\hat{\phi}$. For these 18 columns, we partition them to 9 groups, with each group consisting of 2 columns (as $a = 2$). This is because eventually we will scale down the cycle length of $\hat{\phi}$ by 2 to construct ϕ. In other words, scheduling decision in every 2 consecutive columns in $\hat{\phi}$ will be combined into 1 column in ϕ. The bandwidth of $\hat{\phi}$ is set to $\hat{K}_{\boldsymbol{l}} = 2$. This is because corresponding to $K_{\boldsymbol{l}} = 3$ bandwidth for ϕ, the bandwidth (rows) required by $\hat{\phi}$ is $\lceil 3/2 \rceil = 2$. Note that this still leaves us one more scheduling slot to $\hat{\phi}$ in *each group* than those to ϕ in each *time slot*. To ensure we do not have an overflow when we perform the scaling down step, we set a countdown counter M_i for each group $i \in \{1, 2, \cdots, 9\}$ in $\hat{\phi}$, which is initialized to $K_{\boldsymbol{l}}$, i.e., $M_i = 3$. Each time when we schedule a resource block in group i in $\hat{\phi}$ for transmission, the counter M_i is decremented by 1.

In general, $\hat{\phi}$ transmits each source i every $a \cdot l_i$ time slots. The question is which resource block is the first one that we assign to source i. When determining this first block, there are two requirements. (i) To ensure that $\hat{\phi}$ transmits source i every $a \cdot l_i$ time slots periodically, the first assigned block should be within the first $a \cdot l_i$ time slots. (ii) Note that after constructing $\hat{\phi}$ by N iterations, we want the number of assigned resource blocks in each group to be no greater than $K_{\boldsymbol{l}}$, i.e., we want $M_i \geq 0$ for all $M_i \in \boldsymbol{M}$ after N iterations. Hence when determining the first block, we want to balance the values of all $M_i \in \boldsymbol{M}$. Combining the two requirements, we find the largest element in $[M_1\ M_2\ \cdots\ M_{\lceil l_i \rceil}]$ to help figure out the first resource block which we assign to source i.

For source A, since $\hat{l}_A = 3$, we simply assign every 3 time slots to source A. And we have $\boldsymbol{M} = [2\ 2\ 3\ 2\ 2\ 3\ 2\ 2\ 3]$.

To schedule source B, since $\hat{l}_B = 3$, we need to assign a resource block to B every 3 time slots. Since $M_1 = 2$ is the largest element among M_1 and M_2, the first resource block is from the group 1 and can either come from time slot 1 or slot 2. Here from the purpose of balancing the number of assigned blocks over all time slots, we assign slot 2 to source B. Following the first assignment, we assign a resource block every 3 time slots to B and update the counter vector $\boldsymbol{M}$ for the 9 groups to be $\boldsymbol{M} = [1\ 2\ 2\ 1\ 2\ 2\ 1\ 2\ 2]$.

Similar to source B, for source C, the first resource block assigned is from the time slot 3. And following this first assigned block, we assign a resource block to C every 3 time slots. The updated $\boldsymbol{M}$ is now $\boldsymbol{M} = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$.

The scheduling for source D follows the same token as that for source C, except that we assign every 9 time slots to D since $\hat{l}_D = 9$. Time slots 1 and 10 are assigned to D and the $\boldsymbol{M}$ vector becomes $\boldsymbol{M} = [0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1]$. Similarly, time slots 3 and 12 are assigned to E and the $\boldsymbol{M}$ vector becomes $\boldsymbol{M} = [0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1]$. For source F, since $\hat{l}_F = 18$, we only assign a resource block in time slot 5 to F. The updated $\boldsymbol{M}$ vector is now $\boldsymbol{M} = [0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1]$.

Upon the completion of the last step, we have a scheduler $\hat{\phi}$ that satisfies $\hat{\boldsymbol{l}} = [3\ 3\ 3\ 9\ 9\ 18]$. To construct scheduler ϕ, we scale down the cycle length of $\hat{\phi}$ by a factor of 2. Specifically, we combine the resource blocks in each group (consisting of 2 consecutive columns) into 3 rows. For example, resource blocks in time slot 1 assigned to A and D and the resource block in time slot 2 assigned to B under $\hat{\phi}$ are combined into three resource blocks (3 rows) assigned to A, B, and D in time slot 1 under ϕ. It can be easily verified that ϕ achieves the original vector $\boldsymbol{l} = [1.5\ 1.5\ 1.5\ 4.5\ 4.5\ 9]$. ■

We formalize the ideas in the above example in Algorithm 2, which we call SUD. With SUD in hand, the proof of Lemma 2 can be easily constructed (by following the above example and the pseudocode in SUD). We omit the proof of the correctness of SUD here due to space limitation. To be succinct, (i) in Line 9 of SUD, we can find a slot within the first $a \cdot l_i$ slots of the cycle for scheduling source i, as the number of samples that can be transmitted within the first $a \cdot l_i$ slots when scheduling source i is no greater than $K_{\boldsymbol{l}} \cdot l_i$, and

$$K_{\boldsymbol{l}} \cdot l_i = \left\lceil \sum_{j=1}^{N} \frac{1}{l_j} \right\rceil \cdot l_i \geq \sum_{j=1}^{i} \frac{1}{l_j} \cdot l_i = 1 + \sum_{j=1}^{i-1} \frac{l_i}{l_j},$$

where l_i/l_j upper bounds the number of assigned samples for

Algorithm 2 Scaling Up and Down (SUD)

1: **input**: $\boldsymbol{l} = [l_1\ l_2\ \cdots\ l_N]$, and $\boldsymbol{l}$ is FCD
2: **output**: A cyclic scheduler ϕ and $K_{\boldsymbol{l}}$
3: **procedure**
4: Set cycle length of the scheduler $\hat{\phi}$ to be $a \cdot l_N$ where a is the smallest integer such that $a \cdot l_1 \in \mathbb{Z}^+$
5: Let $K_{\boldsymbol{l}} = \left\lceil \sum_{i=1}^{N} \frac{1}{l_i} \right\rceil$ and $\hat{K}_{\boldsymbol{l}} = \lceil \frac{K_{\boldsymbol{l}}}{a} \rceil$
6: Let $\boldsymbol{M} = [M_1\ M_2\ \cdots\ M_{\lceil l_N \rceil}]$ with each $M_j = K_{\boldsymbol{l}}$
7: **for** $i = 1, 2, \cdots, N$ **do**
8: Find the first largest element in the vector $[M_1\ M_2\ \cdots\ M_{\lceil l_i \rceil}]$, say M_r
9: Among the set of slots $\{(r-1)a+1, (r-1)a+2, \cdots, \min\{ra, al_i\}\}$, choose the first slot that currently carries the least number of samples, and assign it to source i; Let $M_r \leftarrow M_r - 1$
10: Following the slot assigned in Line 9, assign a slot to source i every $a \cdot l_i$ time slots along the cycle length of $\hat{\phi}$ until we reach the end of cycle length; Whenever we assign a slot, say slot k, to source i, let $M_{\lceil k/a \rceil} \leftarrow M_{\lceil k/a \rceil} - 1$
11: Construct ϕ by scaling $\hat{\phi}$ down by a times:
12: Set the cycle length of ϕ to be $\lceil l_N \rceil$
13: At each slot $i \in \{1, 2, \cdots, \lceil l_N \rceil\}$, ϕ transmits those samples scheduled for transmission by the set of slots $\{(i-1) \cdot a + 1, (i-1) \cdot a + 2, \cdots, (i-1) \cdot a + a\}$ in $\hat{\phi}$

source j ($j < i$); (ii) in Line 10 of SUD, by contradiction, we can prove that we can assign a slot to source i every $a \cdot l_i$ slots, considering that in Lines 9 and 10 we balance the number of assigned samples in the slots of a cycle.

It is easy to show that SUD has a time complexity of $O(a \cdot l_N \cdot N)$. Also, returning to Example 1, one can use SUD to construct a scheduler ϕ (given in that example) to satisfy $\boldsymbol{l} = [2.5\ 5\ 5\ 5]$ with a bandwidth of 1, which is minimum.

For SUD for each source i, as $\hat{\phi}$ transmits its sample exactly every $a \cdot l_i$ time slots, $\hat{\phi}$ satisfies an AoI constraint $a \cdot l_i$. Hence after scaling $\hat{\phi}$ down by a times, ϕ will achieve an average transmission interval l_i and satisfy an AoI constraint $\lceil l_i \rceil$ for each source i. Recall that in Section II, the AoI threshold d_i of each source i is an integer. Thus when SUD constructs the scheduler ϕ under an average transmission vector $\boldsymbol{l}$ that is FCD, ϕ will meet any AoI constraint vector $\boldsymbol{d}$ with $\boldsymbol{d} \geq \boldsymbol{l}$. In the following we further quantify the gap between the minimum required bandwidth K^* for $\boldsymbol{d}$ and $K_{\boldsymbol{l}}$ for $\boldsymbol{l}$.

Lemma 3: For any AoI constraint vector $\boldsymbol{d}$, if a FCD vector $\boldsymbol{l}$ satisfies $\boldsymbol{l} \leq \boldsymbol{d}$, then the bandwidth $K_{\boldsymbol{l}}$ required by ϕ (through SUD) must satisfy

$$K_{\boldsymbol{l}} \ \leq \ \left\lceil K^* \cdot \max_{i \in \{1,2,\cdots,N\}} \frac{d_i}{l_i} \right\rceil,$$

where K^* is the minimum bandwidth required to satisfy the AoI constraint vector $\boldsymbol{d}$.

Proof: It holds because of the following inequalities

$$\begin{aligned} K_{\boldsymbol{l}} \quad &= \left\lceil \sum_{i=1}^{N} \frac{1}{l_i} \right\rceil = \left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \cdot \frac{d_i}{l_i} \right\rceil \\ &\leq \left\lceil \max_{i \in \{1,2,\cdots,N\}} \frac{d_i}{l_i} \cdot \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil \leq \left\lceil \max_{i \in \{1,2,\cdots,N\}} \frac{d_i}{l_i} \cdot K^* \right\rceil \end{aligned}$$

where the last inequality holds due to Lemma 1. ∎

C. The Mapping Problem

In this section, we investigate the following mapping problem (denoted as MP): For an arbitrary AoI constraint vector $\boldsymbol{d}$, how to find a FCD vector $\boldsymbol{l}$ with $\boldsymbol{l} \leq \boldsymbol{d}$ such that the bandwidth ($K_{\boldsymbol{l}}$) required by $\boldsymbol{l}$ is minimized.

$$\begin{aligned} \textbf{MP} \quad \min_{l_1, l_2, \cdots, l_N \in \mathbb{R}} \quad & \left\lceil \sum_{i=1}^{N} \frac{1}{l_i} \right\rceil \\ \text{s.t.} \quad & 1 \leq l_i \leq d_i, \ \text{for } i \in \{1, 2, \cdots, N\}, \\ & \frac{l_i}{l_{i-1}} \in \mathbb{Z}^+, \ \text{for } i \in \{2, 3, \cdots, N\}, \\ & l_i \in \mathbb{R}^+, \ \text{for } i \in \{1, 2, \cdots, N\}, \end{aligned}$$

where $\mathbb{Z}^+$ denotes positive integers and $\mathbb{R}^+$ denotes positive real numbers. It is easy to verify that to solve MP, it is sufficient to solve the following problem (denoted as R-MP):

$$\begin{aligned} \textbf{R-MP} \quad \min_{l_1, l_2, \cdots, l_N \in \mathbb{R}} \quad & \sum_{i=1}^{N} \frac{1}{l_i} \\ \text{s.t.} \quad & 1 \leq l_i \leq d_i, \ \text{for } i \in \{1, 2, \cdots, N\}, \\ & \frac{l_i}{l_{i-1}} \in \mathbb{Z}^+, \ \text{for } i \in \{2, 3, \cdots, N\}, \\ & l_i \in \mathbb{R}^+, \ \text{for } i \in \{1, 2, \cdots, N\}. \end{aligned}$$

Property 2: If $\boldsymbol{l}^* = [l_1^*\ l_2^*\ \cdots\ l_N^*]$ is an optimal solution to R-MP, then there exists at least one $i \in \{1, 2, \cdots, N\}$ such that $l_i^* = d_i$.

Proof: Our proof is based on contradiction. Suppose there exists an optimal solution $\boldsymbol{l}^*$ to R-MP with $l_i^* < d_i$ for all $i \in \{1, 2, \cdots, N\}$. We will show that we can always construct a better solution $\tilde{\boldsymbol{l}}$ to R-MP (i.e., $\sum_{i=1}^{N}(1/\tilde{l}_i) < \sum_{i=1}^{N}(1/l_i^*)$) such that $\tilde{l}_i = d_i$ for some $i \in \{1, 2, \cdots, N\}$.

To construct such a $\tilde{\boldsymbol{l}}$, let us denote $\delta = \min_{1 \leq i \leq N}(d_i/l_i^*)$. Since $l_i^* \neq d_i$ and $1 \leq l_i^* \leq d_i$ for all $i \in \{1, 2, \cdots, N\}$, we must have $l_j^* < d_i$. Then $\delta > 1$ and we can construct $\tilde{\boldsymbol{l}} = [\tilde{l}_1\ \tilde{l}_2\ \cdots\ \tilde{l}_N]$ by letting $\tilde{l}_i = \delta \cdot l_i^*$ for all i. Clearly, $\tilde{\boldsymbol{l}}$ remains FCD and is feasible to R-MP. Further $\sum_{i=1}^{N}(1/\tilde{l}_i) < \sum_{i=1}^{N}(1/l_i^*)$ and this completes our proof. ∎

Reducing Candidate Sets (RCS) In essence, Property 2 says that comparing $\boldsymbol{l}^*$ and $\boldsymbol{d}$ along its element index, there is at least one index on which the corresponding elements in both vectors are equal. This property can be exploited to narrow down the search space for each l_i^*, $i \in \{1, 2, \cdots, N\}$.

Example 3. Consider 3 sources $\{A, B, C\}$ with an AoI constraint vector $\boldsymbol{d} = [2\ 3\ 6]$. According to Property 2, for

the optimal solution $\boldsymbol{l}^*$ to R-MP, we must have either (i) $l_A^* = d_A = 2$, or (ii) $l_B^* = d_B = 3$, or (iii) $l_C^* = d_C = 6$.

Denote $\mathcal{L}_A$, $\mathcal{L}_B$, and $\mathcal{L}_C$ as the set of values that l_A^*, l_B^*, and l_C^* may take, respectively. Initially, $\mathcal{L}_A = \mathcal{L}_B = \mathcal{L}_C = \emptyset$. Case (i): $l_A^* = d_A = 2$. For the source A, clearly we need to put the value 2 to $\mathcal{L}_A$.

For the source B, due to the FCD requirement, the value of l_B^* must be an integer multiple of $l_A^* = 2$. Further, we must have $1 \le l_B^* \le d_B = 3$. So l_B^* can only take the value of 2 and we add 2 into $\mathcal{L}_B$.

For source C, due to the FCD requirement, the value of l_C^* must be an integer multiple of $l_B^* = 2$. Further, we must have $1 \le l_C^* \le d_C = 6$. So l_C^* can only take the values of 2, 4, or 6 and we add these 3 values into $\mathcal{L}_C$. Now we have:

$$\mathcal{L}_A = \{2\},\ \mathcal{L}_B = \{2\},\ \mathcal{L}_C = \{2,\ 4,\ 6\}.$$

Case (ii): $l_B^* = d_B = 3$. For source B, we put 3 to $\mathcal{L}_B$.

For source A, due to the FCD requirement, the value of l_A^* must be $l_B^* = 3$ divided by an integer. Further, we must have $1 \le l_A^* \le d_A = 2$. So l_A^* can only take the values of 1 or 1.5 (when l_B^* is divided by 3 or 2). We add both values into $\mathcal{L}_A$.

For source C, due to the FCD requirement, the value of l_C^* must be an integer multiple of $l_B^* = 3$. Further, we must have $1 \le l_C^* \le d_C = 6$. So l_C^* can only take the values of 3 or 6, and we add both into $\mathcal{L}_C$. Since 6 is already in $\mathcal{L}_C$, it does not need to be added again. Now we have:

$$\mathcal{L}_A = \{1,\ 1.5,\ 2\},\ \mathcal{L}_B = \{2,\ 3\},\ \mathcal{L}_C = \{2,\ 3,\ 4,\ 6\}.$$

Case (iii): $l_C^* = d_C = 6$. For the source C, we put the value 6 to $\mathcal{L}_C$. Since $\mathcal{L}_C$ already has an element with value 6, we don't need to put it in again.

For source B, due to the FCD requirement, l_B^* must be $l_C^* = 6$ divided by an integer. Further, we have $1 \le l_B^* \le d_B = 3$. So l_B^* can only take values of 1, 1.2, 1.5, 2, or 3. Since 2 and 3 are already in $\mathcal{L}_B$, we add 1, 1.2, and 1.5 into $\mathcal{L}_B$.

For source A, due to the FCD requirement, the value of l_A^* must be l_B^* divided by an integer. Considering that in Case (iii), l_B^* can only take the values of 1, 1.2, 1.5, 2, or 3, and we require $1 \le l_A^* \le d_A = 2$, we find that l_A^* can only take the values of 1, 1.2, 1.5, or 2. Since 1, 1.5, and 2 are already in $\mathcal{L}_A$, we only add 1.2 into $\mathcal{L}_A$. Now we have:

$$\mathcal{L}_A = \{1,\ 1.2,\ 1.5,\ 2\},\ \mathcal{L}_B = \{1,\ 1.2,\ 1.5,\ 2,\ 3\},$$
$$\mathcal{L}_C = \{2,\ 3,\ 4,\ 6\}.$$

Note that our analysis (based on Property 2) has drastically narrowed down the search space for l_A^*, l_B^*, and l_C^*. ■

In general, denote $\mathcal{L}_i$ as the "reduced" set of possible values that l_i^* can take in an optimal solution $\boldsymbol{l}^*$. Then following the ideas in the above example, Algorithm 3 (code-named RCS) find the set $\mathcal{L}_i$ for each $i \in \{1, 2, \cdots, N\}$. The key ideas are: for each case of $l_i^* = d_i$, exploit the FCD requirement and the range of each l_j (i.e., $1 \le l_j \le d_j$) to determine a reduced set of values for each $\mathcal{L}_i$, $i \in \{1, 2, \cdots, N\}$. It is easy to verify that RCS has a time complexity of $O(N^2 \cdot d_N)$.

Algorithm 3 Reducing Candidate Sets (RCS)

input: $\boldsymbol{d} = [d_1\ d_2\ \cdots\ d_N]$
output: $\mathcal{L}_1,\ \mathcal{L}_2,\ \cdots,\ \mathcal{L}_N$
procedure
  Set $\mathcal{L}_i = \emptyset$ for all $i \in \{1, 2, \cdots, N\}$
  for each $i = 1, 2, \cdots, N$ **do**
    $\mathcal{I}_i = \{d_i\}$, $\mathcal{I}_j = \emptyset$ for all $j \in \{1, \cdots, N\}$, $j \ne i$
    for each $j = i-1, i-2, \cdots, 1$ **do**
      Find all real numbers that are lower bounded by 1, upper bounded by d_j, and are divisible of at least one element in $\mathcal{I}_{j+1}$ by an integer. Add these real numbers into $\mathcal{I}_j$
    for each $j = i+1, i+2, \cdots, N$ **do**
      Find all real numbers that are lower bounded by 1, upper bounded by d_j, and are integer multiples of at least one element in $\mathcal{I}_{j-1}$. Add these real numbers into $\mathcal{I}_j$
    for each $j = 1, 2, \cdots, N$ **do**
      Add each element of $\mathcal{I}_j$ to $\mathcal{L}_j$ if it is not in $\mathcal{L}_j$

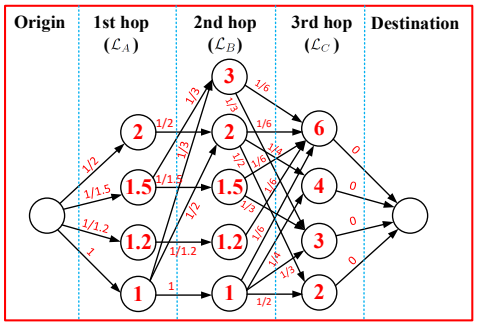

Fig. 3. Construction of a directed graph with $\mathcal{L}_A = \{1,\ 1.2,\ 1.5,\ 2\}$, $\mathcal{L}_B = \{1,\ 1.2,\ 1.5,\ 2,\ 3\}$, and $\mathcal{L}_C = \{2,\ 3,\ 4,\ 6\}$.

Finding Optimal Mapping Now we have a reduced set of possible values $\mathcal{L}_i$ for each l_i^*. The problem to solve now is to pick one value from each set so that the vector $\boldsymbol{l}$ constructed by these values is both FCD and that $\sum_{i=1}^{N}(1/l_i)$ is minimized. We show that this problem is equivalent to finding the shortest path in a directed network graph.

Example 4. Continuing with Example 4, we have: $\mathcal{L}_A = \{1,\ 1.2,\ 1.5,\ 2\}$, $\mathcal{L}_B = \{1,\ 1.2,\ 1.5,\ 2,\ 3\}$, and $\mathcal{L}_C = \{2,\ 3,\ 4,\ 6\}$. We can construct a direct graph as shown in Fig. 3. In this figure, we put an origin node and destination node to denote the start and finish of our problem. Then we put down the nodes for each of the three hops from original to destination, corresponding to different values from sets $\mathcal{L}_A$, $\mathcal{L}_B$, and $\mathcal{L}_C$. That is, we have 4 nodes for the first hop (corresponding to the values in $\mathcal{L}_A$), 5 nodes for the second hop (corresponding to the values in $\mathcal{L}_B$), and 4 nodes for the third hop (corresponding to the values in $\mathcal{L}_C$).

Now we add directed links to the graph, along with their costs. From the origin node, we can have 4 directed links to the 4 nodes in the first hop, with the cost of each link being

Algorithm 4 Aion – The Complete Algorithm

1: **input**: $\boldsymbol{d} = [d_1\ d_2\ \cdots\ d_N]$
2: **output**: A cyclic scheduler ϕ and $K_{\boldsymbol{l}^*}$
3: **procedure**
4: Run RCS to find $\mathcal{L}_i$ for all $i \in \{1, 2, \cdots, N\}$
5: Use $\mathcal{L}_i$'s to construct a direct graph and use shortest path routing to find optimal mapping $\boldsymbol{l}^*$
6: Run SUD with input $\boldsymbol{l}^*$ to find scheduler ϕ and $K_{\boldsymbol{l}^*}$

$1/l_A^*$, $l_A^* \in \{1, 1.2, 1.5, 2\} = \mathcal{L}_A$, as shown in the figure. For the second hop, a link from one of the 4 nodes in $\mathcal{L}_A$ to one of the 5 nodes in $\mathcal{L}_B$ exists only if it meets the FCD requirement. That is, a link from a node in $\mathcal{L}_A$ to a node in $\mathcal{L}_B$ exists only if l_B^*/l_A^* is a positive integer. For example, a link from node 2 in $\mathcal{L}_A$ to node 2 in $\mathcal{L}_B$ exists because $2/2 = 1$. Node 1.5 in $\mathcal{L}_A$ has a link to nodes 1.5 and 3 in $\mathcal{L}_B$ because $1.5/1.5 = 1$ and $3/1.5 = 2$. After we find the 7 links from the 4 nodes in $\mathcal{L}_A$ to the 5 nodes in $\mathcal{L}_B$, we mark the cost of each link in the figure, which is $1/l_B^*$, $l_B^* \in \{1, 1.2, 1.5, 2, 3\} = \mathcal{L}_B$. We repeat the same process to add the links for the third hop (from the 5 nodes in $\mathcal{L}_B$ to the 4 nodes in $\mathcal{L}_C$). Finally, for the 4 nodes in $\mathcal{L}_C$, each has a directed link to the destination node, with an artificial link cost of 0.

Based on this directed graph, it is easy to see that to minimize $\sum_{i=1}^{N}(1/l_i)$ is equivalent to finding a least cost path from the origin to the destination. For this example, there are two shortest paths: (origin $\rightarrow 2 \rightarrow 2 \rightarrow 6 \rightarrow$ destination) and (origin $\rightarrow 1.5 \rightarrow 3 \rightarrow 6 \rightarrow$ destination), both of which have a total cost of $7/6$. Hence, the optimal vector $\boldsymbol{l}^*$ can be either [2 2 6] or [1.5 3 6], both of which are FCD. ■

In general, we can always use the key idea in the above example to construct a directed graph from a fictitious origin node to a fictitious destination node. The set of nodes in the i-th hop ($i \in \{1, 2, \cdots, N\}$) corresponds to the elements in $\mathcal{L}_i$ and the links from an $(i-1)$-th hop node to an i-th hop node are determined by the FCD requirement. Further, the link cost is the reciprocal of the value of the corresponding i-th hop node. Subsequently, any shortest path routing algorithm (e.g., Dijkstra's algorithm) can be used to find the optimal solution.

D. Aion: The Complete Algorithm and Performance Analysis

Main Algorithm Combining results in Sections IV-B and IV-C, Algorithm 4 shows the complete procedure for Aion on how to find a scheduler for an arbitrary AoI constraint vector $\boldsymbol{d}$.

Performance The performance gap between the required minimum bandwidth $K_{\boldsymbol{l}^*}$ by Aion and that by the optimal (unknown) K^* is given in the following theorem.

Theorem 2: To guarantee an arbitrary AoI vector $\boldsymbol{d} = [d_1\ d_2\ \cdots\ d_N]$, the required bandwidth $K_{\boldsymbol{l}^*}$ for the scheduler ϕ that is constructed by Aion (in Algorithm 4) satisfies:

$$\frac{K_{\boldsymbol{l}^*}}{K^*} \leq \frac{\left\lceil \sum_{i=1}^{N} \frac{1}{l_i^*} \right\rceil}{\left\lceil \sum_{i=1}^{N} \frac{1}{d_i} \right\rceil},$$

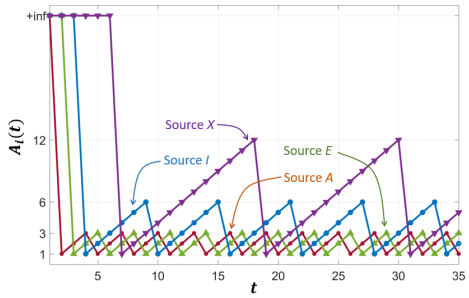

Fig. 4. Behavior of $A_i(t)$ for sources $i = A,\ E,\ I,\ X$.

where K^* is the minimum bandwidth to satisfy $\boldsymbol{d}$.

Proof: It holds directly due to Lemmas 1 and 2. ■

Theorem 2 suggests that the theoretical gap between the bandwidth $K_{\boldsymbol{l}^*}$ required by Aion and the minimum K^* heavily depends on the mapping between $\boldsymbol{l}^*$ and $\boldsymbol{d}$. For the special case when $\boldsymbol{d}$ is consecutively divisible, $\boldsymbol{l}^*$ obtained by Aion is identical to $\boldsymbol{d}$ and hence $K_{\boldsymbol{l}^*}$ is optimal. This special case result is consistent to that of CDS.

Computational Complexity Aion has a time complexity of $O(N^4 \cdot (d_N)^2)$: In line 4, the time complexity of running RCS to find $\mathcal{L}_i$, $i \in \{1, 2, \cdots, N\}$ is $O(N^2 \cdot d_N)$. In line 5, the time complexity of constructing the directed graph is $O(N^4 \cdot (d_N)^2)$. Also in line 5, the time complexity of using the shortest path routing to find $\boldsymbol{l}^*$ is $O(N^4 \cdot (d_N)^2)$. In line 6, running SUD has a time complexity of $O(N \cdot (d_N)^2)$.

V. Numerical Results

In this section we evaluate Aion. First, we use a case study to demonstrate that the scheduler ϕ constructed by Aion can indeed guarantee the given AoI constraint vector $\boldsymbol{d}$. Then we study the bandwidth $K_{\boldsymbol{l}^*}$ required by Aion, and compare it against the lower bound and upper bound derived for K^* in Lemma 1. We further study how the bandwidth achieved by Aion is affected by the number of sources N.

A Case Study Let's consider 25 sources $\{A, B \cdots, X, Y\}$ with the following AoI constraint vector $\boldsymbol{d}$:

[3 3 3 4 5 5 6 6 8 8 8 8 9 9 9 9 9 9 12 12 14 15 15 15 16].

Following the Aion algorithm, we first find the following FCD vector $\boldsymbol{l}^*$ for $\boldsymbol{d}$ as follows:

[3 3 3 3 3 3 6 6 6 6 6 6 6 6 6 6 6 6 12 12 12 12 12 12 12].

Then following Aion, we use SUD to construct scheduler ϕ with a cycle length of $L = 12$ and a bandwidth $K_{\boldsymbol{l}^*} = 5$:

Scheduler ϕ $K_{\boldsymbol{l}^*} = 5$	S	T	U	V	W	X	Y	□	□	□	□	□
	M	N	O	P	Q	R	M	N	O	P	Q	R
	G	H	I	J	K	L	G	H	I	J	K	L
	D	E	F	D	E	F	D	E	F	D	E	F
	A	B	C	A	B	C	A	B	C	A	B	C
Time slot t	1	2	3	4	5	6	7	8	9	10	11	12

It is easy to verify that ϕ satisfies $\boldsymbol{d}$. In Fig. 4 we present the time evolution of $A_i(t)$ for the sources. For legibility, we only show $A_i(t)$ for sources $A,\ E,\ I,\ X$, respectively; $A_i(t)$'s for

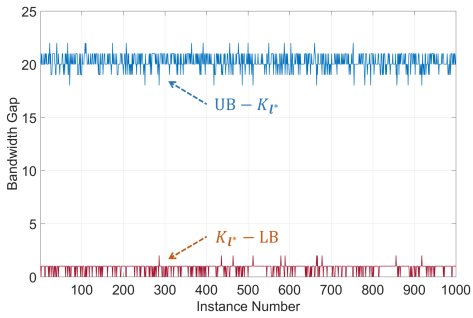

Fig. 5. The gaps between K_{l^*} by Aion and the lower and upper bounds of K^*, respectively, over 1000 instances when $N = 25$.

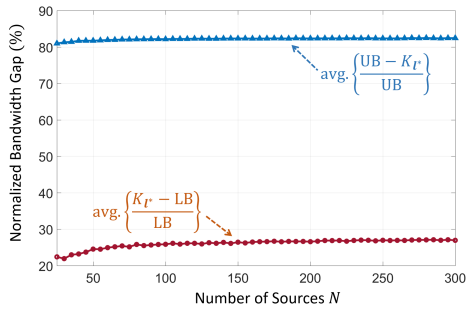

Fig. 6. Average of normalized gaps $(K_{l^*} - LB)/LB$ and $(UB - K_{l^*})/UB$ when N increases from 25 to 300.

the other sources are similar. As described in Section II, we assume the system is turned on at $t = 1$ and there is no sample stored at the BS at $t = 1$. For source A, at $t = 1$, since the BS has not yet received any sample from A, $A_A(1) = +\infty$. At $t = 2$, $A_A(2) = 1$, as A transmits a packet in time slot 1. At $t = 3$, $A_A(3) = 2$ and at $t = 4$, $A_A(4) = 3$. At $t = 5$, $A_A(5) = 1$ again, because in time 4, A transmit a packet to the BS. Based on ϕ, we will have the time evolution of $A_i(t)$ for sources E, I, X respectively.

From Lemma 1, we have $4 \leq K^* \leq 25$. So the $K_{l^*} = 5$ we have from Aion is at most 1 unit more than the lower bound of K^*. This shows that K_{l^*} is either optimal or only one unit away from optimal. In the following set of results, we further substantiate the performance of K_{l^*} against the lower and upper bounds of K^* in Lemma 1.

More Results on K_{l^*} We show more results for K_{l^*} and compare it to the lower and upper bounds of K^*. For $N = 25$, we generate $1,000$ instances for AoI constraint vector $\boldsymbol{d}$. For each $\boldsymbol{d}$, each of its elements (d_i's) is randomly generated by following a uniform distribution from $[2, 20]$. For each of these 1,000 instances, we use Aion to find a scheduler ϕ and compare K_{l^*} to the lower and upper bounds of K^*. Fig. 5 shows complete results for the gap between K_{l^*} and lower bound of K^* (i.e., $K_{l^*} - \text{LB}$) and the gap between K_{l^*} and upper bound of K^* (i.e., $\text{UB} - K_{l^*}$) for all 1,000 instances. Clearly, in all cases, K_{l^*} is either optimal or near-optimal. On average (over 1,000 instances), the gap between K_{l^*} and the lower bound of K^* is 0.83 (less than 1 unit of bandwidth).

Varying Number of Sources N Now let's examine the performance of K_{l^*} by Aion under different N – the number of sources. We vary N from 25 to 300, with an incremental step of 5. Again, we generate $1,000$ instances of $\boldsymbol{d}$'s for each N and for each instance, we use a uniform distribution between $[2, 20]$ to generate each d_i.

When N increases from 25 to 300, it is obvious that the absolute values for $(K_{l^*} - \text{LB})$ and $(\text{UB} - K_{l^*})$ will increase. So it makes more sense and fair to examine the normalized values of these two gaps w.r.t. LB and UB, respectively. In Fig. 6, we plot the average normalized gaps, i.e., avg.$(K_{l^*} - \text{LB})/\text{LB}$ and avg.$(\text{UB} - K_{l^*})/\text{UB}$ over 1,000 instances for each N. In this figure, we observe that although there is a small increase of avg.$(K_{l^*} - \text{LB})/\text{LB}$ when N increases from 25 to 300, the average of this normalized gap w.r.t. LB is rather steady and is no more than 27%. Given that LB is the lower bound of K^*, we can expect K_{l^*} is even closer to K^*, which lies between K_{l^*} and LB.

VI. Summary and Future Work

We study a bandwidth minimization problem with AoI constraints. It is challenging due to its large search space and discrete nature of time-slot based scheduling. We develop a novel fast algorithm called Aion that can always construct a scheduler to guarantee any given AoI constraint vector $\boldsymbol{d}$. Our theoretical performance guarantee of minimizing bandwidth hinger upon a novel concept called *Fractional Consecutively Divisible* (FCD) for the given vector $\boldsymbol{d}$. Specifically, when $\boldsymbol{d}$ is FCD, Aion always requires the minimum bandwidth K^*; when $\boldsymbol{d}$ is not FCD, the bandwidth required by Aion is guaranteed to be within a small factor of the minimum. The design of Aion involves many innovative ideas, such as transforming a mapping problem (from a non-FCD $\boldsymbol{d}$ to a FCD $\boldsymbol{l}^*$) into a shortest path problem, an algorithm to reduce the search space of mapping by exploiting an intrinsic property that relates $\boldsymbol{d}$ to $\boldsymbol{l}^*$, and a scaling-based procedure to handle fractional $\boldsymbol{l}^*$ in the construction of a discrete time-slot based scheduler. We validate the performance of Aion through a large number of simulations and all results confirm our theoretical findings.

The results from this paper lay the foundation for future research in the important area of bandwidth minimization with AoI guarantee. Many interesting problems need to be further explored. For instance, in this paper, we assumed that the required bandwidth K is in direct proportion to achievable bit rate. This may not be the case when the channel is fast fading, where the instantaneous bit rate can vary quickly with channel conditions. So it would be interesting to explore what optimal bandwidth is needed to guarantee AoI in a statistical sense, e.g., in the form of a chance constraint [40], [41].

Acknowledgment

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. of IEEE SECON*, Salt Lake City, UT, USA, June 27–30, 2011, pp. 350–358.

[2] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. of IEEE INFOCOM*, Orlando, FL, USA, March 25–30, 2012, pp. 2731–2735.

[3] Y. Sun, "A collection of recent papers on the age of information," 2019, available at http://webhome.auburn.edu/~yzs0078/.

[4] A. M. Bedewy, Y. Sun, S. Kompella, and N. B. Shroff, "Age-optimal sampling and transmission scheduling in multi-source systems," in *Proc. of ACM MobiHoc*, Catania, Italy, July 2–5, 2019, pp. 121–130.

[5] I. Kadota and E. Modiano, "Minimizing the age of information in wireless networks with stochastic arrivals," in *Proc. of ACM MobiHoc*, Catania, Italy, July 2–5, 2019.

[6] B. Yin, S. Zhang, Y. Cheng, L. X. Cai, Z. Jiang, S. Zhou, and Z. Niu, "Only those requested count: Proactive scheduling policies for minimizing effective age-of-information," in *Proc. of IEEE INFOCOM*, Paris, France, April 29–May 2, 2019, pp. 109–117.

[7] Y.-P. Hsu, "Age of information: Whittle index for scheduling stochastic arrivals," in *Proc. of IEEE ISIT*, Vail, Colorado, USA, June 17–22, 2018, pp. 2634–2638.

[8] C. Li, S. Li, and Y. T. Hou, "A general model for minimizing age of information at network edge," in *Proc. of IEEE INFOCOM*, Paris, France, April 29–May 2, 2019, pp. 118–126.

[9] C. Li, Y. Huang, Y. Chen, B. Jalaian, Y. T. Hou, and W. Lou, "Kronos: A 5G scheduler for AoI minimization under dynamic channel conditions," in *Proc. of IEEE ICDCS*, Dallas, TX, USA, July 7–9, 2019, pp. 1466–1475.

[10] B. T. Bacinoglu and E. Uysal-Biyikoglu, "Scheduling status updates to minimize age of information with an energy harvesting sensor," in *Proc. of IEEE ISIT*, Aachen, Germany, June 25–30, 2017, pp. 1122–1126.

[11] Z. Zhou, C. Fu, C. J. Xue, and S. Han, "Transmit or discard: Optimizing data freshness in networked embedded systems with energy harvesting sources," in *Proc. of Design Automation Conference (DAC)*, Las Vegas, NV, USA, June 3–5, 2019.

[12] S. Feng and J. Yang, "Minimizing age of information for an energy harvesting source with updating failures," in *Proc. of IEEE ISIT*, Vail, Colorado, USA, June 17–22, 2018, pp. 2431–2435.

[13] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor, "Age-minimal transmission for energy harvesting sensors with finite batteries: Online policies," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 534–556, 2019.

[14] R. D. Yates, "Lazy is timely: Status updates by an energy harvesting source," in *Proc. of IEEE ISIT*, Hong Kong, China, June 14–19, 2015, pp. 3008–3012.

[15] I. Kadota, A. Sinha, and E. Modiano, "Optimizing age of information in wireless networks with throughput constraints," in *Proc. of IEEE INFOCOM*, Honolulu, HI, USA, April 15–19, 2018, pp. 1844–1852.

[16] N. Lu, B. Ji, and B. Li, "Age-based scheduling: Improving data freshness for wireless real-time traffic," in *Proc. of ACM MobiHoc*, Los Angeles, CA, USA, June 25–28, 2018, pp. 191–200.

[17] Q. Liu, H. Zeng, and M. Chen, "Minimizing age-of-information with throughput requirements in multi-path network communication," in *Proc. of ACM MobiHoc*, Catania, Italy, July 2–5, 2019, pp. 41–50.

[18] I. Kadota, A. Sinha, and E. Modiano, "Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1359–1372, 2019.

[19] 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. of IEEE MILCOM*, Baltimore, MD, USA, October 23–25, 2017, pp. 436–441.

[20] 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 Transactions on Wireless Communications*, vol. 18, no. 8, pp. 4129–4142, 2019.

[21] V. Tripathi and E. Modiano, "A whittle index approach to minimizing functions of age of information," in *Proc. of IEEE Allerton*, Monticello, IL, USA, September 24–27, 2019, pp. 1160–1167.

[22] R. Talak and E. Modiano, "Age-delay tradeoffs in single server systems," in *Proc. of IEEE ISIT*, Paris, France, July 7–12, 2019, pp. 340–344.

[23] B. Li, A. Eryilmaz, and R. Srikant, "On the universality of age-based scheduling in wireless networks," in *Proc. of IEEE INFOCOM*, Hong Kong, China, April 26–May 1, 2015, pp. 1302–1310.

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

[25] 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. of IEEE INFOCOM*, Honolulu, HI, USA, April 15–19, 2018, pp. 908–916.

[26] Z. Ning, P. Dong, X. Wang, X. Hu, L. Guo, B. Hu, Y. Guo, T. Qiu, and R. Kwok, "Mobile edge computing enabled 5G health monitoring for Internet of medical things: A decentralized game theoretic approach," *IEEE Journal on Selected Areas in Communications*, 2020.

[27] G. D. Nguyen, S. Kompella, C. Kam, J. E. Wieselthier, and A. Ephremides, "Impact of hostile interference on information freshness: A game approach," in *Proc. of WiOpt*, Paris, France, May 15–19, 2017.

[28] Y. Xiao and Y. Sun, "A dynamic jamming game for real-time status updates," in *Proc. of IEEE INFOCOM AoI WKSHPS*, Honolulu, HI, USA, April 15–19, 2018, pp. 354–360.

[29] J. Zhong, R. D. Yates, and E. Soljanin, "Backlog-adaptive compression: Age of information," in *Proc. of IEEE ISIT*, Aachen, Germany, June 25–30, 2017, pp. 566–570.

[30] P. Mayekar, P. Parag, and H. Tyagi, "Optimal source codes for timely updates," *IEEE Transactions on Information Theory*, vol. 66, no. 6, pp. 3714–3731, 2020.

[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. of IEEE ISIT*, Vail, Colorado, USA, June 17-22, 2018, pp. 2471–2475.

[32] P. Mayekar, P. Parag, and H. Tyagi, "Optimal lossless source codes for timely updates," in *Proc. of IEEE ISIT*, Vail, Colorado, USA, June 17–22, 2018, pp. 1246–1250.

[33] S. Ioannidis, A. Chaintreau, and L. Massoulié, "Optimal and scalable distribution of content updates over a mobile social network," in *Proc. of IEEE INFOCOM*, Rio de Janeiro, Brazil, April 19–25, 2009, pp. 1422–1430.

[34] C. Kam, S. Kompella, and A. Ephremides, "Experimental evaluation of the age of information via emulation," in *Proc. of IEEE MILCOM*, Tampa, FL, USA, October 26–28, 2015, pp. 1070–1075.

[35] V. Marbukh, "Towards managing age of network state information in challenged networks," in *Proc. of IEEE INFOCOM AoI WKSHPS*, Honolulu, HI, USA, April 15–19, 2018.

[36] C. Li, S. Li, Y. Chen, Y. T. Hou, and W. Lou, "AoI scheduling with maximum thresholds," in *Proc. of IEEE INFOCOM*, Virtual Conference, July 6–9, 2020.

[37] Q. He, G. Dán, and V. Fodor, "On emptying a wireless network with minimum-energy under age constraints," in *Proc. of IEEE INFOCOM AoI WKSHPS*, Paris, France, April 29–May 2, 2019, pp. 668–673.

[38] R. Talak, S. Karaman, and E. Modiano, "Optimizing information freshness in wireless networks under general interference constraints," in *Proc. of ACM MobiHoc*, Los Angeles, CA, USA, June 25–28, 2018.

[39] ——, "Optimizing age of information in wireless networks with perfect channel state information," in *Proc. of WiOpt*, Shanghai, China, May 7–11, 2018.

[40] S. Li, Y. Huang, C. Li, B. A. Jalaian, Y. T. Hou, and W. Lou, "Coping uncertainty in coexistence via exploitation of interference threshold violation," in *Proc. of ACM MobiHoc*, Catania, Italy, July 2–5, 2019, pp. 71–80.

[41] S. Li, Y. T. Hou, W. Lou, B. Jalaian, S. Russell, and B. MacCall, "Optimal power control with channel uncertainty in ad hoc networks," in *Proc. of IEEE MILCOM*, Norfolk, VA, USA, November 12–14, 2019, pp. 652–657.