Weak Amnesiac Flooding

Zahra Bayramzadeh*, Ajay D. Kshemkalyani†, Anisur Rahaman Molla‡, and Gokarna Sharma*
**Kent State University, Kent, OH 44242, USA.* {zbayramz@,sharma@cs.}kent.edu
†*University of Illinois at Chicago, Chicago, IL 60607, USA.* ajay@uic.edu
‡*Indian Statistical Institute, Kolkata 700108, India.* molla@isical.ac.in

***Abstract*—Flooding is a fundamental concept in distributed computing. In flooding, typically, a node forwards a message to its neighbors for the first time when it receives a message. Later if the node receives the same message again, it simply ignores the message and does not forward it. The nodes store a "message record" to ensure that the same message is not forwarded again. Hussak and Trehan introduced *amnesiac flooding* where nodes do not require to keep the message record. They established a surprising result that the amnesic flooding of a single ($k = 1$) message starting from some source node always terminates in bipartite graphs in e rounds and in non-bipartite graphs in $[e + 1, e + D + 1]$ rounds, where e is the eccentricity of the source node and D is the diameter of the graph. Recently, Hussak and Trehan introduced *dynamic* amnesiac flooding initiated in possibly multiple rounds with possibly multiple ($k > 1$) messages from possibly multiple source nodes. They showed that the *partial-send* case where a node only sends a message to neighbours from which it did not receive *any* message in the previous round and the *ranked full-send* case where a node sends some highest ranked message to all neighbors from which it did not receive *that* message in the previous round, both terminate. However, they showed that the *unranked full-send* case, where a node sends some random message (not necessarily the highest ranked message) to all the neighbors from which it did not receive *that* message in the previous round, does not terminate. In this paper, we show that the unranked full-send case also terminates, provided that diameter D is known to graph nodes. We further show that the termination time is $D \cdot (2k - 1)$ rounds in bipartite graphs and $(2D + 1) \cdot (2k - 1)$ rounds in non-bipartite graphs.**

***Index Terms*—Distributed Algorithms, Classic Flooding, Amnesiac Flooding, Termination, Time Complexity, Memory**

I. Introduction

Flooding is one of the fundamental and most useful primitives in distributed computing. In flooding, the task is to disseminate message(s) from source nodes to all the nodes of the network. Suppose a distinguished source node has a message θ initially. The goal is to disseminate θ to all the nodes of the network. In a synchronous, round-based distributed network, flooding is typically performed as follows: In the first round, the distinguished source node sends θ to all its neighbors. From the next round onwards, when a node receives θ for the first time, it sends a copy of θ to its neighbors (except the neighbors from which it receives θ). If it receives θ again, it doesn't do anything. This essentially requires each node in the network to maintain a "message record" of θ to indicate whether that node has seen θ in some previous round. If a node receives θ and it has a record that it has seen θ before, then it does not forward θ. This ensures that the node never floods θ twice. It is well-known that this *classic* flooding process always terminates and the number of rounds until termination is $D + 1$, the diameter of the network. The message record is of size at least 1 bit for a message.

Moving from single message flooding to multiple message flooding, the flooding approach for a single message has to be applied to each of the messages separately. Therefore, each node has to have the message record of at least 1 bit per message, i.e., $\Omega(k)$ bits for $k > 1$ messages, which may be a problem for resource-constrained devices [16], [17].

Hussak and Trehan [9] asked an interesting question for the single message flooding starting from a distinguished source node: What will happen if nodes do not keep the record of the message θ? Will the flooding process still terminate? Not keeping a record means that message travels on its own without depending on a message record. Not having a message record simplifies client-server application design as well as makes it scalable due to the fact that servers do not need to keep track of session information [18]. It will also provide fault tolerance even when network nodes crash.

Intuitively, if the nodes do not keep any record, they may forward the message again and again when received in subsequent rounds. Thus, the absence of a message record raises the possibility that θ may be circulated infinitely. Hussak and Trehan [9] formally studied flooding without the message record, calling it *amnesiac flooding*, and showed that the single message ($k = 1$) flooding that starts from a distinguished source node terminates in bipartite graphs in e rounds and in non-bipartite graphs in $[e + 1, e + D + 1]$ rounds, where e is the eccentricity of the source node. Using two rounds to initiate flooding with the second round dependant on the first, Turau [17] proved termination time of $e + 1$ rounds in any (non-bipartite) network, reducing the $e + D + 1$ rounds of [9] by D rounds. However, the dependency on the first two rounds makes the result from Turau [17] not truly amnesiac compared to Hussak and Trehan [9]. In the recent followup work, Hussak and Trehan [10] showed that the same termination time of e rounds in bipartite graphs and $[e + 1, e + D + 1]$ rounds in non-bipartite graphs can be achieved for a single message θ starting from multiple source nodes concurrently. Turau [18] gave an alternative detailed proof.

Additionally in [10], Hussak and Trehan considered *dynamic* amnesiac flooding of multiple $k > 1$ messages, where the messages may be initiated in possibly different rounds (i.e., not necessarily in the same first round) by different

978-1-6654-3281-8/21/$31.00 ©2021 IEEE

source nodes in the graph. Dynamic flooding arises in different real-world applications. One prominent example is disaster monitoring [18] where a distributed system of sensors is deployed to monitor a disaster event. As soon as sensors detect an event which may happen at different times for different sensors, they start flooding this information in the network. Furthermore, one source node may initiate multiple (different) messages (the source nodes may not be all different, i.e., $1 \leq k' < k$ source nodes for k messages). They considered the following three cases (problems) of dynamic amnesiac flooding in the synchronous message passing setting.

- *partial-send*: a node only sends a message to its neighbors from which it did not receive *any* message in the previous round.
- *ranked full-send*: a node sends some highest ranked message to all neighbors from which it did not receive *that* message in the previous round.
- *unranked full-send*: a node sends some random message (not necessarily the higest ranked message) to all neighbors from which it did not receive *that* message in the previous round.

Hussak and Trehan [10] showed that both the partial-send and ranked full-send problems terminate, but the unranked full-send problem does not terminate. In this paper, we show that the unranked full-send problem also terminates, provided that diameter D is known to network nodes. We further prove the termination time for the unranked full-send problem in both bipartite and non-bipartite graphs.

Overview of the Model and Results. Let the communication network be modeled as an undirected and unweighted but connected graph $G = (V, E)$, where V is the set of network nodes and $E \subseteq V \times V$ is the set of edges of G. The nodes are allowed to communicate through the edges of the graph G. We consider a *synchronous message passing*[1] network, where computation proceeds in synchronous rounds with a node performing the following three tasks in each round: (i) receive messages from its neighbors, (ii) perform local computation, and (iii) send messages to its neighbors. No message is lost in transit. The messages are assumed to have unique IDs (which may not necessarily be consecutive and the smallest message ID may not be 1). The message IDs are provided as an input. The unique may be assigned through a message generator based on a counter so that no two messages receive the same counter. A message θ is called globally i-th ranked if and only if the ID of θ is i-th largest among the IDs of all the messages in the set. The (global) rank of the messages is not known to graph nodes (i.e., the unranked problem), otherwise it becomes the ranked problem which terminates. We prove the following theorem for the unranked full-send problem.

Theorem 1 (unranked full-send). *Given a set $\{\theta^1, \ldots, \theta^k\}$ of $k > 1$ messages positioned on $1 \leq k' \leq k$ nodes of a network G initiated at possibly different rounds, the unranked full-send problem terminates in bipartite graphs in $D \cdot (2k-1)$ rounds and in non-bipartite graphs in $(2D+1) \cdot (2k-1)$ rounds*[2] *with each node storing $O(\log(\max\{k, D\}))$ bits, provided that the diameter D is known to the graph nodes.*

Comparison to Amnesiac and Classic Flooding. Nodes do not need to store any information in '***truly***' amnesiac flooding [9]. However, the assumption of graph nodes knowing D demands each graph node to keep $\lceil \log D \rceil$ bits record in memory. Therefore, the storage requirement for any algorithm knowing D is at least $\Omega(\log D)$ bits. The total storage $O(\log(\max\{k, D\}))$ bits at each node in our algorithm is due to the fact that it also uses a $wait$ variable which needs $O(\log k)$ bits. Therefore, our algorithm provides a trade-off between two parameters k and D regarding memory; $O(\log D)$ bits when $k = O(D)$ and $O(\log k)$ bits otherwise. In classic (non-amnesiac) flooding, nodes need to store at least $\Omega(k)$ bits to flood k different messages. Therefore, the memory requirement in our algorithm is a significant reduction compared to classic flooding when $k > \Omega(\log D)$.

The above comparison shows that our algorithm provides a '***weak***' variant of 'truly' amnesiac flooding [9], [17]. An interesting direction for future research is whether a weaker assumption than D is enough to make the unranked full-send problem terminate. Finally, we prove the termination time of our algorithm using the single message termination time of [9]. One interesting property of our algorithm is that if a better termination time is available for the single message flooding, then the termination time improves proportionally.

Techniques. Suppose all messages are initiated in the beginning of round 1. Knowing D, the proposed algorithm asks messages to start their flooding process in the interval of $(2D+1)$ rounds, i.e., at rounds $1, (2D+1)+1, 2 \cdot (2D+1)+1, \ldots, (k-1) \cdot (2D+1)+1$. Suppose the source nodes of $k > 1$ messages $\theta^1, \ldots, \theta^k$ know the rank (ID) of all the messages, say, $1, \ldots, k$, with message θ_i having rank i. Let us call this rank order as *global rank*. Knowing the global rank, θ^i can immediately decide how long to wait before starting the flooding process. Since it is known that a single message θ^i finishes flooding in $(2D+1)$ rounds [9] ($e \leq D$), all k messages finish flooding by $k \cdot (2D+1)$ rounds. That is, the ranked full-send problem terminates in $k \cdot (2D+1)$ rounds.

The challenge to overcome is when the source nodes do not know the global rank of the messages (the unranked problem). We devise an algorithm that takes into account local ranks of the messages (i.e., the positions in the ranks of the messages at a node) in deciding the wait time for the messages. Except the globally lowest ranked message, the wait time assigned at round 1 may not be equal to its wait time knowing its global rank. The algorithm asks locally lowest ranked messages to start amnesiac flooding at round $(\kappa-1) \cdot (2D+1)+1$, $\kappa \geq 1$ following the single message algorithm of Hussak and Trehan

[1] In the asynchronous message passing framework, it was shown by Hussak and Trehan [9] that amnesiac flooding does not terminate.

[2] If eccentricity $e_1, e_2, \ldots, e_{k'}$ of the k' source nodes is known instead of D, then the bounds translate to $e_{\max} \cdot (2k-1)$ in bipartite graphs and $(2e_{\max}+1) \cdot (2k-1)$ in non-bipartite graphs with memory $O(\log(\max\{k, e_{\max}\}))$ bits, where $e_{\max} := \max_{1 \leq l \leq k'} e_l$.

[9]. If the message that starts flooding at round $(\kappa-1)\cdot(2D+1)+1$, $\kappa \geq 1$, is globally κ ranked, we show that it terminates by round $\kappa \cdot (2D+1)$; otherwise during the round between $(\kappa-1)\cdot(2D+1)+2$ and $\kappa\cdot(2D+1)+1$ (inclusive), it finds that its global rank is higher than κ and starts waiting increasing its wait time proportional to its local rank at that time. We will also show that the wait time update stops at round $(\kappa'-1)\cdot(2D)+1$ for the globally κ' ranked message. This altogether guarantees that the algorithm terminates in $k \cdot (2D+1)$ rounds for $k > 1$ messages.

Finally, we show that this approach extends for messages initiated at different rounds with termination time $(2k-1)\cdot(2D+1)$. For bipartite graphs, the only change is replacing $(2D+1)$ with D so that the bound becomes $(2k-1)\cdot D$.

Other Related Works. Amnesiac flooding was first introduced and studied by Hussak and Trehan [9]. Recently, improved results and extensions were given in [10], [16], [17], [18], [19]. Turau [17] proved that the problem of selecting κ source nodes with minimal termination time for amensiac flooding is NP-hard. Particularly, Turau showed that unless NP = P there is no approximation algorithm for amnesiac flooding with approximation ratio $3/2-\epsilon$. For asynchronous systems, Turau proved that deterministic amnesiac flooding is only possible if a large enough part of the message can be updated by each node. Very recently, Turau [18] provided an alternative detailed proof for the single message flooding starting from multiple source nodes in the beginning of round 1. Specifically, Turau showed that, for every non-bipartite graph G and every set V' of source nodes that start flooding simultaneously, there exists a bipartite graph $G(V')$ such that the execution of amnesiac flooding on both graphs G and $G(V')$ is strongly correlated and termination times coincide. This led to bounds that are independent of the diameter as well as it allowed to determine source nodes for which amnesiac flooding terminates in minimal time. Turau also gave tight lower and upper bounds for the time complexity in special cases of $|V'|=1$ and $|V'|>1$. In fact, the case of $|V'|>1$ was reduced to the case of $|V'|=1$. Very recently, Turau [19] extended this line of work for amnesiac flooding in intermittent channels with bounded capacities.

Flooding is a fundamental concept used in solving a diverse set of fundamental problems in distributed computing, e.g., leader election [11], [12], spanning tree construction [2], [13], shortest paths computation [7], [8], [15], aggregation [4], routing [14], etc. Flooding of multiple messages is a must in, e.g., k-information dissemination or gossiping [1], [3], [13].

Amnesiac flooding uses the most recent edges from which the message is received to a node to decide which neighboring edges of that node are used to flood the message from that node. This concept finds applications and uses in social networks [5], broadcasting [6], and client-server application design [18]. More details in [9], [10], [16], [17], [18].

Roadmap. We discuss some preliminaries and formal problem definition in Section II. We prove our main result Theorem 1 in Section III. Finally, we conclude in Section IV with a short discussion on possible future work.

II. Preliminaries

Communication Network. We model the communication network as an undirected and unweighted but connected graph $G=(V,E)$. Each graph node $v \in V$ has a distinct ID. We assume that the graph nodes send/receive messages through the neighboring edges, i.e., when a node u sends a message through an edge $e=(u,v)$, then node v receives that message. We assume $\mathcal{CONGEST}$ model such that at any time only one message can be sent through edge e. The graph nodes know D, the diameter of G. The graph nodes have memory to store some limited information, such as D.

G is said to be *bipartite* if its vertices V can be divided into two disjoint and independent sets U and W such that every edge in G connects a vertex in U to one in W. If G does not satisfy this criteria, it is *non-bipartite* (a.k.a. *arbitrary*).

Messages. There are $k>1$ messages $\{\theta^1,\ldots,\theta^k\}$ positioned at graph nodes. We denote the initial position of message θ^i in G by a pair (v_i, r_i), where $v_i \in V$ denotes the node on which θ^i is positioned initially and $r_i \geq 1$ the round at which θ^i is initiated at v_i. No message can reside on the edges of G, but one or more messages can occupy the same node of G.

Communication Model. We consider a synchronous model where communication and computation proceeds in synchronous rounds or steps and a graph node $v \in V$ may perform the following three tasks in the order in each round $r_i \geq 1$.

- *Receive messages:* sent by its neighbor nodes through the adjacent edges connecting v to those neighbors. Only at most one message per edge is received.
- *Local computation:* v may perform an arbitrary local computation to decide which message (among the ones positioned on it) to send to neighbors and, if there is a message to send, which edges to use to send that message.
- *Send messages:* At the end of the round, v sends the message it decided to send (if any) using the computed edges (to reach to the neighbors of v).

We assume that the nodes are fault-free and no message is lost in transit. Given the communication model above, time to termination is measured in rounds or steps. Another important parameter is memory which comes from a single source – the number bits stored at each graph node.

Dynamic Amnesiac Flooding. Let the message θ^i be initially positioned on node v_i in the beginning of round $r_i \geq 1$. Node v_i sends θ^i to all its neighbors at round r_i. At round r_i+1 and after, it is done as follows. Let $N(v)$ be the set of all neighbors of a node $v \in V$. Let $N_{l,j}(v)$ be the set of neighbors of v from which v receives message θ^j in the beginning of round r_l.

Definition 1 (partial-send). *Suppose a node v receives messages from its neighbors in the beginning of round $r_l \geq 2$. Let the message θ^j be such that $N_{l,j}(v) \neq \emptyset$. Then, at the end of round r_l, v sends θ^j to all its neighbors $N(v)\backslash \cup_{h\geq 1} N_{l,h}(v)$. In case of multiple messages satisfy that $N_{l,j}(v) \neq \emptyset$, pick arbitrarily one among them.*

Definition 2 (ranked full-send). *Suppose a node v receives messages from its neighbors in the beginning of round $r_l \geq 2$. Let the rank of each message θ^i be $rank(\theta^i)$. Let the message θ^j be the largest ranked message such that $N_{l,j}(v) \neq \emptyset$. Then, at the end of round r_l, v sends θ^j to all its neighbors $N(v) \backslash N_{l,j}(v)$.*

Definition 3 (unranked full-send). *Suppose a node v receives messages from its neighbors in round $r_l \geq 2$. Let the message θ^j be such that $N_{l,j}(v) \neq \emptyset$. Then, in round r_l, v sends θ^j to all its neighbors $N(v) \backslash N_{l,j}(v)$. In case of multiple messages satisfy that $N_{l,j}(v) \neq \emptyset$, pick arbitrarily one among them.*

III. Unranked Full-send Algorithm

We describe an algorithm that solves unranked full-send problem provided that the diameter D of graph G is known to the nodes. Note that the algorithm works even if a finite upper bound on D or number of nodes n is known, but for simplicity we assume that D is known. Consider a set $\Theta = \{\theta^1, \ldots, \theta^k\}$ of k messages positioned initially arbitrarily on the nodes of graph G in the beginning of round 1; we discuss later how to deal with the case of messages initiating at different rounds $r \geq 1$. Suppose, w.l.o.g. the messages have unique integer IDs, which may not be consecutive and the smallest ID need not be 1. The message with largest ID is called the k-th ranked or the largest message and denoted by θ^k. Similarly, the message with smallest ID is called the 1st ranked or the smallest message and denoted by θ^1. Message $\theta^i \in \Theta, 1 \leq i \leq k$, is called the i-th ranked message, meaning that there are exactly $i-1$ other messages in Θ ranked smaller than θ^i. If we say θ then it can be any message in Θ.

The pseudocode is given in Algorithm 1. Consider a (global) i-th ranked message $\theta^i \in \Theta$. $\theta^i \in \Theta$ does not know that it has (unique) global rank i among the messages in Θ. Therefore, a major challenge in Algorithm 1 is how θ^i correctly finds its (global) rank i, so that it can finish flooding applying single-message algorithm of Hussak and Trehan [9].

Algorithm 1 uses the concept of ***local rank*** for each message $\theta^i \in \Theta$ to be able to eventually find its global rank i. The local rank for θ^i at any round $r \geq 1$ is computed as follows. Suppose θ^i is positioned at node w at round r. Let $\Theta_r(w) \subseteq \Theta$ denote the messages in Θ that are on node w at round r. If θ^i is j-th largest (w.r.t. ID) among the messages in $\Theta_r(w)$, then its local rank at w at round r is j.

We start the discussion of the algorithm with what happens in the first round $r = 1$. At round $r = 1$, θ^i computes its local rank and sets its wait time $wait(\theta^i) = (j-1) \cdot (2D+1)+1$, proportional to its local rank j. Since rank j is local for θ^i at node w, j may be smaller than i (i.e., j may not represent the (global) rank i for θ^i). The implication is that $wait(\theta^i)$ at round 1 may get updated later depending on the other messages of Θ that θ^i gets aware during the execution of Algorithm 1. It is interesting to note also that each update on $wait(\theta^i)$ increases its value by an additive factor $\kappa \cdot (2D+1), \kappa \geq 1$, which will be more clear as we proceed with the algorithm discussion in the following.

Algorithm 1: Unranked full-send (for message $\theta^i \in \Theta$ at some node $w \in G$)

```
1  if $r = 1$ then
2      $j \leftarrow$ rank of $\theta^i$ among the messages on the node $w$;
3      $wait(\theta^i) \leftarrow (j-1) \cdot (2D+1) + 1$;
4  for $2 \leq r < wait(\theta^i)$ do
5      $j \leftarrow$ rank of $\theta^i$ at $w$ at $r$;
6      if $j > 1$ then
7          $\theta \leftarrow$ 1-st ranked message at $w$ at $r$;
8          if $(r \mod (2D+1)) = 0$ then
9              if $wait(\theta) > r$ then
10                 $wait(\theta^i) \leftarrow r + (j-1) \cdot (2D+1) + 1$;
11             if $wait(\theta) < r$ then
12                 $wait(\theta^i) \leftarrow r + (j-2) \cdot (2D+1) + 1$;
13         if $(r \mod (2D+1)) \neq 0$ then
14             if $wait(\theta) > r$ then
15                 $wait(\theta^i) \leftarrow r - (r$
                     $\mod (2D+1)) + j \cdot (2D+1) + 1$;
16             if $wait(\theta) < r$ then
17                 $wait(\theta^i) \leftarrow r - (r$
                     $\mod (2D+1)) + (j-1) \cdot (2D+1) + 1$;
18 if $r = wait(\theta^i)$ and the edge set $E^{in}_{i,wait(\theta^i)} = \emptyset$ then
19     $j \leftarrow$ rank of $\theta^i$ at $w$ at $r$;
20     if $j > 1$ then
21         $wait(\theta^i) \leftarrow r - 1 + (j-1) \cdot (2D+1) + 1$;
22     else
23         run the algorithm of Hussak and Trehan [9];
24         if $\theta^i$ meets some other message $\theta$ at some node $w$ in any
             round $wait(\theta^i) + 1 \leq r < wait(\theta^i) + (2D+1)$ then
25             $j \leftarrow$ rank of $\theta^i$ at $w$;
26             if $j > 1$ then
27                 if $(r \mod (2D+1)) = 0$ then
28                     $wait(\theta^i) \leftarrow r + (j-2) \cdot (2D+1) + 1$;
29                 if $(r \mod (2D+1)) \neq 0$ then
30                     $wait(\theta^i) \leftarrow r - (r$
                         $\mod (2D+1)) + (j-1) \cdot (2D+1) + 1$;
31                 stop the algorithm of Hussak and Trehan [9] and
                     set the edge set $E^{in}_{i,r} \leftarrow \emptyset$;
32         if $\theta^i$ meets a copy of itself, say $\theta^{i,c}$, at some node $w$ in any
             round $wait(\theta^i) + 2 \leq r \leq wait(\theta^i) + (2D+1)$ then
33             if $wait(\theta^{i,c}) > wait(\theta^i)$ then
34                 stop the algorithm of Hussak and Trehan [9] and
                     set the edge set $E^{in}_{i,r} \leftarrow \emptyset$;
35                 merge with $\theta^{i,c}$ and act as the same message $\theta^i$;
36 if $r = wait(\theta^i) + (2D+1) + 1$ and $\theta^i$ has ther edge set
     $E^{in}_{i,wait(\theta^i)+(2D+1)+1}(w) \neq \emptyset$ then
37     Set ther edge set $E^{in}_{i,wait(\theta^i)+(2D+1)+1}(w) \leftarrow \emptyset$;
38     $j \leftarrow$ rank of $\theta^i$ at $w$ at $r$;
39     $wait(\theta^i) \leftarrow r - 1 + (j-1) \cdot (2D+1) + 1$;
40     if $r - wait(\theta^i) = 0$ then execute Lines 18-35;
```

Let Θ_r^γ denotes the γ number of messages that are on the nodes of G at some round r. In the initial configuration, the messages in Θ_1^γ were on $\gamma \leq k$ different nodes of G. Denote the γ message set, 1 message per node, by Θ_1^γ. In round $r = 1$, only the messages in the set Θ_1^γ will have their local rank $j = 1$. Each message $\theta^i \in \Theta_1^\gamma$ sets its wait time $wait(\theta^i) = (j-1) \cdot (2D+1) + 1$, which is 1, in round $r = 1$. Each message

$\theta^i \in \Theta_1^\gamma$ starts flooding using the algorithm of Hussak and Trehan [9] in round $r = 1$ as $wait(\theta^i) = r = 1$.

It is immediate that only the messages in Θ_1^γ run the flooding algorithm from round $r = 1$ until $r = (2D + 1)$ because any message $\theta \in \Theta \setminus \Theta_1^\gamma$ has its wait time $wait(\theta) \geq (2D+1)+1$. Therefore, we now discuss all possible situations for the messages in Θ_1^γ from round $r = 2$ until round $r = (2D + 1)$. We need a notion of meeting.

Definition 4 (meeting). *We say that a* meeting *happens between two messages θ^i, θ^j in a round $r \geq 2$ when either*

i. *both θ^i, θ^j enter the empty node $w \in G$ at the end of round $r - 1$ or*
ii. *θ^i (or θ^j) enters a node $w \in G$ where θ^j (or θ^i) is positioned at the end of round $r - 1$.*

It is immediate from the algorithm of [9] that if a meeting does not happen for a message $\theta^i \in \Theta_r^\gamma$ in all the rounds from $r = 2$ until $r = (2D+1)$ (inclusive), θ^i finishes flooding. Note that meeting cannot happen in round $r = 1$ as the flooding starts at that round. However, there may be the case that at least a meeting happens at some round $2 \leq r \leq (2D + 1)$ for message $\theta^i \in \Theta_1^\gamma$. We now discuss the execution scenario in all the rounds $2 \leq r \leq (2D + 1)$ for a message θ^i. We consider four cases:

Case (a): $\theta^i \in \Theta_1^\gamma$ and it meets (at least) one other message in Θ_1^γ but no message in $\Theta \setminus \Theta_1^\gamma$ at some node $w \in G$ in round $2 \leq r \leq (2D + 1)$: All the messages at w that θ^i meets (including θ^i) at round r have the same waiting time of 1. θ^i computes its local rank at w at round r. If θ^i has rank $j = 1$, it continues with the algorithm of Hussak and Trehan [9]. Otherwise, θ^i starts to wait at w starting from round r setting its wait time $wait(\theta^i)$ as follows. Suppose θ^i has rank $j > 1$ among the messages at w. If $r = (2D + 1)$, θ^i sets $wait(\theta^l) = r + (j-2) \cdot (2D+1) + 1$, otherwise ($r < 2D+1$) θ^i sets $wait(\theta^l) = r - (r \mod (2D+1)) + (j-1) \cdot (2D+1) + 1$. Lines 24-31 of Algorithm 1 illustrates this case.

Case (b): $\theta^i \in \Theta_1^\gamma$ and it meets (at least) one other message in $\Theta \setminus \Theta_1^\gamma$ but no message in Θ_1^γ at some node w in round $2 \leq r \leq 2D + 1$: message $\theta^i \in \Theta_1^\gamma$ that met message θ^l can easily figure out that whether θ^l is in $\Theta \setminus \Theta_1^\gamma$ or in Θ_1^γ. The main reason is that if $\theta^l \in \Theta \setminus \Theta_1^\gamma$ then $wait(\theta^l) \geq wait(\theta^i) + (2D+1)$ but if $\theta^l \in \Theta_1^\gamma$ then $wait(\theta^l) = wait(\theta^i)$. If θ^i is ranked $j = 1$, it continues with the algorithm of Hussak and Trehan [9]. Otherwise, θ^i starts to wait at w starting from round r setting its wait time $wait(\theta^i)$ as follows. Suppose θ^i has rank $j > 1$ among the messages in w. If $r = (2D+1)$, θ^i sets $wait(\theta^i) = r + (j-2) \cdot (2D+1) + 1$, otherwise ($r < 2D+1$) θ^i sets $wait(\theta^l) = r - (r \mod (2D+1)) + (j-1) \cdot (2D+1) + 1$. Lines 24-31 of Algorithm 1 illustrates this case.

Case (c): $\theta^i \in \Theta_r^\gamma$ and it meets at least one message in Θ_1^γ and at least one message in $\Theta \setminus \Theta_1^\gamma$ at some node w in some round $2 \leq r \leq (2D + 1)$: This is the combination scenario of the cases (a) and (b) where we dealt with these scenarios separately. If θ^i is ranked $j = 1$, then θ^i continues with the algorithm of Hussak and Trehan [9]. If θ^i is ranked $j > 1$, θ^i waits at w as follows. If $r = (2D + 1)$, θ^i sets $wait(\theta^i) = r + (j-2) \cdot (2D+1) + 1$, otherwise ($r < 2D+1$) θ^i sets $wait(\theta^l) = r - (r \mod (2D+1)) + (j-1) \cdot (2D+1) + 1$. Lines 24-31 of Algorithm 1 illustrates this case.

Case (d): $\theta^i \in \Theta \setminus \Theta_r^\gamma$ (waiting at some node w) and it meets at least one message in Θ_1^γ in some round $2 \leq r \leq (2D + 1)$: Suppose θ^i meets exactly one message $\theta \in \Theta_1^\gamma$ at round $2 \leq r \leq (2D + 1)$. Since θ^i is waiting at w, $wait(\theta^i) \geq wait(\theta) + (2D + 1)$. Furthermore, θ is running the algorithm of Hussak and Trehan [9]. Therefore, θ^i does nothing. Now suppose θ^i meets two or more messages in Θ_1^γ. All the messages in Θ_1^γ that θ^i met at w (except the lowest ranked one in Θ_1^γ) start to wait at w (Case (c) handles what a message in Θ_1^γ does at w). Therefore, suppose θ^i is ranked $j > 1$ among the messages at w. Note that θ^i cannot be ranked $j = 1$ at w since it is waiting at w. Let $\theta \neq \theta^i$ be the message at w with rank $j = 1$. θ^i sets its wait time depending on whether θ belong to Θ_1^γ or $\Theta \setminus \Theta_1^\gamma$.

Consider the case where θ belongs to Θ_1^γ (i.e., $wait(\theta) < r$). If $r = (2D+1)$, θ^i sets $wait(\theta^i) = r + (j-2) \cdot (2D+1) + 1$, but if $r \neq (2D + 1)$, then θ^i sets $wait(\theta^i) = r - (r \mod (2D + 1)) + (j - 1) \cdot (2D + 1) + 1$.

Consider the case where θ belongs to $\Theta \setminus \Theta_1^\gamma$ (i.e., $wait(\theta) > r$). If $r = (2D + 1)$, θ^i sets $wait(\theta^i) = r + (j - 1) \cdot (2D + 1) + 1$, but if $r \neq (2D + 1)$, then θ^i sets $wait(\theta^i) = r - (r \mod (2D + 1)) + j \cdot (2D + 1) + 1$. Lines 4–17 of Algorithm 1 illustrate this case. There are two additional situations θ^i may need to deal with in some round $2 \leq r \leq (2D + 1)$ if it belongs to Θ_1^γ, which we discuss below.

If $\theta^i \in \Theta_1^\gamma$ and starts waiting at node w at round $2 \leq r \leq 2D + 1$: θ^i clears at that round r the information about the edges in the set $E_{i,r}^{in}(w)$; $E_{i,r}^{in}(w)$ denotes the set of adjacent edges of w from which θ^i entered w in the beginning of round r.

If $\theta^i \in \Theta_1^\gamma$ and finds (at least) a copy of itself, say $\theta^{i,c}$ at node w at some round $2 \leq r \leq 2D + 1$: If a copy $\theta^{i,c}$ of θ^i has $wait(\theta^{i,c}) > wait(\theta^i)$, then that copy $\theta^{i,c}$ must have reached w prior to θ^i in some round $2 \leq t \leq r-1$ and it is now waiting at w since one of the cases (a), (b) or (c) applied to $\theta^{i,c}$ at round t. θ^i stops the algorithm of Hussak and Trehan [9], clears the information about $E_{i,r}^{in}(w)$, and merges with $\theta^{i,c}$ so that the both copies act as a single message θ^i. However, if all the copies of θ^i have same $wait(\theta^i)$, they must have reached w in the beginning of round r. They act as a single copy and proceed with cases (a), (b), or (c) for the rank calculation and decision on whether to wait at w or to continue flooding.

The discussion so far captures all possible situations for θ^i in any round $1 \leq r \leq (2D+1)$. We will show in our analysis that if θ^i is globally ranked 1st then it will finish flooding by round $2D + 1$. Furthermore, we will show in the analysis that no message in the set $\Theta \setminus \Theta_{r=1}^\gamma$ (messages that were not ranked $j = 1$ at round $r = 1$) runs the flooding algorithm of [9] in any round $r \leq (2D + 1)$.

Let $\Theta_1^{\gamma'}$ be the messages in the set Θ_1^γ that started flooding at round $r = 1$ but could not finish flooding by round $2D + 1$ (they either started waiting at some round $1 < r \leq (2D + 1)$ or still running the algorithm). We have that $|\Theta_1^{\gamma'}| \leq |\Theta_1^{\gamma}| - 1$. Furthermore, $\Theta_1^{\gamma'}$ can be divided into two sets $\Theta_{1,w}^{\gamma'}$ and $\Theta_{1,nt}^{\gamma'}$, where $\Theta_{1,w}^{\gamma'} \subset \Theta_1^{\gamma}$ are the messages that started flooding at round $r = 1$ but switched to waiting at some round $2 \leq r \leq (2D + 1)$, and $\Theta_{1,nt}^{\gamma'} \subset \Theta_1^{\gamma}$ are the messages that started flooding at round $r = 1$ but their flooding did not finish by round $r = (2D + 1)$. Let

$$\Theta_{2D+1+1} := \Theta_1^{\gamma'} \cup \{\Theta \setminus \Theta_1^{\gamma}\} = \Theta \setminus \{\Theta_1^{\gamma} \setminus \Theta_1^{\gamma'}\} \subset \Theta.$$

Note that Θ_{2D+1+1} has all the messages in Θ except the ones finished flooding by round $2D + 1$. In the worst-case, $\Theta_{2D+1+1} = \Theta \setminus \{\theta^1\}$. The goal is to go back to configuration similar to round $r = 1$ in round $r = (2D + 1) + 1$ for the messages in Θ_{2D+1+1}. We would like to guarantee that a fresh new flooding can be started for the messages in Θ_{2D+1+1} in round $r = (2D + 1) + 1$. However, the messages in the set $\Theta_{1,nt}^{\gamma'} \subset \Theta_1^{\gamma}$ have not finished flooding yet. To deal with this issue, before starting the flooding at round $(2D + 1) + 1$, for each message $\theta^i \in \Theta_{1,nt}^{\gamma'}$, it is asked to perform the following two steps one after another (Lines 36–40 of Algorithm 1 handle this case).

(i) θ^i stops its flooding process, and clears the information about $E^{in}_{i,2D+1+1}(w)$, the adjacent edges of w from which it entered w in the beginning of round $2D + 1 + 1$. θ^i can easily figure out at round $r = (2D + 1) + 1$ that it belongs to $\Theta_{1,nt}^{\gamma'}$, since it must have wait time $wait(\theta^i) = r - (2D + 1)$ and $E^{in}_{i,2D+1+1}(w) \neq \emptyset$.

(ii) θ^i calculates its local rank j among the messages in w. θ^i sets its wait time $wait(\theta^i) \leftarrow r-1+(j-1)\cdot(2D+1)+1$; note that $r = (2D + 1) + 1$.

Suppose $\theta^i \notin \Theta_{1,nt}^{\gamma'}$ and $wait(\theta^i) = (2D+1)+1$. θ^i should be able to run the algorithm of [9] at round $r = (2D+1)+1$. However, some other message $\theta^j \in \Theta_{1,nt}^{\gamma'}$ may happen to be on the same node w where θ^i is positioned on round $r = (2D + 1) + 1$. If θ^j is ranked $j = 1$ on w, θ^i must not start flooding and wait further. Therefore, if $wait(\theta^i) = (2D + 1) + 1$, then θ^i checks its local rank at w. If it is ranked $j > 1$ on w at round $(2D + 1) + 1$ then it updates its wait time $wait(\theta^i) = r - 1 + (j - 1) \cdot (2D + 1) + 1$ and waits further on w (Lines 18–21 of Algorithm 1 handle this case). After all this, if $\theta^i \in \Theta_{2D+1+1}$, $wait(\theta) \geq \kappa \cdot (2D + 1)$, with $k \geq 1$.

Let Θ^{ϕ}_{2D+1+1} be the messages in the set Θ_{2D+1+1} that are locally 1-th ranked in the nodes of G, i.e., $\Theta^{\phi}_{2D+1+1} \subseteq \Theta_{2D+1+1}$. Θ^{ϕ}_{2D+1+1} are the messages with wait time exactly $2D + 1$ (exactly one per non-empty node of G). If $\theta^i \in \Theta_{2D+1+1} \setminus \Theta^{\phi}_{2D+1+1}$, then it does not start flooding from $r = (2D + 1) + 1$ until $r = 2 \cdot (2D + 1)$. Therefore, we now discuss what happens from $r = (2D + 1) + 1$ until $r = 2 \cdot (2D + 1)$ for θ^i when $\theta^i \in \Theta^{\phi}_{2D+1+1}$. θ^i starts the flooding algorithm of Hussak and Trehan [9] in round $r = (2D + 1) + 1$. We can again consider four cases for θ^i from round $(2D+1)+2 \leq r \leq 2\cdot(2D+1)$. Note that θ^i does not meet any other message at $r = (2D + 1) + 1$ since all the messages in Θ^{ϕ}_{2D+1+1} start flooding at $r = (2D+1)+1$. The techniques discussed for cases (a), (b), (c), or (d) for $2 \leq r \leq (2D + 1)$ apply to θ^i for $(2D + 1) + 2 \leq r \leq 2 \cdot (2D + 1)$. Therefore, if θ^i is the globally 2-nd ranked message in Θ, it will finish flooding by $r = 2 \cdot (2D + 1)$.

At round $r = 2\cdot(2D+1)+1$, we have configuration similar to $r = (2D + 1) + 1$. Let $\Theta_{2\cdot(2D+1)+1}$ be the messages in Θ except ones finished flooding by round $r = 2 \cdot (2D + 1)$. In the worst-case, $\Theta_{2\cdot(2D+1)+1} = \Theta \setminus \{\theta^1, \theta^2\}$, as the globally 2-nd ranked message θ^2 finishes flooding by $r = 2\cdot(2D+1)$. If $\theta^i \in \Theta_{2\cdot(2D+1)+1}$, it performs the steps similar to $r = (2D + 1) + 1$ and either start flooding using the algorithm of [9] or start waiting further updating $wait(\theta^i)$.

Thus, at any round $r = \kappa \cdot (2D + 1)$, $\kappa \geq 0$, κ messages from the set Θ of k messages globally ranked 1 to κ finish flooding. The globally ranked $\kappa+1$ to k messages in Θ either start flooding at round $r = \kappa \cdot (2D + 1) + 1$ or wait until $r = \kappa' \cdot (2D + 1) + 1$, where $\kappa < \kappa' < k - 1$ and finish flooding by $r = (\kappa' + 1) \cdot (2D + 1)$.

A. Analysis of the Algorithm

We now analyze Algorithm 1 for its correctness, termination time (number of rounds), and memory requirement (number of bits). We start with the following lemma.

Lemma 1. *At any round $r \geq 1$, the wait time for message $\theta^i \in \Theta$ is $wait(\theta^i) = \kappa \cdot (2D + 1) + 1$ with $\kappa \geq 0$.*

Proof. At round 1, each message $\theta^i \in \Theta$ positioned on some node $w \in G$ is assigned $wait(\theta^i) = (j - 1) \cdot (2D + 1) + 1$, where j is the rank of θ^i among the messages $\Theta(w)$ on w. This $wait(\theta^i)$ is equal to $\kappa\cdot(2D+1)+1$ for $\kappa = (j-1) \geq 0$.

Now consider $wait(\theta^i)$ at any round $r > 1$. The wait time for θ^i at round $r > 1$ is again $wait(\theta^i) = r-(r \mod (2D+1)) + (j-1)\cdot(2D+1)+1$. We have that $r-(r \mod (2D+1)) = \kappa'\cdot(2D+1)$, with an integer $\kappa' \geq 1$. Thus, $wait(\theta^i) = r-r \mod (2D+1)+(j-1)\cdot(2D+1)+1 = \kappa'\cdot(2D+1)+(j-1)\cdot(2D+1)+1 = (\kappa'+j-1)\cdot(2D+1)+1 = \kappa\cdot(2D+1)+1$, where $\kappa = \kappa' + (j - 1)$. □

Lemma 2. *If message $\theta^i \in \Theta$ starts flooding then that round is $\kappa \cdot (2D + 1) + 1$ with $\kappa \geq 0$.*

Proof. Consider any message $\theta^i \in \Theta$. Suppose θ^i is positioned on node $w \in G$. Let $\Theta(w)$ be the messages positioned on w; since $\theta^i \in \Theta(w)$, $|\Theta(w)| \geq 1$. At round $r = 1$, θ^i starts flooding from node w if it is the 'locally' 1-ranked message in the set $\Theta(w)$. In this case $wait(\theta^i) = \kappa \cdot (2D + 1) + 1$ with $\kappa = 0$, otherwise, $wait(\theta^i) = \kappa \cdot (2D + 1) + 1$ with $\kappa \geq 1$. Suppose θ^i reaches to some node x in round $r = 2$. If θ^i is 'locally' 1-ranked among all the messages $\Theta(x)$ on x, then it continues flooding from node x. However, if θ^i is not the 'locally' 1-st ranked message at node x, θ^i updates its wait time such that the updated wait time $wait(\theta^i) = r - r \mod (2D+1)+(j-1)\cdot(2D+1)+1 = \kappa''\cdot(2D+1)+(j-1)\cdot(2D+1)+1 = (\kappa''+j-1)\cdot(2D+1) = \kappa'\cdot(2D+1)+1$ for $\kappa' = \kappa''+j-1$, since $r-(r \mod (2D+1)) = \kappa''\cdot(2D+1)$

for any $r \geq 1$. Therefore, message θ^i always starts its flooding process at round $\kappa \cdot (2D+1)+1$ with $\kappa \geq 0$. □

Lemma 3. *For any $\kappa \geq 0$, consider a subset $\Theta_{\kappa\cdot(2D+1)+1} \subseteq \Theta$ of messages that start flooding at round $r = \kappa \cdot (2D+1) + 1$. At least a message in $\Theta_{\kappa\cdot(2D+1)+1}$ finishes flooding and terminates by round $r = (\kappa+1)\cdot(2D+1)$.*

Proof. Let θ^1 be the 1-st ranked message in Θ. We first show that, in round 1, $\theta^1 \in \Theta_1$ (for $\kappa = 0$, $\kappa\cdot(2D+1)+1 = 1$). The argument is as follows: Let $w \in G$ be the node where message θ^1 is positioned in the initial configuration and hence in round 1. θ^1 will be 'locally' ranked 1-st among the messages on node w in round 1 because its global rank in Θ and its local rank on node w match. Therefore, $\theta^1 \in \Theta_1$.

We would like to show that θ^1 finishes flooding by round $r = (\kappa+1)\cdot(2D+1) = (2D+1)$. The proof is immediate if θ^1 does not meet any other message in Θ in all the rounds $2 \leq r \leq (2D+1)$ from the termination proof of the algorithm of Hussak and Trehan [9]. However, even if θ^1 meets other messages at some node x at some round r, the local rank of θ^1 at node x remains 1 among the messages in $\Theta(x)$. Therefore, θ^1 will run the flooding process uninterruptedly. The flooding of the messages in Θ_1 that were interrupted in $2 \leq r \leq (2D+1)$ cannot start their new flooding process in any round $r < (2D+1)+1$ since their updated wait time becomes at least $(2D+1)+1$.

Let $\Theta_{(2D+1)+1} \subset \Theta$ (for $\kappa = 1$) be the messages in Θ that start flooding at $r = (2D+1)+1$. Let θ^2 be the 2-ranked message in Θ. It is easy to prove similar to θ^1 that $\theta^2 \in \Theta_{(2D+1)+1}$ and θ^2 is the lowest ranked message among the messages in Θ that have not finished their flooding by $r \leq (2D+1)$. Therefore, θ^2 floods uninterruptedly starting at $r = (2D+1)+1$ and finishes by $r = 2\cdot(2D+1)$.

Furthermore, at round $r = (2D+1)+1$, if θ^2 is at only a single node of G, then the proof of the algorithm of Hussak and Trehan [9] provides correctness of flooding and termination by $r = 2\cdot(2D+1)$. However, at $r = (2D+1)+1$, if θ^2 is at two or more nodes of G, then it must locally 1st ranked at all the nodes it is positioned. θ^2 starts flooding from round $r = (2D+1)+1$ concurrently from all the nodes it is positioned. Hussak and Trehan [10] recently showed that θ^2 finishes flooding and terminates in time $\leq (2D+1)$ rounds, even when starting from multiple nodes. Therefore, θ^2 finishes flooding and terminates by $r = 2\cdot(2D+1)$ in our algorithm.

Arguing similarly, $\Theta_{(i-1)\cdot(2D+1)+1} \subset \Theta$ be the messages in Θ that start flooding at round $r = (i-1)\cdot(2D+1)+1$. The i-th ranked message $\theta^i \in \Theta$ becomes the 1-st ranked message in $\Theta_{(i-1)\cdot(2D+1)+1}$ (with $\kappa = (i-1)$) to start flooding at round $r = (i-1)\cdot(2D+1)+1$ and finishes its flooding process by round $r = i\cdot(2D+1)$. □

Lemma 4. *Let message $\theta^i \in \Theta$ sets (or updates) its wait time $wait(\theta^i)$ at round $r \geq 1$. If θ^i is positioned on node $w \in G$ at round r, it will be on w until round $r = wait(\theta^i)$.*

Proof. Suppose θ^i starts waiting at node w at round $t \geq 1$. It is immediate from Algorithm 1 that θ^i does not start flooding in any round from $r = t$ to $r = wait(\theta^i)$. Furthermore, except in flooding, θ^i does not leave the node it is positioned. □

Lemma 5. *Consider message $\theta^i \in \Theta$ with (global) rank $i \geq 1$. If θ^i sets $wait(\theta^i) = (i-1)\cdot(2D+1)+1$ in round 1, then $wait(\theta^i)$ never gets updated.*

Proof is omitted due to space constraints.

Lemma 6. *Consider message $\theta^i \in \Theta$ with (global) rank is $i \geq 1$, $wait(\theta^i)$ never grows more than $(i-1)\cdot(2D+1)+1$.*

Proof is omitted due to space constraints.

Lemma 7. *If message $\theta \in \Theta$ starts flooding at round $t = (\kappa-1)\cdot(2D+1)+1$ for some $\kappa \geq 0$ but does not finish its flooding process by round $t+(2D+1)$ then*

- *θ has global rank $> \kappa$ in Θ.*
- *At least one copy of θ, say θ^c, must be positioned on some node $w \in G$ with $wait(\theta^c) \geq t + \ell\cdot(2D+1)+1, \ell \geq 1$.*

Proof. If message θ starts flooding at round $t = (\kappa-1)\cdot(2D+1)+1$ and θ is κ-ranked among the messages in Θ, it will finish flooding by $t+(2D+1)$. Therefore, if θ does not finish flooding by $t+(2D+1)$, then it must be the case that θ is the κ'-ranked among the messages in Θ with $\kappa' > \kappa$.

Note also that if θ starts flooding at round $t = (\kappa-1)\cdot(2D+1)+1$ and θ is κ-ranked among the messages in Θ, then its flooding never gets interrupted by the meeting of any other message in Θ in all the rounds $t+1$ upto $t+(2D+1)$. Since θ is $\kappa' > \kappa$ ranked, when it is interrupted at some node w with $< \kappa'$-ranked message, it will stay at w. Due to this interruption, the other copies, even if run uninterrupted the flooding for $t+(2D+1)$, some of them may not enter a particular node at the same round and they may not terminate. □

Lemma 8. *Suppose message θ that is globally κ-ranked starts flooding at round $t = (\ell-1)\cdot(2D+1)+1$, with $\ell < \kappa$ and becomes unsuccessful in finishing its flooding by round $t+(2D+1)$. The unsuccessful flooding process of θ does not interrupt the flooding process of other messages ranked $\delta < \kappa$. Furthermore, there can be at most $(\kappa-\ell)$ unsuccessful flooding attempts by θ before it eventually becomes successful when it starts it flooding at round $t = (\kappa-1)\cdot(2D+1)+1$.*

Proof. If θ is globally κ-ranked and starts its flooding at round $t = (\kappa-1)\cdot(2D+1)+1$, then from Lemmas 5 and 6, θ finishes flooding by round $t+(2D+1)$. Therefore, if θ starts flooding at round $t = (\ell-1)\cdot(2D+1)$, then there are $\kappa-\ell$ messages ranked between ℓ and κ and they may interrupt the flooding of κ every time κ performs flooding. □

Theorem 2. *Given any initial configuration of a set $\Theta = \{\theta^1,\ldots,\theta^k\}$ of k messages positioned on the nodes of an arbitrary graph G in the beginning of round 1, Algorithm 1 solves flooding in $k\cdot(2D+1)$ rounds, given that nodes know D with messages being oblivious to their global ranks.*

Proof. The proof follows combining Lemmas 1–8. □

B. *The Case of Messages Initiated at Different Rounds*

Note that Algorithm 1 considers all $k > 1$ messages initiated in the beginning of round 1. We describe here how Algorithm 1 applies to the case of messages initiated at any round $r \geq 1$. We assume that at least a message is initiated at round 1. If this is not the case, we can start Algorithm 1 starting from the round the first message in Θ is initiated. We also assume that at least a new message is initiated within the interval of $(2D+1)$ rounds, otherwise Algorithm 1 will be idle with no message to flood after $(2D+1)$ rounds of the current message until a new message is initiated.

Suppose $\theta^i \in \Theta$ be the message that is initiated at round 1 and $\theta^j \in \Theta$ be the another message initiated at some round $1 \leq r \leq (2D+1)+1$. If θ^i has rank lower than θ^j, then the flooding of θ^i will not be interrupted by θ^j and θ^i finishes flooding by round $r = (2D+1)$. However, if θ^i has rank higher than θ^j then the flooding of θ^i may be interrupted by θ^j. Since there are k messages, the flooding of a message that is initiated at round $r = 1$ may be interrupted $k-1$ times due to the arrival of a new message within at most $r + (2D+1)$ rounds each after the arrival of that message at round r. The flooding of a message that arrives from round $r = 2$ until round $r = (2D+1)+1$ will be interrupted by at most $k-2$ other messages, the flooding of a message that arrives from round $r = (2D+1)+2$ up to $r = 2\cdot(2D+1)+1$ will be interrupted by $k-3$ other messages, and so on. Therefore, in the worst-case, no message terminates flooding by round $(k-1)\cdot(2D+1)+1$. Interestingly, by round $(k-1)\cdot(2D+1)+1$, all k messages are initiated in the system. Therefore, the configuration at round $(k-1)\cdot(2D+1)+1$ resembles configuration of Algorithm 1 with all messages initiated at round $r = 1$. Therefore, Algorithm 1 applies starting from round $(k-1)\cdot(2D+1)+1$ onwards and all k messages terminate flooding in next $k\cdot(2D+1)$ rounds. This gives total $k-1\cdot(2D+1)+k\cdot(2D+1) = (2k-1)\cdot(2D+1)$ rounds. All other cases are subsumed by the above worse-case. We summarize the results in the following theorem.

Theorem 3. *Given any initial configuration of a set $\Theta = \{\theta^1, \ldots, \theta^k\}$ of k messages initiated on the nodes of an arbitrary graph G in possibly different rounds, Algorithm 1 solves flooding in $(2k-1)\cdot(2D+1)$ rounds, given that nodes know D with messages being oblivious to their global ranks.*

Remark: The above approach can handle the continuous case of newly generated stream of messages, removing our assumption of all k messages be generated within round $(k-1)\cdot(2D+1)+1$ as long as the messages generated later in the stream have higher IDs then the previously generated messages. In other words, message IDs increase with increasing initiation time. Theorem 3 considers worst-case of message IDs decrease with increasing initiation time.

Proof of Theorem 1: For arbitrary (non-bipartite) graphs, we have Theorem 1 combining the results of Theorems 2 and 3. If G is bipartite, Hussak and Trehan [10] proved that a single message starting concurrently from possibly multiple source nodes finishes flooding in D rounds. Therefore, we can modify Algorithm 1 replacing all occurrences of $(2D+1)$ with D, which immediately proofs $(2k-1)\cdot D$ rounds for bipartite graphs. Regarding memory requirement, knowing D requires to store $\lceil \log D \rceil$ bits and knowing wait time requires at most $\lceil \log((2k-1)\cdot(2D+1)) \rceil = O(\log k + \log D)$ bits. Therefore, each node needs to store in total $O(\log(\max\{k, D\}))$ bits, combining the bits for D and wait time. $\square$

IV. Concluding Remarks

In this paper, we have considered dynamic amnesiac flooding of $k > 1$ messages initiated at possibly different rounds from possibly multiple source nodes. Hussak and Trehan [10] showed that the partial-send and ranked full-send variations terminate and the unranked full-send variation does not terminate. We have shown that the unranked full-send variation also terminates, provided that the graph nodes know diameter D. Our algorithm uses a non-trivial distributed technique of computing global rank for each message based on its local rank, so that the message with global rank i finishes flooding in time proportional to $i\cdot(2D+1)$ in arbitrary graphs ($i \cdot D$ in bipartite graphs). For future work, it will be interesting to improve termination time to $O(D+k)$ rounds.

References

[1] M. Ahmadi, F. Kuhn, S. Kutten, A. R. Molla, and G. Pandurangan. The communication cost of information spreading in dynamic networks. In *ICDCS*, pages 368–378, 2019.

[2] H. Attiya and J. L. Welch. *Distributed computing - fundamentals, simulations, and advanced topics (2. ed.).* Wiley, 2004.

[3] M. Borokhovich, C. Avin, and Z. Lotker. Tight bounds for algebraic gossip on graphs. In *IEEE ISIT*, pages 1758–1762, 2010.

[4] J. Chen and G. Pandurangan. Almost-optimal gossip-based aggregate computation. *SIAM J. Comput.*, 41(3):455–483, 2012.

[5] B. Doerr, M. Fouz, and T. Friedrich. Social networks spread rumors in sublogarithmic time. In *STOC*, page 21–30, 2011.

[6] R. Elsässer and T. Sauerwald. The power of memory in randomized broadcasting. In *SODA*, pages 218–227, 2008.

[7] M. Ghaffari and J. Li. Improved distributed algorithms for exact shortest paths. In *STOC*, pages 431–444, 2018.

[8] S. Holzer and R. Wattenhofer. Optimal distributed all pairs shortest paths and applications. In *PODC*, pages 355–364, 2012.

[9] W. Hussak and A. Trehan. On the termination of flooding. In *STACS*, pages 17:1–17:13, 2020.

[10] W. Hussak and A. Trehan. Terminating cases of flooding. *CoRR*, abs/2009.05776, 2020.

[11] S. Kutten, G. Pandurangan, D. Peleg, P. Robinson, and A. Trehan. On the complexity of universal leader election. *J. ACM*, 62(1):7:1–7:27, 2015.

[12] S. Kutten, G. Pandurangan, D. Peleg, P. Robinson, and A. Trehan. Sublinear bounds for randomized leader election. *Theor. Comput. Sci.*, 561:134–143, 2015.

[13] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[14] A. Rahman, W. Olesinski, and P. Gburzynski. Controlled flooding in wireless ad-hoc networks. In *International Workshop on Wireless Ad-Hoc Networks, 2004.*, pages 73–78, 2004.

[15] A. S. Tanenbaum and D. J. Wetherall. *Computer Networks*. Prentice Hall Press, USA, 5th edition, 2010.

[16] V. Turau. Analysis of amnesiac flooding. *CoRR*, abs/2002.10752, 2020.

[17] V. Turau. Stateless information dissemination algorithms. In *SIROCCO*, pages 183–199, 2020.

[18] V. Turau. Amnesiac flooding: Synchronous stateless information dissemination. In *SOFSEM*, pages 59–73, 2021.

[19] V. Turau. Synchronous concurrent broadcasts for intermittent channels with bounded capacities. In *SIROCCO*, pages 296–312, 2021.