Age of Information with Unreliable Transmissions in Multi-Source Multi-Hop Status Update Systems

Shahab Farazi[†], Andrew G. Klein[‡] and D. Richard Brown III[†]
[†]Worcester Polytechnic Institute, 100 Institute Rd., Worcester, MA 01609, Email: {sfarazi,drb}@wpi.edu
[‡]Western Washington University, 516 High St., Bellingham, WA 98225, Email: andy.klein@wwu.edu

***Abstract*—This paper studies the "age of information" (AoI) in multi-hop networks with time-slotted transmissions and packet loss in a setting where each node is both a source and monitor of information. Nodes take turns broadcasting their information to other nodes while also maintaining tables of updates for the information received from other nodes. It is assumed that transmission errors in the network occur with a fixed error probability and that transmission errors are independent across links. Using tools from graph theory, two algorithms are developed based on sequential flooding with repetitive transmissions when transmission errors occur. These algorithms generate status update dissemination schedules for any network with a connected topology. The two algorithms differ in terms of whether the root nodes in each sequential flooding tree resample their local information when transmission errors occur. A lower bound on the average peak AoI as a function of fundamental graph properties is also derived for schedules generated by the algorithm without resampling by the root nodes. Numerical results are presented to evaluate the achieved average peak AoI for some canonical graph topologies.**

***Index Terms*—Age of information, multi-source, multi-hop, packetized communications, transmission error.**

I. Introduction

Information freshness is of critical importance in a variety of networked monitoring and control systems such as intelligent vehicular systems, channel state feedback, and environmental monitoring as well as applications such as financial trading and online learning. A recent line of research has considered information freshness from a fundamental perspective under an *Age of Information* (AoI) metric first proposed in [1] and further studied in [2]–[19]. The main idea is that there are one or more sources of information along with one or more monitors. A source generates timestamped status updates which are received by a monitor after some delay. The AoI is defined as the difference between the current time and the timestamp of the most recent status update at the monitor.

A recent line of work has considered the effect of packet delivery errors on AoI in single-source, single-hop systems [20]–[22]. Multi-source and/or multi-monitor extensions, also in the single-hop context, were studied in [23]–[26]. Despite the general interference constraints, the setting in all of these studies assumes that all information flows are single-hop, i.e., every source is directly connected to a corresponding monitor. Hence there is a gap in understanding AoI in multi-source, multi-hop systems with packet transmission errors, where some information packets need to be relayed multiple times to reach the monitor/destination.

This work was supported by National Science Foundation awards CNS-1836690 and CNS-1836695.

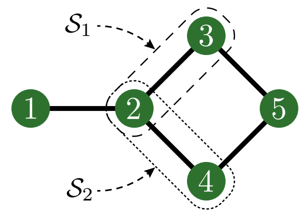

Fig. 1. A 5-node pan network. The nodes are indexed by $\mathcal{V} = \{1, 2, 3, 4, 5\}$. Here there exist two MCDS's, $\mathcal{S}_1 = \{2, 3\}$ and $\mathcal{S}_2 = \{2, 4\}$, so that $\mathcal{L} = \mathcal{V} - (\mathcal{S}_1 \cup \mathcal{S}_2) = \{1, 5\}$ is the set of pseudo-leaf nodes.

This paper studies AoI in a general multi-source, multi-hop, time-slotted network setting with *explicit contention* in the sense that all delays between sources and monitors are due to explicit channel uses by other nodes in the network. Each node in the system is both a source and monitor of information. Since the only assumption on the network is that it is connected, some nodes in the network also serve as relays to facilitate multi-hop dissemination of information between nodes that are not directly connected. Using a graph theoretical approach, this paper builds on our prior results in [17]–[19] by generalizing our analysis to account for packet delivery errors and unsuccessful transmissions of status updates. The main contributions of this paper are:

1) Two explicit algorithms for constructing status update dissemination schedules for any connected network.
2) A lower bound on the average peak AoI for the schedules generated by the algorithm without resampling by the root nodes in the sequential flooding trees.

We also provide numerical examples demonstrating the achieved average peak AoI for some canonical network topologies. The results show that resampling by the root nodes tends to provide greater gains in highly connected graphs.

II. System Model

Consider an N-node wireless network with a connected topology modeled by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. An example 5-node network is shown in Figure 1. The vertex set $\mathcal{V}$ represents the nodes and the edge set $\mathcal{E}$ represents the channels between the nodes in the network. Two vertices $i, j \in \mathcal{V}$ are adjacent if edge $e_{i,j}$ is in set $\mathcal{E}$. Equivalently, there exists a channel between nodes i and j; as such, any wireless transmission broadcast from node i is received at all nodes in the one-hop neighborhood of i, which we denote as $\mathcal{N}_1(i)$. For example, in Figure 1, we have $\mathcal{N}_1(2) = \{1, 3, 4\}$.

Each node $i \in \mathcal{V}$ can generate samples of a local random process $H_i(t)$ at any time $t \in \mathbb{Z}$ with *zero delay*. In addition to the status of its own process, every node in the network is also interested in updates of the status of the remaining $N-1$ processes in the network. We denote the status of process $H_i(t)$ from the perspective of node j at time t by $H_i^{(j)}(t)$. At any time t, each of the N nodes keeps a table of its most recently obtained status update of each of the N processes, giving a total of N^2 parameters throughout the network. Out of the N parameters at each node, one is obtained locally by *direct* observation, and $N-1$ are obtained by *indirect* observation from the statuses disseminated by other nodes.

We assume transmissions of status update packets require a fixed duration of one unit of time. Each packet includes information about the one process that is being transmitted and a time stamp indicating the time that the information was generated. For $i \in \mathcal{V}$, a packet transmitted by node i is successfully received by node j with probability $1-\epsilon$, where $0 \le \epsilon < 1$ and $j \in \mathcal{N}_1(i) \Leftrightarrow e_{i,j} \in \mathcal{E}$. Packet transmission errors are assumed to be independent across channels. We also assume that the transmitter knows whether the transmission to each of its neighbors was successful or not. The following definition formalizes the age metric considered in this paper.

Definition 1 (Age)**.** *Assume the most recent status update of the H_i process received at node j was timestamped at time t'. The **age** of status update $H_i^{(j)}$ at time $t \ge t'$ is defined as $\Delta_i^{(j)}(t) \triangleq t - t'$ for $j \neq i$.*

Since each node is assumed to have zero-delay access to the status of its local process, we have $\Delta_i^{(i)}(t) = 0$ for any $i \in \mathcal{V}$ and t. Hereafter, we only keep track of the $N^2 - N$ indirectly observed statuses throughout the network. Finally, we refer to a *schedule* as an ordered sequence of transmitting nodes and the corresponding status update parameter that they disseminate in each time slot.

III. Schedule Design for Status Update Dissemination

In this section we provide two algorithms that generate schedules for refreshing all of the status update parameters throughout the network with any arbitrary topology. The main idea for the schedule design is similar to the development of a sequential flooding algorithm that generates a periodic minimum-length schedule for a given network topology in [17], [18] except that nodes retransmit status updates until all "modified" one-hop neighbors have received at least one successful update of the process. The notion of a "modified" one-hop neighbor will be defined below.

To formalize this algorithm, recall that a set $\mathcal{S} \subset \mathcal{V}$ of vertices in a graph is called a *dominating set* if every vertex not in $\mathcal{S}$ is adjacent to a vertex in $\mathcal{S}$ [27]. A *minimum connected dominating set* (MCDS) $\mathcal{S} \subset \mathcal{V}$ is a dominating set with the properties that (i) the subgraph induced by $\mathcal{S}$, $\mathcal{G}[\mathcal{S}]$ is connected and (ii) $\mathcal{S}$ has the smallest cardinality among all connected dominating sets of $\mathcal{G}$. The cardinality of any MCDS is called the *connected domination number* of $\mathcal{G}$ and denoted by γ_c. In general, the MCDS is not unique [28], [29].

Definition 2 (Pseudo-leaf vertex)**.** *We refer to a vertex as a **pseudo-leaf** if it is not a member of any MCDS. That is $i \in \mathcal{V}$ is a pseudo-leaf if $i \notin \mathcal{U}$ where $\mathcal{S}_k \subset \mathcal{V}$ for $k = \{1, 2, \ldots, K\}$ represent all K possible MCDS's of $\mathcal{G}$ and*

$$\mathcal{U} \triangleq \bigcup_{k=1}^{K} \mathcal{S}_k. \tag{1}$$

Further, we refer to the set of all pseudo-leaf vertices of $\mathcal{G}$ by

$$\mathcal{L} \triangleq \mathcal{V} - \mathcal{U}. \tag{2}$$

Under this definition, every true leaf (i.e., every vertex with degree one) is also a pseudo-leaf. An example illustrating the notion of pseudo-leaf vertices and MCDS's is shown in Figure 1.

The sequential flooding schedule proceeds as follows. First, a fresh update of the $H_1(t)$ process is transmitted by node 1 until it is successfully received by all of its neighbors. Then this status is disseminated by the nodes in a MCDS if $1 \in \mathcal{L}$. Otherwise if $1 \notin \mathcal{L}$, or equivalently, node 1 is a member of at least one MCDS, the $H_1(t)$ status is disseminated by the remaining $\gamma_c - 1$ nodes in the MCDS. Since transmission errors may occur, we assume a retransmission policy where each node i retransmits a status update until the update has been received successfully by all nodes in the modified one-hop neighborhood of node i, denoted by set $\bar{\mathcal{N}_1}(i)$. This process is then repeated for the $H_2(t)$ process and so on until an update of the $H_N(t)$ process is successfully received at least once by all nodes in the network. This process then starts over again with node 1. In the following we formalize the notion of modified one-hop neighborhood and provide an example to further clarify the schedule design.

Without loss of generality assume that the indices of the nodes in the sequential flooding tree that disseminates updates of the $H_i(t)$ process form set $\mathcal{S}_{\text{sorted},i}$. The first index in this sorted set represents node i, since every time in order to refresh the H_i statuses throughout the network, first a fresh update is required to be disseminated by node i. Algorithm 1 describes how this sorted set is obtained. For the m^{th} element in the sorted set $\mathcal{S}_{\text{sorted},i}$, $m \in \{1, 2, \ldots, |\mathcal{S}_{\text{sorted},i}|\}$, where $|\mathcal{S}_{\text{sorted},i}| = \gamma_c + 1_{i \in \mathcal{L}}$ and $i \in \mathcal{V}$, we define the modified one-hop neighborhood of node $j = \mathcal{S}_{\text{sorted},i}(m)$ as

$$\bar{\mathcal{N}_1}(j) \triangleq \left[\mathcal{N}_1(j) - \bigcup_{m'=1}^{m-1} \mathcal{N}_1(\mathcal{S}_{\text{sorted},i}(m')) \right] - \{i\}. \tag{3}$$

In other words based on (3), node j does not need to refresh the statuses at the nodes that have been updated by the previous nodes in the sequential flooding tree. We also define the cardinality of the modified one-hop neighborhood in (3) as

$$J_{i,m} \triangleq |\bar{\mathcal{N}_1}(\mathcal{S}_{\text{sorted},i}(m))|. \tag{4}$$

There are some subtleties to this schedule that can be illustrated by considering an example in the setting shown in Figure 1. Suppose node 1 successfully transmits a status update to node 2 and that MCDS $\mathcal{S}_1$ is used for sequential flooding. Node 2 retransmits the update until both node 3 and node 4 successfully receive the update once. Although node 1 is in the one-hop neighborhood of node 2, it is not in the *modified* one-hop neighborhood of node 2 since node 1 has already been updated (directly). Similarly, when node 3 retransmits the update, it only requires node 5 to successfully receive the update once. Although node 2 is in the one-hop neighborhood of node 3, node 2 received the update (indirectly) earlier in this schedule. In this example we have $\mathcal{S}_{\text{sorted},1} = \{1, 2, 3\}$, $\mathcal{N}_1(1) = \{2\}$, $\mathcal{N}_1(2) = \{1, 3, 4\}$, $\mathcal{N}_1(3) = \{2, 5\}$, $\bar{\mathcal{N}}_1(1) = \{2\}$, $\bar{\mathcal{N}}_1(2) = \{3, 4\}$, $\bar{\mathcal{N}}_1(3) = \{5\}$, $J_{1,1} = 1$, $J_{1,2} = 2$, and $J_{1,3} = 1$.

We consider two variations of the sequential flooding tree schedule for updating the statuses throughout the network. The only difference is with regards to the root node in each of the sequential flooding trees. In the first variation, we assume the root node i retransmits without resampling $H_i(t)$ until all of its neighbors successfully receive an update. In the second variation, we assume that root node i resamples $H_i(t)$ in each retransmission until all of its neighbors successfully receive an update. Note that each neighbor may receive a different sample of $H_i(t)$ in the second variation. The former case is easier to analyze, whereas the latter case provides better performance.

In the following, Algorithm 1 and Algorithm 2 summarize these two approaches.

Algorithm 1: Schedule design *without resampling* by root nodes

1 initialize time, $t \leftarrow 0$;
2 **for** node $i = 1 : N$ **do**
3   **if** $\exists$ MCDS $\bar{\mathcal{S}}$ s.t. $i \in \bar{\mathcal{S}}$ **then**
4     $\mathcal{S} \leftarrow \bar{\mathcal{S}}$;
5   **else**
6     $\mathcal{S} \leftarrow \bar{\mathcal{S}} \cup \{i\}$, for any MCDS $\bar{\mathcal{S}} \subset \mathcal{V}$;
7   **end**
8   $\mathcal{S}_{\text{sorted},i}$ = Depth-First Search$(\mathcal{G}[\mathcal{S}], i)$;
9   node i generates a fresh sample of H_i;
10   **for** $m = 1 : |\mathcal{S}_{\text{sorted},i}|$ **do**
11     $j = \mathcal{S}_{\text{sorted},i}(m)$;
12     $t' \leftarrow t$;
13   **while** $\exists\ n \in \bar{\mathcal{N}}_1(j)$ s.t. n has not received a packet during interval $(t', t]$ **or** $t' == t$ **do**
14   node j transmits $H_i^{(j)}(t')$;
15       $t \leftarrow t + 1$;
16   **end**
17   **end**
18 **end**
19 go to line 2;

Observe that "Depth-First Search$(\mathcal{G}[\mathcal{S}], i)$" in Algorithm 1 describes an ordered list of vertices generated by performing a depth-first search of the graph induced by $\mathcal{S}$ where the search starts at root node i. The schedule generated by Algorithm 1 is periodic in the absence of packet errors; however, when packet errors are present it is not periodic in general.

Algorithm 2: Schedule design *with resampling* by root nodes

1 initialize time, $t \leftarrow 0$;
2 **for** node $i = 1 : N$ **do**
3   **if** $\exists$ MCDS $\bar{\mathcal{S}}$ s.t. $i \in \bar{\mathcal{S}}$ **then**
4     $\mathcal{S} \leftarrow \bar{\mathcal{S}}$;
5   **else**
6     $\mathcal{S} \leftarrow \bar{\mathcal{S}} \cup \{i\}$, for any MCDS $\bar{\mathcal{S}} \subset \mathcal{V}$;
7   **end**
8   $\mathcal{S}_{\text{sorted},i}$ = Depth-First Search$(\mathcal{G}[\mathcal{S}], i)$;
9   **for** $m = 1 : |\mathcal{S}_{\text{sorted},i}|$ **do**
10     $j = \mathcal{S}_{\text{sorted},i}(m)$;
11     $t' \leftarrow t$;
12   **while** $\exists\ n \in \bar{\mathcal{N}}_1(j)$ s.t. n has not received a packet during interval $(t', t]$ **or** $t' == t$ **do**
13   **if** $j == i$ **then**
14   node j generates a fresh sample of H_i;
15   **end**
16   node j transmits $H_i^{(j)}(t)$;
17       $t \leftarrow t + 1$;
18   **end**
19   **end**
20 **end**
21 go to line 2;

Similarly, the schedule generated by Algorithm 2 is not periodic in general.

IV. Lower Bound on the Average Peak Age of Information

In this section, we present an expression that lower bounds the a*verage peak AoI* for the schedules generated by Algorithm 1. Before proceeding, we first define the average peak AoI. Figure 2 represents an example age $\Delta_i^{(j)}(t)$ for some i and j where $i, j \in \mathcal{V},\ i \neq j$. The age value immediately before arrival of the q^{th} update of the H_i process at node j is

$$A_i^{(j)}(q) = a_i^{(j)}(q-1) + \tau_i^{(j)}(q), \tag{5}$$

where $a_i^{(j)}(q)$ represents the age of the q^{th} update at its arrival time at node j and $\tau_i^{(j)}(q)$ represents the interarrival time of the $(q-1)^{\text{th}}$ and q^{th} updates for $q \in \{1, 2, \ldots\}$. The initial age is denoted by $a_i^{(j)}(0)$.

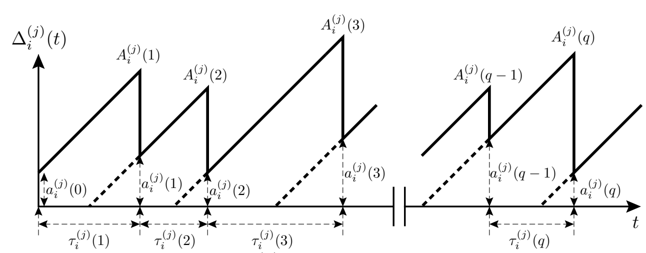

Fig. 2. An example age $\Delta_i^{(j)}(t)$ for some i and j where $i, j \in \mathcal{V},\ i \neq j$.

We define the average peak AoI over the N^2-N indirectly observed statuses throughout the network as

$$\bar{\Delta}_{\text{peak}} \triangleq \frac{1}{(N^2-N)} \sum_{\substack{i,j\in\mathcal{V}\\ i\neq j}} \bar{\Delta}_i^{(j)}, \tag{6}$$

where

$$\bar{\Delta}_i^{(j)} = \lim_{Q\to\infty} \frac{1}{Q}\sum_{q=1}^{Q} A_i^{(j)}(q). \tag{7}$$

Theorem 1 represents a lower bound on the average peak AoI of the schedules generated by Algorithm 1.

Theorem 1. *The average peak AoI of the N^2-N indirectly observed statuses throughout the network for the schedules generated by Algorithm 1 is lower bounded by*

$$\bar{\Delta}_{\text{peak}} = \bar{d} + \sum_{i=1}^{N}\sum_{m=1}^{\gamma_c+1_{i\in\mathcal{L}}}\sum_{n=1}^{J_{i,m}} \binom{J_{i,m}}{n}\frac{(-1)^{n+1}}{(1-\epsilon^n)}, \tag{8}$$

where

$$\bar{d} \triangleq \frac{1}{N^2-N}\sum_{\substack{i,j\in\mathcal{V}\\ i\neq j}} d(i,j) \tag{9}$$

is the average distance of the network, and $d(i,j)$ is the distance in hops of the shortest path between nodes i and j.

Proof: From (5) and (6) we can write

$$\bar{\Delta}_{\text{peak}} = \frac{1}{(N^2-N)}\sum_{\substack{i,j\in\mathcal{V}\\ i\neq j}}\left[\lim_{Q\to\infty}\frac{1}{Q}\sum_{q=1}^{Q} a_i^{(j)}(q-1)+\tau_i^{(j)}(q)\right], \tag{10a}$$

$$\geq \bar{d} + \underbrace{\frac{1}{(N^2-N)}\sum_{\substack{i,j\in\mathcal{V}\\ i\neq j}}\left[\lim_{Q\to\infty}\frac{1}{Q}\sum_{q=1}^{Q}\tau_i^{(j)}(q)\right]}_{\triangleq\bar{\tau}}, \tag{10b}$$

$$= \bar{d}+\bar{\tau}, \tag{10c}$$

where (10b) is obtained considering (9) and the fact that $a_i^{(j)}(q) \geq d(i,j)$ for all i, j and q, and $\bar{\tau}$ in (10c) is obtained in Corollary 1. This completes the proof. ∎

Corollary 1 represents an expression for $\bar{\tau}$, which we refer to as the average interarrival time.

Corollary 1. *The average interarrival time of the N^2-N statuses throughout the network for the schedules generated by Algorithm 1 is given by*

$$\bar{\tau} = \sum_{i=1}^{N}\sum_{m=1}^{\gamma_c+1_{i\in\mathcal{L}}}\sum_{n=1}^{J_{i,m}} \binom{J_{i,m}}{n}\frac{(-1)^{n+1}}{(1-\epsilon^n)}. \tag{11}$$

Proof: Without loss of generality, consider dissemination of the H_i process throughout the network for $i\in\mathcal{V}$. The H_i process should be disseminated by the nodes in the sorted set $\mathcal{S}_{\text{sorted},i}$ in the schedule generated by Algorithm 1. Denote the indices of the nodes in this sorted set by $m\in\{1,\ldots,\gamma_c+1_{i\in\mathcal{L}}\}$, and the number of transmissions by the m^{th} node to update the nodes in $\bar{\mathcal{N}}_1(\mathcal{S}_{\text{sorted},i}(m))$ by $k_{i,m}$. For $\ell=\{1,2,..\}$ we get

$$\Pr\{k_{i,m}=\ell\} = (1-\epsilon^{\ell})^{J_{i,m}} - (1-\epsilon^{\ell-1})^{J_{i,m}}. \tag{12}$$

From (12) we can write

$$\mathrm{E}[k_{i,m}] = \sum_{\ell=1}^{\infty} \ell \Pr\{k_{i,m}=\ell\}, \tag{13a}$$

$$= \sum_{\ell=1}^{\infty} \ell[(1-\epsilon^{\ell})^{J_{i,m}} - (1-\epsilon^{\ell-1})^{J_{i,m}}], \tag{13b}$$

$$= \sum_{\ell=1}^{\infty} \ell\left\{\sum_{n=0}^{J_{i,m}}\binom{J_{i,m}}{n}[(-\epsilon^{\ell})^n-(-\epsilon^{\ell-1})^n]\right\}, \tag{13c}$$

$$= \sum_{n=1}^{J_{i,m}}\binom{J_{i,m}}{n}(-1)^n\left(1-\frac{1}{\epsilon^n}\right)\left[\sum_{\ell=1}^{\infty}\ell(\epsilon^n)^{\ell}\right], \tag{13d}$$

$$= \sum_{n=1}^{J_{i,m}}\binom{J_{i,m}}{n}(-1)^n\left(1-\frac{1}{\epsilon^n}\right)\frac{\epsilon^n}{(1-\epsilon^n)^2}, \tag{13e}$$

$$= \sum_{n=1}^{J_{i,m}}\binom{J_{i,m}}{n}\frac{(-1)^{n+1}}{(1-\epsilon^n)}. \tag{13f}$$

Now, observe the number of transmissions required by any of the nodes in a given sequential flooding tree in a schedule generated by Algorithm 1 is independent of the number of transmissions by any other transmitting node. Considering the result in (13f) over all $m\in\{1,\ldots,\gamma_c+1_{i\in\mathcal{L}}\}$ and $i\in\mathcal{V}$, the average interarrival time in (11) is obtained. ∎

For $\epsilon=0$, note that $\mathrm{E}[k_{i,m}]=1$, which gives $\bar{\tau}=N\gamma_c+|\mathcal{L}|$. This result is consistent with [17], [18].

V. Numerical Results

This section presents numerical examples to illustrate the achieved average peak AoI of the schedules generated by Algorithm 1 and Algorithm 2 and compares the achieved ages with the lower bound in Theorem 1. Figure 3 and Figure 4 represent the achieved average peak AoI for fully-connected networks (K_N) and ring networks (C_N), respectively, versus the number of nodes $N\in\{3,\ldots,10\}$ and for error probabilities $\epsilon\in\{0,0.25,0.5\}$. For the simulation lines, both Algorithm 1 and Algorithm 2 are run over an interval of 10^5 time slots. The results show that the achieved average peak AoI is an strictly increasing function of the number of nodes N and error probability ϵ. When $\epsilon=0$, the schedules generated by Algorithm 1 and Algorithm 2 are identical and have the same average peak AoI.

For the fully-connected network case in Figure 3, each transmitting node needs to update the tables at its $N-1$ neighbors. For the ring network network case in Figure 4, each transmitting node has at most 2 neighbors that it needs to update. As a result, when $\epsilon>0$, resampling by root nodes as specified in Algorithm 2 tends to lead to a more significant reduction in the achieved average peak AoI in fully-connected networks than in ring networks when compared to the average peak AoI of schedules generated by Algorithm 1.

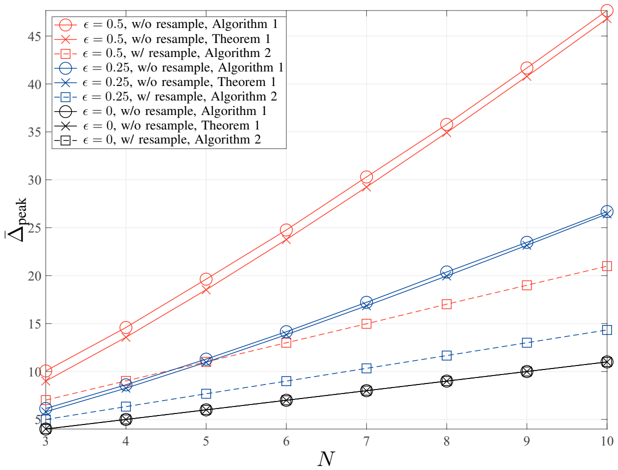

Fig. 3. Achieved average peak AoI versus the number of nodes N and error probabilities $\epsilon \in \{0, 0.25, 0.5\}$ for fully-connected networks K_N.

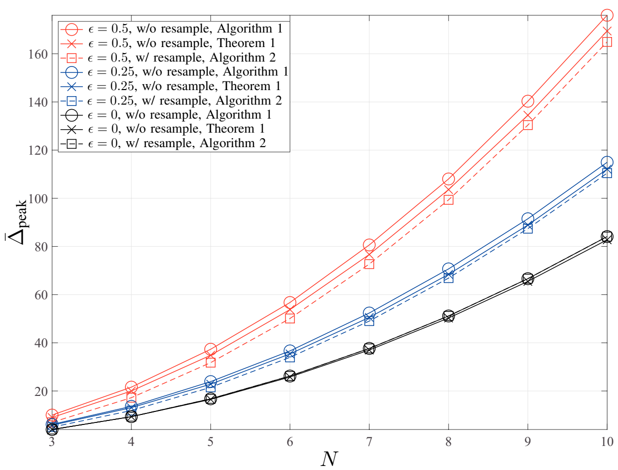

Fig. 4. Achieved average peak AoI versus the number of nodes N and error probabilities $\epsilon \in \{0, 0.25, 0.5\}$ for ring networks C_N.

VI. Conclusion

This paper studied the age of information in a multi-source multi-hop status update system with nodes communicating over unit-delay channels with packet transmission losses. We presented two algorithms that generate schedules for dissemination of status updates throughout any given network with a connected topology. For the schedules constructed by the algorithm without resampling by the root nodes of the flooding trees, we derived a closed-form expression that lower bounds the achieved average peak AoI. Future directions of this work include studying the effect of packet losses due to collisions where multiple nodes transmit simultaneously.

References

[1] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *Proc. IEEE SECON*, 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*, Mar. 2012, pp. 2731–2735.

[3] M. Costa, M. Codreanu, and A. Ephremides, "On the age of information in status update systems with packet management," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1897–1910, Feb. 2016.

[4] S. Farazi, A. G. Klein, and D. R. Brown III, "On the average staleness of global channel state information in wireless networks with random transmit node selection," in *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 3621–3625.

[5] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7492–7508, Aug. 2017.

[6] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Age-optimal information updates in multihop networks," in *Proc. IEEE ISIT*, Aug. 2017, pp. 576–580.

[7] S. Farazi, D. R. Brown III, and A. G. Klein, "On global channel state estimation and dissemination in ring networks," in *Proc. Asilomar Conf. on Signals, Systems, and Computers*, Nov. 2016, pp. 1122–1127.

[8] 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.

[9] A. G. Klein, S. Farazi, W. He, and D. R. Brown III, "Staleness bounds and efficient protocols for dissemination of global channel state information," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5732–5746, Sep. 2017.

[10] S. Farazi, A. G. Klein, and D. R. Brown III, "Bounds on the age of information for global channel state dissemination in fully-connected networks," in *Proc. Intl. Conf. on Computer Communication and Networks (ICCCN)*, Jul. 2017, pp. 1–7.

[11] Y.-P. Hsu, E. Modiano, and L. Duan, "Age of information: Design and analysis of optimal scheduling algorithms," in *Proc. IEEE ISIT*, Aug. 2017, pp. 561–565.

[12] R. Talak, S. Karaman, and E. Modiano, "Minimizing age-of-information in multi-hop wireless networks," in *Proc. of Allerton Conf. on Commun., Contr., and Computing*, Oct. 2017.

[13] V. Tripathi and S. Moharir, "Age of information in multi-source systems," in *Proc. IEEE GLOBECOM*, Dec. 2017, pp. 1–6.

[14] S. Farazi, A. G. Klein, and D. R. Brown III, "Average age of information for status update systems with an energy harvesting server," in *Proc. IEEE INFOCOM WKSHPS*, Apr. 2018, pp. 112–117.

[15] S. Farazi, A. G. Klein, and D. R. Brown III, "Age of information in energy harvesting status update systems: When to preempt in service?" in *Proc. IEEE ISIT*, Jun. 2018, pp. 2436–2440.

[16] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1807–1827, Mar. 2019.

[17] S. Farazi, A. G. Klein, J. A. McNeill, and D. R. Brown III, "On the age of information in multi-source multi-hop wireless status update networks," in *Proc. IEEE SPAWC*, Jun. 2018.

[18] S. Farazi, A. G. Klein, and D. R. Brown III, "Fundamental bounds on the age of information in multi-hop global status update networks," *Journal of Communications and Networks*, vol. 21, no. 3, pp. 268–279, 2019.

[19] S. Farazi, A. G. Klein, and D. R. Brown III, "Fundamental bounds on the age of information in general multi-hop interference networks," in *Proc. IEEE INFOCOM WKSHPS*, Apr. 2019, pp. 96–101.

[20] K. Chen and L. Huang, "Age-of-information in the presence of error," in *Proc. IEEE ISIT*, Jul. 2016, pp. 2579–2583.

[21] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor, "Online timely status updates with erasures for energy harvesting sensors," in *Proc. of Allerton Conf. on Commun., Contr., and Computing*, Oct. 2018, pp. 966–972.

[22] S. Feng and J. Yang, "Age of information minimization for an energy harvesting source with updating erasures: With and without feedback," *arXiv preprint arXiv:1808.05141*, 2018.

[23] R. Yates, E. Najm, E. Soljanin, and J. Zhong, "Timely updates over an erasure channel," *arXiv preprint arXiv:1704.04155*, 2017.

[24] R. D. Yates and S. K. Kaul, "Status updates over unreliable multiaccess channels," in *Proc. IEEE ISIT*, Aug. 2017, pp. 331–335.

[25] 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*, 2019.

[26] R. Talak, I. Kadota, S. Karaman, and E. Modiano, "Scheduling policies for age minimization in wireless networks with unknown channel state," in *Proc. IEEE ISIT*, Jun. 2018, pp. 2564–2568.

[27] D. B. West, *Introduction to Graph Theory*, 2nd ed. Prentice Hall, 2000.

[28] E. Sampathkumar and H. Walikar, "The connected domination number of a graph," *J. Math. Phys*, 1979.

[29] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Proc. IEEE Intl. Conf. on Comm. (ICC)*. IEEE, 1997, pp. 376–380.