Timely Gossiping with File Slicing and Network Coding

Priyanka Kaswan  Sennur Ulukus
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742
pkaswan@umd.edu  ulukus@umd.edu

Abstract—We consider a system consisting of a large network of \( n \) users and a library of files, wherein inter-user communication is established based upon gossip mechanisms. Each file is initially present at exactly one node, which is designated as the file source. The source gets updated with newer versions of the file according to an arbitrary distribution in real time, and the other users in the network wish to acquire the latest possible version of the file. We present a class of gossip protocols that achieve \( O(1) \) age at a typical node in a single-file system and \( O(n) \) age at a typical node for a given file in an \( n \)-file system. We show that file slicing and network coding based protocols fall under the presented class of protocols. Numerical evaluation results are presented to confirm the aforementioned bounds.

I. INTRODUCTION

Last couple of decades have seen a surge in different types of large communication networks, e.g., dense sensor networks, mobile ad-hoc networks, content distribution networks, autonomous driving networks, and so on, where the network unpredictability and resource-constrained infrastructure calls for employment of decentralized algorithms for efficient communication of information across the network.

Gossip algorithms are decentralized algorithms where two nodes of the network randomly come into contact and communicate exclusively based on the limited information available at the liaising nodes. As opposed to centralized protocols, where a centralized server coordinates or assists the exchange of files between nodes of the network, in gossip algorithms, the nodes are unaware of the information available at other nodes and all actions by a node are taken solely based on its local status or information obtained from its neighbors. The idea of gossip protocols was first mentioned in [1] and since then there have been numerous works on gossip protocols, e.g., [2]–[10].

In this respect, the works that are directly related to our work are [5]–[7]. In [5], dissemination time of \( k \) messages in a large network of \( n \) nodes is studied for gossip protocols based on random linear coding (RLC), random message selection (RMS), and sequential dissemination. [5] shows that RLC-based protocol has superior performance and has \( ck + O(\sqrt{k \log k \log n}) \) dissemination time in complete graphs. [6] further extends the result to arbitrary graphs. [7] studies the dissemination of a file in a large network of \( n \) nodes by dividing the file into \( k \) pieces. Therein, the authors propose a hybrid piece selection protocol, called INTERLEAVE, which achieves a \( O(k + \log n) \) total dissemination time for a file.

However, all these works focus on using the dissemination time of a specific message or a set of messages to all nodes of the network as the performance measure. In most practical networks, information is dynamic in nature, and all nodes need to be continuously delivered with the latest file versions in real time. Therefore, in such networks, the age of information (AoI) of messages at the nodes proves to be a more relevant performance metric. AoI has been studied in a range of contexts, such as, queueing networks, energy harvesting systems, caching systems, web crawling, scheduling problems, remote estimation, UAV systems, and so on, see e.g., [11]–[53].

In this paper, we use version age of information as the metric to measure the timeliness of information, which counts how many versions behind a file is at a node from the version currently prevailing at the source. A predecessor of version age was introduced in [29] in the context of the timeliness of the Google Scholar citation index by using the difference between the actual citation numbers and the latest updated citation numbers of the researchers on the citation index. The concept was formally defined in [8], [50].

Gossip networks have been analyzed from an information timeliness perspective in [8]–[10]. [8] shows that in a complete symmetric network of \( n \) nodes with a single file, the average age at each node is \( O(\log n) \). [9] demonstrates that \( O(\log n) \) average version age is achievable per node in fully connected cluster models. [10] extends [8] from the version age metric to the binary freshness metric. An interesting question to ask is whether we can do better than \( O(\log n) \) version age [8]. [9] in a fully connected network. Similarly, if we disseminate \( n \) files simultaneously to all \( n \) nodes in real time in a network by dividing all rate resources among \( n \) files, we obtain a version age of order \( O(n \log n) \). Another interesting question is whether we can do better than \( O(n \log n) \) for \( n \) files.

In this work, we achieve \( O(1) \) version age for single-file dissemination and \( O(n) \) version age for \( n \)-file dissemination (improving both results by \( \log n \)), in a network of \( n \) nodes, that communicate using a random phone call model. We show that complete dissemination of a file to the entire network is not necessary to achieve these age bounds. Further, periodic pausing of dissemination of newer file versions from the source does not hinder the achievement of these bounds. We show that specific file splitting and network coding based gossip protocols achieve the mentioned bounds. Finally, we present simulation results to validate the aforementioned bounds.

Abstract—We consider a system consisting of a large network of \( n \) users and a library of files, wherein inter-user communication is established based upon gossip mechanisms. Each file is initially present at exactly one node, which is designated as the file source. The source gets updated with newer versions of the file according to an arbitrary distribution in real time, and the other users in the network wish to acquire the latest possible version of the file. We present a class of gossip protocols that achieve \( O(1) \) age at a typical node in a single-file system and \( O(n) \) age at a typical node for a given file in an \( n \)-file system. We show that file slicing and network coding based protocols fall under the presented class of protocols. Numerical evaluation results are presented to confirm the aforementioned bounds.

I. INTRODUCTION

Last couple of decades have seen a surge in different types of large communication networks, e.g., dense sensor networks, mobile ad-hoc networks, content distribution networks, autonomous driving networks, and so on, where the network unpredictability and resource-constrained infrastructure calls for employment of decentralized algorithms for efficient communication of information across the network.

Gossip algorithms are decentralized algorithms where two nodes of the network randomly come into contact and communicate exclusively based on the limited information available at the liaising nodes. As opposed to centralized protocols, where a centralized server coordinates or assists the exchange of files between nodes of the network, in gossip algorithms, the nodes are unaware of the information available at other nodes and all actions by a node are taken solely based on its local status or information obtained from its neighbors. The idea of gossip protocols was first mentioned in [1] and since then there have been numerous works on gossip protocols, e.g., [2]–[10].

In this respect, the works that are directly related to our work are [5]–[7]. In [5], dissemination time of \( k \) messages in a large network of \( n \) nodes is studied for gossip protocols based on random linear coding (RLC), random message selection (RMS), and sequential dissemination. [5] shows that RLC-based protocol has superior performance and has \( ck + O(\sqrt{k \log k \log n}) \) dissemination time in complete graphs. [6] further extends the result to arbitrary graphs. [7] studies the dissemination of a file in a large network of \( n \) nodes by dividing the file into \( k \) pieces. Therein, the authors propose a hybrid piece selection protocol, called INTERLEAVE, which achieves a \( O(k + \log n) \) total dissemination time for a file.

However, all these works focus on using the dissemination time of a specific message or a set of messages to all nodes of the network as the performance measure. In most practical networks, information is dynamic in nature, and all nodes need to be continuously delivered with the latest file versions in real time. Therefore, in such networks, the age of information (AoI) of messages at the nodes proves to be a more relevant performance metric. AoI has been studied in a range of contexts, such as, queueing networks, energy harvesting systems, caching systems, web crawling, scheduling problems, remote estimation, UAV systems, and so on, see e.g., [11]–[53].

In this paper, we use version age of information as the metric to measure the timeliness of information, which counts how many versions behind a file is at a node from the version currently prevailing at the source. A predecessor of version age was introduced in [29] in the context of the timeliness of the Google Scholar citation index by using the difference between the actual citation numbers and the latest updated citation numbers of the researchers on the citation index. The concept was formally defined in [8], [50].

Gossip networks have been analyzed from an information timeliness perspective in [8]–[10]. [8] shows that in a complete symmetric network of \( n \) nodes with a single file, the average age at each node is \( O(\log n) \). [9] demonstrates that \( O(\log n) \) average version age is achievable per node in fully connected cluster models. [10] extends [8] from the version age metric to the binary freshness metric. An interesting question to ask is whether we can do better than \( O(\log n) \) version age [8]. [9] in a fully connected network. Similarly, if we disseminate \( n \) files simultaneously to all \( n \) nodes in real time in a network by dividing all rate resources among \( n \) files, we obtain a version age of order \( O(n \log n) \). Another interesting question is whether we can do better than \( O(n \log n) \) for \( n \) files.

In this work, we achieve \( O(1) \) version age for single-file dissemination and \( O(n) \) version age for \( n \)-file dissemination (improving both results by \( \log n \)), in a network of \( n \) nodes, that communicate using a random phone call model. We show that complete dissemination of a file to the entire network is not necessary to achieve these age bounds. Further, periodic pausing of dissemination of newer file versions from the source does not hinder the achievement of these bounds. We show that specific file splitting and network coding based gossip protocols achieve the mentioned bounds. Finally, we present simulation results to validate the aforementioned bounds.
II. GOSSIPING WITH A SINGLE FILE

We consider slotted time, where one time slot is the duration of time needed to receive an entire file. The network consists of \( n \) nodes, and a node, called the source, receives updates for the file according to an arbitrary distribution. All source updates start at the beginning of a time slot. Hence, the source may get a new version of the file at any time. We follow the random phone call model (i.e., rumor mongering model), where the algorithm works in rounds, and in every round, each user, uniformly at random, selects another user, called the target node, to communicate with.

**Theorem 1** In a network of \( n \) users, where users communicate according to a gossip protocol, if there exists \( p \in [0, 1] \) and \( n_0 \in \mathbb{N} \), such that for all \( n > n_0 \), the probability of a random node receiving the file in \( O(1) \) time is greater than \( 1 - \varepsilon \), then the protocol achieves an \( O(1) \) age at each node.

**Proof:** Let \( c = O(1) \) time slots, \( c \in \mathbb{N} \), which is also taken to be the duration of a cycle. The timeline is thus divided into consecutive cycles, with each cycle consisting of \( c \) time slots; see Fig. 1. At the beginning of each cycle, the source begins to transmit the latest version it has received by the end of the previous time slot to the network. The network is allowed to gossip for \( c \) time slots according to the given protocol. To avoid mixing of different versions of a file, when the cycle ends, the nodes lay off transmission of the file versions already in their possession, and the circulation of a newer version is set in motion with the beginning of the next cycle.

In time slot \( t \), let \( N^t_i \) be the latest version of the file available at node \( i \), and let \( N^t_i \) be the latest version present at the source. Thus, the instantaneous age of information at node \( i \) at time \( t \) is \( X^t_i = N^s_i - N^t_i \). Therefore, every time the source receives an update, \( N^s_i \), and hence, the instantaneous age at each node in the network, increment by one. We assume that all nodes start with zero age at \( t = 0 \). At \( t = kc \), \( k \in \mathbb{N} \), the cycle \( k \) begins, and the source node starts the transmission of version \( N^s_{kc} \) to the network. During cycle \( k \), i.e., between time slot \( kc \) and \((k + 1)c\), the source may receive a maximum of \( c \) updates in the \( c \) time slots, but it continues to circulate the version \( N^s_{kc} \) and does not transmit the newer versions. If node \( i \) receives the version \( N^s_{kc} \) at any time \( t \), \( kc \leq t < (k + 1)c \), then the instantaneous version age at node \( i \) drops to \( X^t_i = N^s_i - N^s_{kc} \).

Let \( Z^t_k \) indicate if node \( i \) successfully receives the file in cycle \( k \). Hence,

\[
Z^t_k = \begin{cases} 
1, & \text{if node } i \text{ receives version } N^s_{kc}, \text{in cycle } k \\
0, & \text{otherwise}
\end{cases} \tag{1}
\]

Note that \( Z^t_k \) subscript \( k \) denotes the cycle index whereas \( N^s_i \), \( N^t_i \) and \( X^t_i \) subscripts \( t \) denote the time slot index (see Fig. 1).

At the end of cycle \( k \),

\[
X^t_{(k+1)c} = \begin{cases} 
X^t_{kc} + N^s_{(k+1)c} - N^s_{kc}, & \text{if } Z^t_k = 0 \\
N^s_{(k+1)c} - N^s_{kc}, & \text{if } Z^t_k = 1
\end{cases} \tag{2}
\]

where we note that the source would have moved from version \( N^s_{kc} \) to version \( N^s_{(k+1)c} \) through cycle \( k \), and \( X^t_{kc} = N^s_{kc} - N^t_{kc} \). Here, observe that if \( Z^t_k = 0 \), then \( N^t_{(k+1)c} = N^t_{kc} \), and if \( Z^t_k = 1 \), then \( N^t_{(k+1)c} = N^s_{kc} \).

Next, we define \( Y^t_i \) for node \( i \) (see Fig. 1), where the subscript \( t \) is the time slot index. For \( kc \leq t < (k + 1)c \),

\[
Y^t_i = 2cZ^t_{k-1} + (1 - Z^t_{k-1})(Y^t_{kc-1} + c) \tag{3}
\]

with \( Y^0_i = 2c \) for cycle 0. Note that \( Y^t_i \) remains constant over the duration of any cycle.

Next, we show that \( Y^t_i \) forms an upper bound on \( X^t_i \) for all \( t \). First, note that, since the source can get a maximum of \( c \) new updates in a cycle of \( c \) slots, we have \( N^s_{kc} - N^s_{(k-1)c} \leq c \). Then, if \( Z^t_{k-1} = 1 \), i.e., if node \( i \) is successfully updated in cycle \( k - 1 \), from (2), we have \( X^t_{kc} \leq c \), i.e., at the beginning of the \( k \)th cycle, the \( i \)th node’s version age is upper bounded by \( c \). By repeating this argument, \( N^s_{(k+1)c} - N^s_{kc} \leq c \), and re-employing (2) gives \( X^t_i \leq 2c \) throughout the \( k \)th cycle, irrespective of the value of \( Z^t_k \), i.e., whether the \( i \)th node is updated or not in the \( k \)th cycle. Note from (3) that when \( Z^t_{k-1} = 1 \) we have \( Y^t_i = 2c \) for the entire \( k \)th cycle. Thus, when \( Z^t_{k-1} = 1 \), we have \( X^t_i \leq Y^t_i = 2c \) throughout the \( k \)th cycle. On the other hand, if \( Z^t_{k-1} = 0 \), since the source can get at most \( c \) updates over \( c \) slots, \( X^t_i \leq X^t_{kc-1} + c \) for \( t \in \{kc, (k+1)c\} \). In this case, from (3), \( Y^t_i = Y^t_{kc-1} + c \). All of these imply that \( Y^t_i \leq X^t_i \) for all \( t \), as graphically shown in Fig. 1.

Next, we bound \( \mathbb{E}[Y^t_i] \) which in turn bounds \( \mathbb{E}[X^t_i] \) due to \( X^t_i \leq Y^t_i \). Consider \( kc \leq t < (k + 1)c \). Recall that if node \( i \) successfully receives a file in cycle \( (k - 1) \), i.e., \( Z^t_{k-1} = 1 \), then \( Y^t_i \) drops to \( 2c \), otherwise it increases by \( c \). Hence, for \( m \in \{2, \ldots, k + 1\} \), \( Y^t_i = mc \) is possible only if \( Z^t_{k-m+1} = 1 \) and \( Z^t_j = 0 \) for all \( j \in \{k - m + 2, \ldots, k-1, \ldots, 1\} \). That is, the node was updated \( m - 1 \) slots ago, and could not get updated after that, accumulating an increment of \( c \) in \( Y^t_i \) value in each of the later slots. Further, if \( Y^t_i = (k + 2)c \), then that means that node \( i \) failed to receive any updates since \( t = 0 \). Since \( Z^t_k \) are i.i.d. in \( k \) for node \( i \), define \( p_n = \mathbb{P}(Z^t_k = 0) \). Since we consider a symmetric fully connected network, all nodes have the same per cycle failure probability \( p_n \) in an \( n \) node network. Hence, we have the following bound,

\[
\mathbb{E}[Y^t_i] = \sum_{m=2}^{k+1} mc(1 - p_n)p_n^{m-2} + (k + 2)cp_n^k < \frac{c}{p_n} \sum_{m=2}^{k+1} mp_n^{m-1} \tag{4}
\]

\[
\leq \frac{c}{p_n} \sum_{m=2}^{k+2} mp_n^{m-1} \tag{5}
\]

\[
\leq \frac{c(2 - p_n)}{(1 - p_n)^2} \tag{6}
\]

\[
= \frac{c}{(1 - p_n)^2} + \frac{c}{1 - p_n} \tag{7}
\]

Note that the expression in (7) is an increasing function of \( p_n \). As the statement of the theorem states \( \mathbb{P}(Z^t_j = 1) > 1 - p \) for all \( n > n_0 \), thus \( p_n < p \), and consequently

\[
\mathbb{E}[Y^t_i] < \frac{c}{(1 - p)^2} + \frac{c}{1 - p} \tag{8}
\]
Hence, $\mathbb{E}[X_i^t] < \mathbb{E}[Y_i^t] < \frac{c}{1-p} + \frac{c}{1-p^*}$ for network size $n > n_0$, and the protocol achieves $O(1)$ age at each node.

We have made several relaxations and sub-optimalities in the analysis above. Before we look back at all these relaxations, we first consider an example of such a gossip protocol, the INTERLEAVE protocol proposed in [7]. Motivated by file splitting employed in protocols such BitTorrent, INTERLEAVE is a file splitting based dissemination scheme that is able to satisfy the conditions of Theorem 1 as we show next.

A. An Achievable Gossiping Protocol: INTERLEAVE

The INTERLEAVE protocol of [7] employs an alternating combination of push and pull actions at each user in the network to disseminate a file. The file is split into $\ell$ pieces by the source, and the pieces are numbered before dissemination to the network. Each piece consumes $\frac{1}{\ell}$ fraction of a time slot for transmission, which we define to be one minislot. Thus, in the timeline, each time slot is further divided into $\ell$ minislots.

In an odd minislot, each user chooses a target node uniformly at random from the network, and pushes to its target node the highest numbered piece it has received in the previous odd minislots. The source pushes a fresh piece to a target node in the network in every odd minislot, where the pieces are chosen in the order they are numbered. In an even minislot, each user again chooses a target node uniformly at random from the network, and makes a pull request for the lowest numbered piece that is missing in its collection from its target node. Node $i$ is said to have received a particular version of the file when it has accumulated all pieces of that version.

[7, Thm. 6] provides that INTERLEAVE disseminates all $\ell$ pieces of a file to all $n$ nodes of the network within $9(\ell + \log n)$ minislots, with probability $1 - 5n^{-s}$, for any $s < 0.5$. If we choose $\ell \geq \alpha \log n$ for some constant $\alpha > 0$, then considering a minislot to be $\frac{1}{\ell}$ duration of a full time slot, INTERLEAVE takes $9(1 + \log n) = O(1)$ time for spreading a particular file to the entire network with high probability. Since the probability $1 - 5n^{-s}$ is an increasing function of $n$, the INTERLEAVE protocol satisfies the requirements of Theorem [1] and hence can help achieve $O(1)$ expected age at any node in the network.

The inner-workings of INTERLEAVE may look complicated at first, and it may look as though it requires more information than the gossip algorithm in [8]. To this, we
point out that, in INTERLEAVE all nodes are oblivious of the collection of file pieces existing at other nodes of the network. Each node makes a pull request for the lowest numbered piece missing in its private collection, and pushes out the highest numbered piece in its private collection. Therefore, the decision making in INTERLEAVE is independent of the information at other nodes, and thus, the INTERLEAVE protocol is a gossip protocol, as the one in [8].

Next, we note the relaxations made in the calculations in Theorem 1. 1) The probability of successfully receiving a file in a cycle, \( P(Z_k^c = 1) \), is replaced by its lower bound \( 1 - p \).
2) At the beginning of each new cycle, all nodes are made to halt transmissions of all files in their possession until the previous cycle. This could result in some nodes sitting idle in the beginning of a cycle waiting for a piece of the newer version, while they could have continued disseminating pieces of previous version missing in other nodes. 3) \( Y_i^t \) is a crude upper bound to \( X_i^t \); \( Y_i^t \) only changes value at the end of a cycle, though it is possible for node to gather all pieces of a file before the end of the cycle. 4) The evolution of \( Y_i^t \) assumes the worst case scenario where the source gets the maximum possible \( c \) version updates in every cycle, and hence file version received by the end of the cycle is always \( c \) slots old. Further, \( Y_i^t \) is set constant over cycle \( k \), upper bounding the maximum value of \( X_i^t \) within that cycle. 5) The protocol may take less than \( c \) time slots for larger values of \( n \) for disseminating the file to all nodes with same probability. Hence, use of \( c \) time slots in a cycle might be sub-optimal.

Finally, as a general remark on Theorem 1: interestingly, a protocol can achieve \( O(1) \) age even if the success probability \( 1 - p \) has a very small value, causing very few nodes to really receive a file in a particular cycle. One reason for this can be that every cycle presents a fresh opportunity for a node to receive a new version and reduce its age. Hence it is okay to miss out some updates in a couple of cycles.

III. Gossiping with Multiple Files

We consider a system consisting of a network of \( n \) nodes and a library of \( n \) files. Each node is responsible for creating version updates for a unique file. We label the node acting as source for file \( f \) as node \( f \). Then, file \( f \) is termed as the source file for node \( f \). We consider slotted time, and define one time slot to be the time required to receive an entire file. All file transmissions start at the beginning of a time slot and consume an entire time slot. We follow the random phone call model, where in every time slot, each user uniformly at random picks another user, referred to as the target, and communicates with either a pull or a push action based on a gossip protocol.

Node \( f \) receives version updates for its source file \( f \) according to an arbitrary distribution. Thus, in each time slot, node \( f \) may or may not get a newer version update. The updates at all nodes for their respective source files are considered uncoordinated, implying that some files may get updated more frequently at their respective source nodes than others. All users wish to obtain the latest possible version of all files at any point of time.

**Theorem 2** In a network of \( n \) users and \( n \) files, where users communicate according to a gossip protocol, if there exists \( p \in [0,1) \) and \( n_0 \in \mathbb{N} \), such that for all \( n > n_0 \), the probability of a random node receiving all \( n \) files in \( O(n) \) time is greater than \( 1 - p \), then the protocol achieves \( O(n) \) age for each file at each node.

**Proof:** Let \( cn = O(n) \) time slots, \( c \in \mathbb{N} \), which is also taken to be the duration of one cycle. The timeline is divided into consecutive cycles where each cycle consists of \( cn \) time slots. At the beginning of each cycle, all nodes begin dissemination of the latest version of their respective source files received by the end of the previous time slot to the network. The network is allowed to gossip over one cycle or \( cn \) time slots according to the given protocol. To avoid mixing of multiple versions of different files, at the end of the cycle, all nodes lay off transmission of file versions already in their possession, and circulation of newer versions of all files starts with the beginning of the next cycle.

In time slot \( t \), let \( N_{ij}^f \) denote the latest version of file \( f \) available at node \( i \) and let \( N_{ij}^f \) be the latest version of file \( f \) at its source node \( f \). The instantaneous version age for file \( f \) at node \( i \) is \( X_{ij}^f = N_{ij}^f - N_{ij}^f \). Therefore, every time source node \( f \) is updated with a newer version for file \( f \), the instantaneous version age at every other node for file \( f \) increments by one.

The evolution of \( X_{ij}^f \) is similar to Section III. At the beginning of cycle \( k \), each node \( f \) begins transmission of version \( N_{knc}^f \) of its source file \( f \) to the network. During the cycle, every source node can receive a maximum of \( cn \) updates for its source file. The source nodes hold back the newer versions and so the system continues circulation of version \( N_{knc}^f \) for each file \( f \). We define \( Z_{ij}^f \) for cycle \( k \) as follows

\[
Z_{ij}^f = \begin{cases} 1, & \text{node } i \text{ receives version } N_{knc}^f \text{ in cycle } k \\ 0, & \text{node } i \text{ does not receive } N_{knc}^f \text{ in cycle } k \end{cases}
\]

For upper bounding the expected version age at time \( t \) for file \( f \) at node \( i \), \( E[X_{ij}^f] \), we define \( Y_{ij}^f \), for \( kcn \leq t < (k+1)cn \), as follows

\[
Y_{ij}^f = 2ncZ_{k+1}^f + (1 - Z_{k-1}^f)(Y_{knc}^f + nc)
\]

with \( Y_{ij}^f = 2nc \) for cycle 0. Note that \( Y_{ij}^f \) remains constant over the duration of any cycle.

Since any source node can get a maximum of \( nc \) new updates in a cycle of \( nc \) slots for its source file, \( N_{k+1}^f - N_{knc}^f \leq nc \). Using arguments similar in the proof of Theorem 1, allows us to conclude \( Y_{ij}^f \geq X_{ij}^f \) for all \( t \). In turn, \( E[Y_{ij}^f] \) can be upper bounded as in Theorem 1 as given below

\[
E[Y_{ij}^f] < \frac{cn}{(1-p)^2} + \frac{cn}{(1-p)} = O(n).
\]

A. An Achievable Gossiping Protocol: RLC

We now show that the random linear coding (RLC) gossip protocols proposed in [5] fall under the category of such
protocols, which work on the principle of algebraic mixing of files for faster simultaneous dissemination of all files. In RLC with PUSH protocol, each node creates a linear combination of all the coded packets in its possession at the beginning of each time slot, and transmits it to a randomly chosen target node. The coefficients are randomly chosen from the finite field \( \mathbb{F}_q \), with \( q \geq n \). In RLC with PULL protocol, nodes pull such random coded messages from their target nodes in every time slot. In both protocols, once a node has collected \( n \) independent linear combinations, it can decode all messages.

[5] Thm. 3.1 and Thm. 3.2 provide that both RLC protocols can disseminate \( n \) files to \( n \) nodes in \( O(n) \) time with high probability \( 1 - O(\frac{1}{n}) \). Since the probability \( 1 - O(\frac{1}{n}) \) is an increasing function of \( n \), both RLC protocols satisfy conditions of Theorem 2 in this paper.

IV. Numerical Results

We now further validate our results with numerical evaluations. First, we simulate single-file dissemination using the INTERLEAVE protocol in real-time over 200 cycles. We choose number of pieces \( \ell = \lceil \log_2 n \rceil \) and \( c = 18 \) in Section I. Since \( \ell \) changes value only when \( n \) crosses a power of two, we choose \( n = 2^m \), \( m \in \mathbb{N} \) to have \( \ell \) increase consistently with \( n \). We assume the source node receives version updates according to Bernoulli(0.7) distribution in every time slot. Fig. 2 shows the expected version age per node as a function of network size \( n \). We observe that as the network size grows larger, the version age does not increase, and remains below 9 for the choice of parameters above. For \( c = 18 \), the upper bound derived in Theorem 1 is \( \frac{c}{1-p} + \frac{c}{1-p} \geq 36 \), which shows that the derived upper bound is an overestimate.

Next, we use RLC with PUSH mechanism for real-time dissemination of \( n \) files in an \( n \) node network. The authors in [5] point out that their upper bound in [5] Thm. 3.2] is not tight and their simulations suggest that the dissemination time is close to \( 1.5n + \log_2 n \). We choose \( c = 6 \) and run the scheme of Section I over 100 cycles. We assume all nodes receive version updates for their respective source files according to Bernoulli(0.7) distribution in every time slot, and choose \( n \) to be prime numbers to keep field size \( q = n \) as in [5]. Fig. 3 shows that per node per file expected version age divided by network size \( n \), and we see the graph to be lower than 3 for above choice of parameters, which suggests that the upper bound in Theorem 2 (which is \( \geq 12 \)) is an overestimate. We note that, in \( n \)-file simulations, we consider all files to be successfully decoded by a node at the same time, when \( n \) linearly independent coded packets arrive at the node. For some \( r < n \), the node might receive \( r \) linearly independent combinations of \( r \) files and decode them before receiving \( n \) linearly independent combinations. Hence, Fig. 4 is an upper bound to the actual average version age.

V. Conclusion

We studied classes of gossip protocols which achieve \( O(1) \) age at each node in a single-file \( n \)-node system, and \( O(n) \) age for each file at each node in an \( n \)-file \( n \)-node system. We saw that the bounds hold even though files might not be successfully received by the nodes in every cycle, as each cycle presents a new opportunity to the nodes to get updated. We demonstrated that gossip protocols based on file splitting and network coding fall under the above category of protocols.
