TRA4/16

*The Cost of Unknown Diameter in Dynamic Networks**

Haifeng Yu, Yuda Zhao and Irvan Jahja

April 2016

T e c h n i c a l   R e p o r t

The Cost of Unknown Diameter in Dynamic Networks[*]

Haifeng Yu
Dept. of Computer Science
National Univ. of Singapore
haifeng@comp.nus.edu.sg

Yuda Zhao
Dept. of Computer Science
National Univ. of Singapore
yuda@comp.nus.edu.sg

Irvan Jahja
Dept. of Computer Science
National Univ. of Singapore
irvan@comp.nus.edu.sg

ABSTRACT

For dynamic networks with *unknown diameter*, we prove novel lower bounds on the time complexity of a range of basic distributed computing problems. Together with trivial upper bounds under dynamic networks with *known diameter* for these problems, our lower bounds show that the complexities of all these problems are *sensitive* to whether the diameter is known to the protocol beforehand: Not knowing the diameter increases the time complexities by a large poly(N) factor as compared to when the diameter is known, resulting in an exponential gap. Our lower bounds are obtained via communication complexity arguments and by reducing from the two-party DISJOINTNESSCP problem. We further prove that sometimes this large poly(N) cost can be completely avoided if the protocol is given a good estimate on N. In other words, having such an estimate makes some problems no longer sensitive to unknown diameter.

Keywords

unknown network diameter, dynamic networks, communication complexity, lower bounds

1. INTRODUCTION

Background and motivation. It is well-known that smaller network diameter often implies smaller time complexity for distributed computing problems. If the diameter D is known beforehand to the protocol (e.g., specified as an input parameter to the protocol), then the protocol needs to guarantee correctness *only* for the given D, and does not need to provide any guarantee for other D values. This allows the protocol to incur as small a complexity as possible for the given D. If D is not known beforehand, in typical static networks, D can still be efficiently estimated by building a spanning tree in just $O(D)$ rounds. This estimate can then be plugged into protocols requiring the knowledge of D. Hence, the complexities of problems in static networks are usually not *sensitive* to unknown diameter, or more rigorously, not *sensitive* to whether the diameter is known beforehand.

In recent years, there is growing interest in *dynamic networks* where the network topology may change over time [17]. Similar to static networks, a dynamic network's *diameter* [17] is the smallest D such that any node can causally affect any other node in the network within D rounds (see Section 2 for the formal definition). A dynamic network's diameter depends on the future behavior of the network, and hence is usually unknown to the protocol. Within such a context, we focus on the following question:

In dynamic networks, do any problems (or which problems) have complexities sensitive to unknown diameter?

The answer to this question can have broad implications: If many basic problems are sensitive, it means that different from static networks, unknown diameter in dynamic networks incurs a fundamental cost. If most problems are not sensitive, then there could be great opportunities in improving many existing protocols for dynamic networks (e.g., [5, 7, 11, 12, 14]): Some of these protocol need the diameter D to be specified as an input parameter. When D is not known beforehand, one is forced to pessimistically set $D = N$ to ensure correctness (N being the total number of nodes). Other protocols do not need such input, but internally implicitly use N for the diameter. In summary, these protocols have not yet explored potentially reducing the complexity when D is unknown, but turns out to be much smaller than N.

To our knowledge, so far the only problem that has been proved to be sensitive to unknown diameter in dynamic networks is the *simultaneous consensus* problem. Specifically, Kuhn et al. [15] investigate a dynamic network model that ignores congestion. Using a knowledge-based proof, they show that the time complexity of simultaneous consensus can increase by a large poly(N) factor if the diameter is not known beforehand, as compared to when the diameter is known. (See later for more discussion on related work.)

Overview of our results. This paper is the first to approach the previous question without ignoring congestion. Doing so perhaps is more realistic since real systems do not have unlimited bandwidth. We adopt the standard $\mathcal{CONGEST}$ model [19], with $O(\log N)$ message sizes. Via communication complexity arguments, we prove several novel lower bounds for *confirmed flooding*, *consensus*, and *leader election* (all defined later) in dynamic networks with un-

[*]The first two authors of this paper are alphabetically ordered.

known diameter. Our results also carry over to the Hear-from-N-nodes problem [16], which in turn, reduces to computing globally-sensitive functions [16] such as Max (see Appendix C).

Our lower bounds show that all these basic problems are sensitive to unknown diameter: Not knowing the diameter increases the time complexity by a poly(N) factor, as compared to when the diameter is known. Such extra complexity is due to the protocol's need to be correct under all possible diameters, even though any given execution only experiences a certain (and potentially small) diameter. Without knowing the actual diameter beforehand, the protocol is forced to be conservative. On the other hand, interestingly, we further prove that sometimes this large cost can be completely avoided if the protocol is given a good estimate on N. In other words, having such an estimate makes some problems no longer sensitive to unknown diameter.

Our results differ from Kuhn et al. [13]'s at a fundamental level: We show that when congestion is taken into account, many more basic problems (besides simultaneous consensus[1]) are sensitive to unknown diameter. Furthermore, considering congestion is *necessary* to reveal the sensitivity of these problems — in their model ignoring congestion, Kuhn et al. actually have implicitly shown that *none* of these problems is sensitive.

Our setting and our problems. In our model, nodes proceed in synchronous rounds. In each round, each node may choose to either send $O(\log N)$ bits or receive, as determined by the randomized protocol executed by the node. A message sent is received by all the sender's neighbors who are receiving in that round. The topology in each round is determined by an adversary and can be an arbitrary connected graph. (See full details of our model in Section 2.) Such a dynamic network model is similar to the evolving-graph model [2] and the T-interval model [14]. All our results and proofs also extend to the dual graph model [9, 13] without any modification.

In *confirmed flooding* (CFlood), a certain node V needs to propagate a token of $O(\log N)$ bits to all nodes. V further, intuitively, wants to confirm that all nodes have received the token.[2] More rigorously, a CFlood protocol *terminates* when V outputs a special symbol, and the output is correct if by the time that V outputs, the token has been received by all nodes. In *consensus* (Consensus), each node has an initial binary value, and they aim to achieve a consensus. The standard requirements of termination, agreement, and validity apply. The protocol *terminates* when every node decides. *Leader election* (LeaderElect) aims to elect a leader and the protocol *terminates* when every node outputs the leader's id. The *time complexity* of a protocol captures the number of rounds needed for the protocol to terminate. To discuss the time complexity under different topologies with different diameters in a consistent way, we will often describe the time complexity under a given dynamic network in terms of *flooding rounds*, with each flooding round having D rounds. We say that a protocol has a time complexity of $s(N)$ flooding rounds if it terminates within s flooding rounds on all dynamic networks of size no more than N.

[1]Kuhn et al.'s results on simultaneous consensus trivially carry over to the $\mathcal{CONGEST}$ model as well.

[2]V may potentially confirm without explicit acknowledgements, such as by counting the number of rounds.

With known diameter, the above three problems can all be solved in $O(\log N)$ flooding rounds. With known diameter, $O(\log N)$ protocols also exist for Hear-from-N-nodes, Max, and estimating N. See Appendix C for all these trivial upper bounds. Under unknown diameter, it was not clear whether $O(\log N)$ flooding rounds is still sufficient, and there have been no prior non-trivial lower bounds.

Our main results. Under unknown diameter, we prove the first non-trivial lower bounds of $\Omega(\sqrt[4]{N/\log N})$ flooding rounds for CFlood, Consensus, and LeaderElect. Such lower bounds are at least *exponentially* larger than the upper bounds when the diameter is known, resulting in an exponential gap. It is worth noting that for obtaining these lower bounds, the topologies we construct for each round all have the same diameter (asymptotically), and the dynamic network's diameter also remains fixed (asymptotically) throughout.[3] In other words, these lower bounds are *not* due to changing diameters (or increasing diameters) in each round. Rather, they are due to the difficulty of confirming whether certain information has reached a sufficient number of nodes.

Our lower bound for CFlood holds even if N is known, while the lower bounds for Consensus and LeaderElect hold even if the protocol knows an estimate N' for N that guarantees $|\frac{N'-N}{N}| \leq \frac{1}{3}$. On the other hand, when there exists a positive constant c such that N' guarantees $|\frac{N'-N}{N}| \leq \frac{1}{3} - c$, we further propose novel Consensus and LeaderElect protocols that do not require prior knowledge of D and yet need only $O(\log N)$ flooding rounds.

Our upper bound protocols suggest that interestingly, the cost of unknown diameter sometimes can be avoided if a good estimate for N is known. The upper bounds also imply that obtaining an N' such that $|\frac{N'-N}{N}| \leq \frac{1}{3} - c$ needs $\Omega(\sqrt[4]{N/\log N})$ flooding rounds, under unknown diameter. When the diameter is known, Appendix C explains how to obtain such an N' in only $O(\log N)$ flooding rounds. Hence, obtaining such an N' itself is also a problem that is sensitive to unknown diameter.

Our main techniques. Our lower bounds are from communication complexity arguments, and more specifically, via reductions from the recently introduced two-party DisjointnessCP [4] problem. Our reductions not only are novel themselves, but also rely on some interesting techniques that have not been exploited in related reductions in other contexts [4, 6, 16, 20, 21]:

- We design three novel types of *subnetworks* as building blocks. We then prove a general *composition* lemma that enables flexible composition of these subnetworks.
- In our reduction, we allow the two parties in the DisjointnessCP problem to *disagree* on the dynamic network that they simulate, which is quite different from reductions in other efforts [4, 6, 16, 20, 21].

Additional related work. For static networks, Kuhn et al. [16] show that the time complexity of the Hear-from-N-nodes problem is sensitive to unknown diameter in *directed*

[3]If needed, one can trivially modify our construction so that the diameter is always fixed, and not just fixed asymptotically.

$\mathcal{G}$	a dynamic network	Γ, Λ, Υ	3 types of subnetworks
N	number of nodes in the dynamic network	A_Γ, B_Γ	special nodes in the types-Γ subnetwork
N'	estimate of N	A_Λ, B_Λ	special nodes in the types-Λ subnetwork
D	diameter of the dynamic network	A_Υ, B_Υ	special nodes in the types-Υ subnetwork
		U, V, W	generic nodes in the network
n	size of the DisjointnessCP problem	$\mathbf{x}$, $\mathbf{y}$	input strings to DisjointnessCP
q	parameter in the DisjointnessCP problem	x_i, y_i	ith character of $\mathbf{x}$, $\mathbf{y}$

Table 1: Key notations.

static networks. Specifically, they prove that Hear-from-N-nodes needs $\tilde{\Omega}(\sqrt{N})$ flooding rounds under unknown diameter, even though it takes only a single flooding round under known diameter. Their proof uses a reduction from the classic Disjointness problem, and critically relies on the directed edges to avoid "leaking" one party's input to the other party. In comparison, our setting is undirected dynamic networks. Neither setting can be reduced to the other. Our reductions face different challenges and are perhaps more complex than the one in [16]: Our reduction needs to i) reduce from a more complex and recently proposed DisjointnessCP problem, ii) allow the two parties to disagree on the dynamic network that they simulate, iii) continuously change the topology (e.g., cascading edge removals), and iv) give up simulating certain nodes as the simulation progresses.

Some researchers have obtained various lower bounds under unknown diameter, for other settings such as *asynchronous* networks with edge failures [1, 10] and *anonymous* static networks without congestion [8]. These proofs all critically rely on the specifics of their settings, and have little relevance to this work. Ghaffari et al. [9] have proved that broadcasting needs $\Omega(N/\log N)$ rounds under some constant-diameter dual graph. This lower bound is, however, not due to the lack of the knowledge of the diameter. Finally, this paper builds upon our own previous work [4] on the communication complexity of computing aggregate functions, and we adopt the DisjointnessCP problem from there. But the actual reduction in this paper is quite different from and more complex than those in [4]. In particular, this paper relies on multiple unique techniques as mentioned earlier.

Roadmap. The next section formalizes our model and definitions. Section 3 gives an overview of our lower bounds under unknown diameter, with details in Section 4 through 6. Section 7 presents upper bounds showing that having a good estimate on N makes some problems no longer sensitive to unknown diameter.

2. MODEL AND DEFINITIONS

Dynamic networks. The dynamic network has N nodes (see Table 1 for the notation summary), each with a unique id of $\Theta(\log N)$ bits. The timing model is synchronous, and all nodes start executing the protocol from round 1 simultaneously. For convenience, we also discuss round 0, where the protocol does nothing. The set of N nodes is always fixed, but the (undirected) edges among the N nodes may change arbitrarily from round to round, subject to the constraint that the topology at any specific point of time must be connected.

For convenience, we say that the topology is determined by an *adversary*. In each round, the nodes first flip their coins (for the randomized protocol). The adversary then determines the topology for the current round, based on the randomized protocol, all the coin flip outcomes so far, and the states of the nodes. (The adversary cannot predict future coin flip outcomes.) Next each node does some local processing, and then either sends or receives, as decided by the randomized protocol.[4]

In a round, a node that chooses to send can send a single message of $O(\log N)$ bits. All neighboring nodes that choose to receive in that round will receive that message. A node may receive multiple messages from multiple neighbors in a round. The topology in each round is unknown to each node, and a node does not know its neighbors unless it receives messages from them.

Following [15], given any round $r \geq 0$ and any two nodes U and V, we say that $(U, r) \to (V, r+1)$ if either (U, V) is an edge in the dynamic network in round $r+1$ or $U = V$. We define "$\rightsquigarrow$" as the transitive closure of "$\to$": Intuitively, $(U, r) \rightsquigarrow (V, r+z)$ means that U's behavior/state in round r may potentially influence V's behavior/state in round $r+z$. The *(dynamic) diameter* of the dynamic network is defined as the minimum D such that for any round $r \geq 0$ and any two nodes U and V, $(U, r) \rightsquigarrow (V, r+D)$.

Time complexity. Since this paper mainly focuses on lower bounds, we consider Monte Carlo protocols with δ error probability (or simply called *δ-error protocols*) for solving various distributed computing problems, and we define time complexity over average coin flips and worst-case input. For lower bounds, the coins will be public, while for upper bounds the coins will be private. This enables all our lower bounds to trivially extend to worst-case coin flips, Las Vegas protocols, and private coin protocols. To discuss the time complexities of these protocols under different topologies with different diameters in a consistent way, we will often describe the time complexity under a given dynamic network in terms of *flooding rounds*, with each flooding rounds being exactly D rounds. Given a dynamic net-

[4]Such an adversary model and send/receive model have been used in prior work on dynamic networks as well (e.g., [3, 9, 13]). There have also been alternative prior models (e.g. [7, 14]) that allow a node to both send and receive in a round (e.g., if the rounds are sufficiently long to accommodate both a send and a receive). All our results continue to hold under such an alternative model, as long as the dynamic network's topology may potentially change in the middle of a round. In fact, when a round contains multiple operations, it is perhaps more realistic not to rule out potential topology changes in the middle of a round. As an analogy, for classic fault-tolerant distributed consensus in synchronous systems, researchers have always considered the possibility of a node failing in the middle of a round, where the node has completed some but not all of the operations it intended to do in that round.

work $\mathcal{G}$, a protocol's *time complexity over* $\mathcal{G}$ is defined as the number of flooding rounds needed for the protocol to terminate, over average coin flips and worst-case input over $\mathcal{G}$. The protocol's *time complexity* is defined as the largest time complexity over all possible $\mathcal{G}$'s with no more than N nodes.

Communication complexity. For positive integer n and positive odd integer $q \geq 3$, DisjointnessCP$_{n,q}$ [4] is a two-party communication complexity problem where Alice and Bob have input strings $\mathbf{x}$ and $\mathbf{y}$, respectively. Here $\mathbf{x}$ and $\mathbf{y}$ each have n characters, with each character being an integer in $[0, q-1]$. Let x_i (y_i) denote the ith character in $\mathbf{x}$ ($\mathbf{y}$), for $1 \leq i \leq n$. Alice and Bob aim to compute DisjointnessCP$(\mathbf{x}, \mathbf{y})$, which is defined to be 0 if there exists any i such that $x_i = y_i = 0$, and 1 otherwise. The inputs $\mathbf{x}$ and $\mathbf{y}$ must satisfy the *cycle promise* [4] in the sense that for any $i \in [1, n]$, we must have either i) $y_i = x_i - 1$, or ii) $y_i = x_i + 1$, or iii) $(x_i, y_i) = (0, 0)$, or iv) $(x_i, y_i) = (q-1, q-1)$. Chen et al. [4] have shown that the cycle promise is not an ad hoc promise — for a wide class of reductions, the cycle promise can in some sense be "derived" from the reduction. Larger q in DisjointnessCP$_{n,q}$ means that some character will have a small number of occurrences in $\mathbf{x}$ and $\mathbf{y}$, and this can be exploited by the upper bound protocol [4] to solve DisjointnessCP$_{n,q}$ more efficiently. The following lower bound is from [4]:

Theorem 1. (From [4].) *A $\frac{1}{5}$-error public coin Monte Carlo protocol for* DisjointnessCP$_{n,q}$ *must incur at least $\Omega(\frac{n}{q^2}) - O(\log n)$ bits of communication between Alice and Bob, over worst-case* $\mathbf{x}$, $\mathbf{y}$, *and worst-case coin flips.*

For this paper, we will need the following corollary (see Appendix A for the simple proof):

Corollary 2. *For any $\frac{1}{6}$-error public coin Monte Carlo protocol for* DisjointnessCP$_{n,q}$, *there exist* $\mathbf{x}_0$ *and* $\mathbf{y}_0$ *such that* DisjointnessCP$_{n,q}(\mathbf{x}_0, \mathbf{y}_0) = 1$ *and the protocol incurs at least $\Omega(\frac{n}{q^2}) - O(\log n)$ bits over* $\mathbf{x}_0$, $\mathbf{y}_0$, *and average coin flips.*

3. OVERVIEW OF OUR LOWER BOUNDS

This section provides an overview of our lower bound proofs for CFlood and Consensus. Consensus can be easily reduced to LeaderElect (see Appendix C), so we do not need to separately prove a lower bound for LeaderElect.

3.1 High-level Structure of Our Proof

Our lower bound proofs for CFlood and Consensus have similar high-level structures, and we use CFlood as an example. Our lower bound for CFlood is based on a reduction from DisjointnessCP. Putting it another way, given a (black-box) oracle protocol for solving CFlood, Alice and Bob will solve DisjointnessCP$(\mathbf{x}, \mathbf{y})$ by simulating the execution of that oracle protocol under a certain dynamic network.

The specifics of this dynamic network depend on the values of both $\mathbf{x}$ and $\mathbf{y}$: The dynamic network is constructed in such a way that it has $O(1)$ diameter if DisjointnessCP$(\mathbf{x}, \mathbf{y}) = 1$, and $\Omega(q)$ diameter if DisjointnessCP$(\mathbf{x}, \mathbf{y}) = 0$. Assume that the CFlood oracle protocol promises to terminate within s flooding rounds. We can then show that i) if DisjointnessCP$(\mathbf{x}, \mathbf{y}) = 1$, then the CFlood protocol under the corresponding network should terminate within $O(s)$ rounds, and ii) if DisjointnessCP$(\mathbf{x}, \mathbf{y}) = 0$, then the CFlood protocol under the corresponding network takes $\Omega(q)$ rounds to terminate. (Note that it is not $\Omega(qs)$ rounds since the protocol may terminate in less than s flooding rounds.)

We will choose a proper value of q to ensure a gap between $O(s)$ and $\Omega(q)$. Alice and Bob will simulate the execution of the CFlood oracle protocol for $O(s)$ rounds. If the CFlood protocol terminates within these $O(s)$ rounds, then Alice and Bob claim DisjointnessCP$(\mathbf{x}, \mathbf{y}) = 1$. Otherwise they claim DisjointnessCP$(\mathbf{x}, \mathbf{y}) = 0$. In such a way, Alice and Bob successfully solves DisjointnessCP after simulating the CFlood oracle protocol for $O(s)$ rounds.

Alice and Bob will need to incur communication during this simulation, as following. There are two special nodes A_Γ and A_Λ in the dynamic network. During each round of the simulation, to enable the simulation to later continue onto the next round, Alice needs to forward to Bob all messages sent by A_Γ and A_Λ in the CFlood oracle protocol in that round. Under the $\mathcal{CONGEST}$ model, A_Γ and A_Λ will altogether send at most $O(\log N)$ bits in one round. Hence Alice will send at most $O(\log N)$ bits to Bob in one round of the simulation, and $O(s \log N)$ bits throughout the simulation. Similarly, there are two additional special nodes B_Γ and B_Λ in the dynamic network, and Bob will forward to Alice all messages sent by B_Γ and B_Λ. Altogether, the number of bits exchanged between Alice and Bob during the simulation is $O(s \log N)$ bits. (This is where the time complexity upper bound of s gets connected to the communication complexity upper bound of $O(s \log N)$.)

Finally, by the communication complexity lower bound on DisjointnessCP, Alice and Bob needs to exchange at least $\Omega(\frac{n}{q^2}) - O(\log n)$ bits to solve DisjointnessCP. This gives us the equation $O(s \log N) = \Omega(\frac{n}{q^2}) - O(\log n)$. Solving this equation gives a lower bound on s.

3.2 Adversaries and Subnetworks

The 3 adversaries. As explained above, the specifics of the dynamic network used in the reduction depends on the value of both $\mathbf{x}$ and $\mathbf{y}$. More precisely, we say that the dynamic network's topology in each round is determined by a *reference adversary*, whose behavior is a function of $\mathbf{x}$ and $\mathbf{y}$. Since Alice does not see $\mathbf{y}$, Alice does not know the reference adversary. Hence the crux in the reduction is to enable Alice to still properly simulate without seeing $\mathbf{y}$. Putting it another way, we want the dynamic network to be "indistinguishable" from Alice's perspective, as long as $\mathbf{x}$ remains fixed and regardless of $\mathbf{y}$. Such discussions symmetrically apply to Bob as well.

To this end, we will exploit the properties of the cycle promise in the DisjointnessCP problem. Furthermore, our reduction will let Alice simulate her own adversary based on $\mathbf{x}$, and Bob simulate his own adversary based on $\mathbf{y}$. We will allow the 3 adversaries (i.e., the reference adversary, Alice's simulated adversary, and Bob's simulated adversary) to be pairwise slightly different. We will nevertheless ensure that the entire simulation is still "meaningful".

The subnetworks. The dynamic networks we use to prove our lower bounds are obtained by composing various novel types of subnetworks (type-Γ, type-Λ, and type-Υ). Each type of subnetwork is itself a dynamic network uniquely determined by the given $\mathbf{x}$ and $\mathbf{y}$. We will also describe the 3 adversaries for each subnetwork. Unless otherwise stated, a subnetwork has $\Theta(nq)$ nodes, and by itself does not need to be connected in each round. Section 4 and Section 5 will present details of these subnetworks. The following provides an overview:

- Type-Γ subnetwork: If DisjointnessCP$(\mathbf{x},\mathbf{y}) = 0$, then there exist $\Omega(q)$ nodes in the type-Γ subnetwork that are disconnected from the rest of the type-Γ subnetwork, starting from the beginning of round 1. These $\Omega(q)$ nodes are arranged into a line and will connect to some other subnetwork. If DisjointnessCP$(\mathbf{x},\mathbf{y}) = 1$, then the type-$\Gamma$ subnetwork is always connected with $O(1)$ diameter.
- Type-Λ subnetwork: If DisjointnessCP$(\mathbf{x},\mathbf{y}) = 0$, then the type-$\Lambda$ subnetwork will contain at least one node as a *mounting point.* It takes $\Omega(q)$ rounds for a mounting point to causally affect all other nodes in the subnetwork. If DisjointnessCP $(\mathbf{x},\mathbf{y}) = 1$, then there is no mounting point and the diameter of the type-Λ subnetwork is $O(1)$.
- Type-Υ subnetwork: If DisjointnessCP$(\mathbf{x},\mathbf{y}) = 0$, the type-$\Upsilon$ subnetwork is the same as the type-Λ subnetwork. If DisjointnessCP $(\mathbf{x},\mathbf{y}) = 1$, the type-$\Upsilon$ subnetwork is empty and has no nodes.

3.3 Composing Subnetworks to Obtain Lower Bound for CFlood and Consensus

To prove the lower bound on CFlood (details in Section 6), we will compose the type-Γ subnetwork with the type-Λ subnetwork together:

- When DisjointnessCP$(\mathbf{x},\mathbf{y}) = 1$, our composition will connect the type-Γ subnetwork with the type-Λ subnetwork together using one edge. Since each of these subnetworks is itself connected and has a diameter of $O(1)$, doing so will result in a dynamic network with $O(1)$ diameter.
- When DisjointnessCP$(\mathbf{x},\mathbf{y}) = 0$, there will be $\Omega(q)$ nodes disconnected from the rest of the type-Γ subnetwork. Our composition will arrange them into a line and then connect them to a mounting point in the type-Λ subnetwork. (We still connect the rest of the type-Γ subnetwork with the type-Λ subnetwork together using one edge as earlier.) This leads to a dynamic network over which the CFlood protocol needs to take $\Omega(q)$ rounds to terminate.

To prove the lower bound on Consensus, one way is to reduce from CFlood. Specifically, Kuhn et al. describe a reduction from Hear-from-N-nodes to Consensus [15], while CFlood reduces to Hear-from-N-nodes (see Appendix C). But Kuhn et al.'s reduction does not directly capture our model — for example, we consider Monte Carlo protocols. While we could adapt that reduction, since we already have the composition lemma, we will conveniently prove a lower bound on Consensus by composing the type-Λ subnetwork with the type-Υ subnetwork (details in Section 6):

- When DisjointnessCP$(\mathbf{x},\mathbf{y}) = 1$, the type-$\Upsilon$ subnetwork is empty. The dynamic network resulted from our composition will simply be the type-Λ subnetwork with $O(1)$ diameter.
- When DisjointnessCP$(\mathbf{x},\mathbf{y}) = 0$, both the type-$\Lambda$ subnetwork and the type-Υ subnetwork have a mounting point. Our composition connects some arbitrary mounting point in the type-Λ subnetwork with some arbitrary mounting point in the type-Υ subnetwork. Recall that in the Consensus problem, each node has a binary initial value. We set the initial values so that all nodes in the type-Λ subnetwork have the same initial values, while all nodes in the type-Υ subnetwork have the opposite initial values. We will prove that over such a dynamic network, the Consensus protocol takes $\Omega(q)$ rounds to terminate on all nodes.

Finally, note that the number of nodes in the type-Υ subnetwork is not fixed, and depends on $\mathbf{x}$ and $\mathbf{y}$. Hence Alice and Bob, without knowing the other party's input, cannot determine the number of nodes in the type-Υ subnetwork. Thus the Consensus lower bound here does not hold when N is known. But Alice and Bob can nevertheless produce an N' with limited accuracy (i.e., $|\frac{N'-N}{N}| \le \frac{1}{3}$) to feed into the oracle protocol if needed.

4. TYPE-Γ SUBNETWORK

The reference adversary. Given $\mathbf{x}$ and $\mathbf{y}$, Figure 1 illustrates the type-Γ subnetwork. In round 0, it has n groups of vertical chains, where each group has $\frac{q-1}{2}$ vertical chains. Each chain has three nodes and two edges. We call the edge adjacent to the top node (bottom node) as the *top edge* (*bottom edge*). We connect the top node on every chain to a special node A_Γ, and the bottom node on every chain to a special node B_Γ. For $1 \le i \le n$, all the $\frac{q-1}{2}$ top nodes (bottom nodes) in the ith group are labeled x_i (y_i). We use $|_y^x$ to denote a chain whose top node is labeled x and whose bottom node is labeled y.

Let t be any non-negative integer. The reference adversary for the type-Γ subnetwork manipulates certain chains according to the following rules:

1. For every chain in the form of $|_{2t-1}^{2t}$, the adversary removes the top edge at the beginning of round $t+1$.
2. For every chain in the form of $|_{2t}^{2t-1}$, the adversary removes the bottom edge at the beginning of round $t+1$.
3. For every chain in the form of $|_{2t+1}^{2t}$, the adversary removes the top edge at the beginning of either round $t+2$ (if the middle node on the chain is receiving in round $t+1$) or round $t+1$ (otherwise).
4. For every chain in the form of $|_{2t}^{2t+1}$, the adversary removes the bottom edge at the beginning of either round $t+2$ (if the middle node on the chain is receiving in round $t+1$) or round $t+1$ (otherwise).
5. For all chains in the form of $|_0^0$, the adversary removes both the top edges and the bottoms edges at the beginning of round 1, hence disconnecting all the middle nodes on these chains. The adversary connects all such

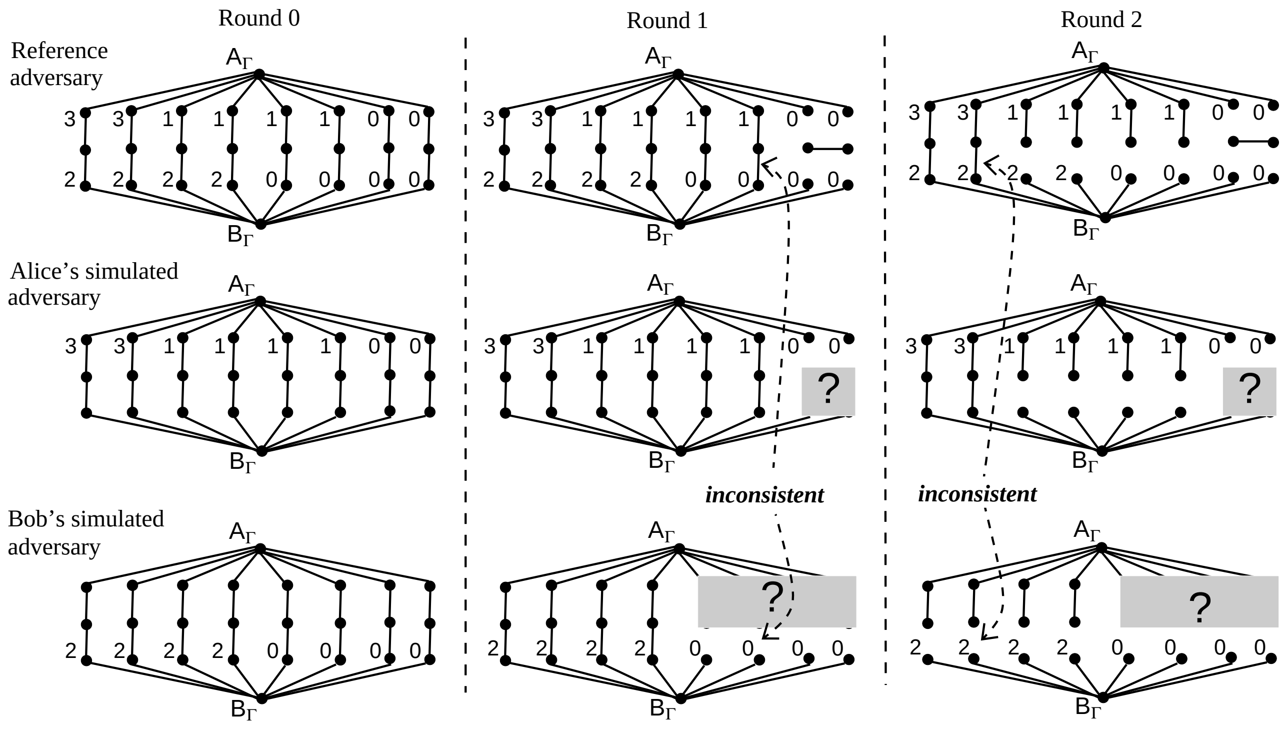

Figure 1: The three adversaries of the type-Γ subnetwork, for $n = 4$, $q = 5$, $\mathbf{x} = 3110$, and $\mathbf{y} = 2200$. The numbers beside the nodes are labels. This example assumes that the middle nodes on all the chains are receiving.

middle nodes into a line. Obviously, we will have at least $\frac{q-1}{2}$ chains in the form of $|_0^0$ if $\text{DISJOINTNESSCP}(\mathbf{x}, \mathbf{y}) = 0$. In such a case, we will have a line of $\Omega(q)$ nodes which can be connected to some other subnetwork to boost the diameter.

Finally, the adversary does not manipulate $|_{q-1}^{q-1}$ chains, which are the only remaining kind of chains.

Adversaries simulated by Alice and Bob. Alice will not be able to simulate this reference adversary, because Alice does not know $\mathbf{y}$ and hence does not know the labels of the bottom nodes. Overcoming this issue is a key challenge in the reduction. Let "*" be a wildcard label and let t be any non-negative integer. Based on only $\mathbf{x}$, Alice simulates the following adversary (called *Alice's simulated adversary*) instead:

- For every chain in the form of $|_*^{2t}$, the adversary removes the top edge at the beginning of round $t + 1$.
- For every chain in the form of $|_*^{2t+1}$, the adversary removes the bottom edge at the beginning of round $t + 2$.

Our simulation will only last for $\frac{q-1}{2}$ rounds, and hence Alice's adversary will not have removed any edges from $|_*^{q-1}$ chains and $|_*^{q-2}$ chains by the end of the simulation.

Similarly, Bob simulates the following adversary (called *Bob's simulated adversary*):

- For every chain in the form of $|_{2t}^*$, the adversary removes the bottom edge at the beginning of round $t+1$.
- For every chain in the form of $|_{2t+1}^*$, the adversary removes the top edge at the beginning of round $t + 2$.

Note that Alice's simulated adversary and Bob's simulated adversary diverge. For example, for a $|_{2t+1}^{2t}$ chain, Alice's simulated adversary removes the top edge in round $t + 1$, while Bob's simulated adversary does so in round $t + 2$.

Communication between Alice and Bob. To allow the simulation to continue properly, during each round of the simulation, Alice will always forward to Bob the message sent by the node A_Γ (if any). We will later show that Alice can always generate those messages. Similarly, Bob will forward to Alice all messages sent by B_Γ.

A concrete example. Part of our proof will need to show that despite the three adversaries (i.e., the reference adversary, Alice's simulated adversary, and Bob's simulated adversary) can be pairwise different and inconsistent, the behavior of the oracle protocol under the three different adversaries will nevertheless be "consistent enough" to allow simulation.

To get some intuition, Figure 1 illustrates the three adversaries. In round 0, the topology is the same under all three adversaries. Next in round 1, under the reference adversary, all the edges in the two $|_0^0$ chains are removed. Under Alice's simulated adversary, the top edges of the two $|_*^0$ chains are removed. Alice cannot infer (as indicated by "?" in the figure) whether the bottom edges are removed under the reference adversary. But since the top edges on these two chains have already been removed, the behavior of the nodes in the "?" region cannot causally affect other nodes, without passing through B_Γ first. Since Bob will forward to Alice all the messages sent by B_Γ, Alice can afford to simply give up simulating those nodes in the "?" region. Similar arguments apply to Bob.

Next consider the $|_0^1$ chains in round 1 and $|_2^3$ chains in round 2. In Figure 1, Bob's simulated adversary removes the bottom edge of every $|_0^1$ chain at the beginning of round 1, while the reference adversary will not remove those edges until round 2. Here this edge removal will causally affect B_Γ, preventing Bob from properly simulating B_Γ. We will leverage the following observation to address this issue. Let the three nodes on a $|_0^1$ chain be U, V, and W, from the top to the bottom. If V is receiving in round 1 (as in Figure 1), then W cannot tell whether the bottom edge is removed in round 1 or round 2. Hence even though Bob simulates edge removal in round 1, so far as the protocol is concerned, it is equivalent to removing the bottom edge in round 2. The case for V sending in round 1 is similar.

Correctness arguments. We next formalize the correctness arguments. In any round, a node is defined to be either *spoiled* or *non-spoiled* with respect to Alice. Intuitively, the behavior of a non-spoiled node for Alice depends only on Alice's input $\mathbf{x}$ and messages sent by B_Γ. We will eventually aim to prove that Alice can properly simulate the execution of the oracle protocol on a node in a round r ($r \le \frac{q-1}{2}$), if it is non-spoiled for Alice in that round. Let t be any non-negative integer. Consider any given chain and let the three nodes on the chain be U, V, and W, from the top to the bottom:

- If the chain is in the form of $|_*^{2t}$, then V and W become spoiled since the beginning of round $t+1$.
- If the chain is in the form of $|_*^{2t+1}$, then W becomes spoiled since the beginning of round $t+1$.

We define B_Γ to be spoiled for Alice from the beginning of round 1. A node is *non-spoiled* to Alice unless it is spoiled for Alice. In particular, A_Γ is always non-spoiled for Alice.

We similarly define these concepts for Bob:

- If the chain is in the form of $|_{2t}^*$, then V and U become spoiled since the beginning of round $t+1$.
- If the chain is in the form of $|_{2t+1}^*$, then U becomes spoiled since the beginning of round $t+1$.

We define A_Γ to be spoiled for Bob from the beginning of round 1. A node is *non-spoiled* to Bob unless it is spoiled for Bob. B_Γ is always non-spoiled for Bob.

The following lemma (with proof in Appendix A) claims that for any non-spoiled receiving node Z in a round, Alice is able to determine which nodes can potentially send messages to Z under the reference adversary, by simulating her own adversary based on only $\mathbf{x}$. Because those nodes are either non-spoiled or B_Γ, this lemma will allow us to later prove in Section 6, via an induction, that Alice can simulate all her non-spoiled nodes.

LEMMA 3. *Consider any round r where $1 \le r \le \frac{q-1}{2}$ in the type-Γ subnetwork. For any non-spoiled node Z for Alice (Bob) in round r that is receiving in round r, let $\mathcal{S}$ be the set of Z's neighbors under the reference adversary in round r, and $\mathcal{S}'$ be the set of Z's neighbors under the adversary simulated by Alice (Bob) in round r. Then i) all nodes in $(\mathcal{S} \setminus \mathcal{S}') \cup (\mathcal{S}' \setminus \mathcal{S})$ are receiving in round r, and ii) a node in $\mathcal{S}'$ is either B_Γ (A_Γ) or a non-spoiled node for Alice (Bob) in round $r-1$.*

5. TYPE-Λ AND TYPE-Υ SUBNETWORKS

We describe type-Λ subnetwork first and type-Υ subnetwork next.

Mounting points. Recall from Section 3 that the type-Λ subnetwork should contain at least one node as a *mounting point* when DISJOINTNESSCP$(\mathbf{x}, \mathbf{y}) = 0$. This allows us to later attach nodes to a mounting point when needed. Whether a mounting point exists depends on both $\mathbf{x}$ and $\mathbf{y}$. Hence if a mounting point exists, without seeing both $\mathbf{x}$ and $\mathbf{y}$, neither Alice nor Bob can properly simulate it. Intuitively, a mounting point is always spoiled for both Alice and Bob. We need to prevent a mounting point from quickly affecting all other nodes, since any nodes causally affected by a mounting point intuitively become spoiled as well. We will carefully remove edges to achieve this. On the other hand, the type-Λ subnetwork needs to have $O(1)$ diameter when DISJOINTNESSCP$(\mathbf{x}, \mathbf{y}) = 1$. Thus the crux here is that such edge removals should not increase the diameter of the type-Λ subnetwork when DISJOINTNESSCP$(\mathbf{x}, \mathbf{y}) = 1$. While we will still use the trick of having three different adversaries, that trick by itself no longer suffices here. We will need an additional technique of *cascading edge removals* over novel *centipede structures*.

The 3 adversaries and the spoiled nodes. Given $\mathbf{x}$ and $\mathbf{y}$, Figure 2 and 3 illustrate the type-Λ subnetwork. In round 0, the topology has n *centipede structures*, one for each index $i \in [1, n]$. Each centipede structure has $\frac{q+1}{2}$ vertical chains, and each chain has three nodes. The middle nodes on all chains in a centipede structure are connected and form a horizontal line. The top (bottom) nodes on all chains of all centipede structures connect to a special node A_Λ (B_Λ). Consider the jth chain in the ith centipede structure for $1 \le j \le \frac{q+1}{2}$ and $1 \le i \le n$, and let the three nodes on the chain to be U, V, and W, from the top to the bottom. We label U and W as $\min(x_i + 2j - 2, q-1)$ and $\min(y_i + 2j - 2, q-1)$, respectively. The middle nodes of all $|_0^0$ chains are defined as mounting points. The reference adversary for the type-Λ subnetwork follows the same rules as the reference adversary for the type-Γ subnetwork in Section 4, except that the 5th rule is replaced by:

5. Let t be any integer in $[0, \frac{q-3}{2}]$. For all chains in the form of $|_{2t}^{2t}$, the adversary removes both the top edges and the bottom edges at the beginning of round $t+1$.

Note that cascading edge removals are implicit here — edge removals following all the rules will be cascading.

Finally, the adversary simulated by Alice (Bob) in the type-Λ subnetwork follows the exact same rules (but now based on the labels in the type-Λ subnetwork) as earlier in the type-Γ subnetwork in Section 4. Spoiled/non-spoiled nodes are also defined according to exactly the same rules as in Section 4 (after replacing A_Γ with A_Λ, and replacing B_Γ with B_Λ). In the simulation, Alice will always forward to Bob all messages sent by A_Λ. (We will later show that Alice can always generate those messages.) Similarly, Bob will forward all messages sent by B_Λ.

A concrete example. We aim to highlight here the role of cascading edge removals, while focusing on the reference adversary. Figure 2 illustrates the ith centipede structure when $x_i = y_i = 0$. This structure has a mounting point, which is the middle node of the $|_0^0$ chain. A mounting point

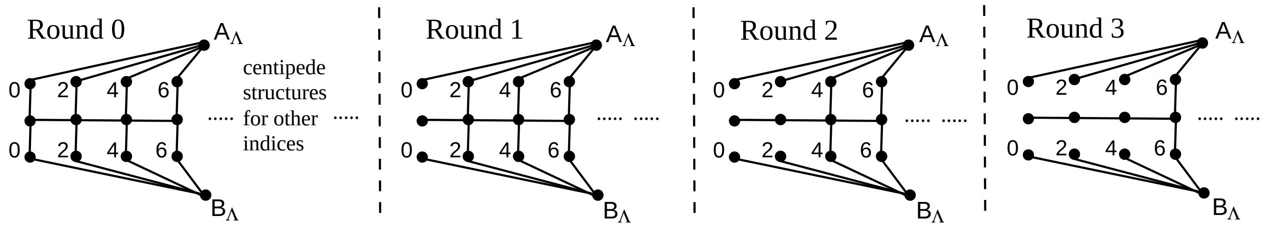

Figure 2: The ith centipede structure in the type-Λ subnetwork under the reference adversary, for $x_i = y_i = 0$ and $q = 7$.

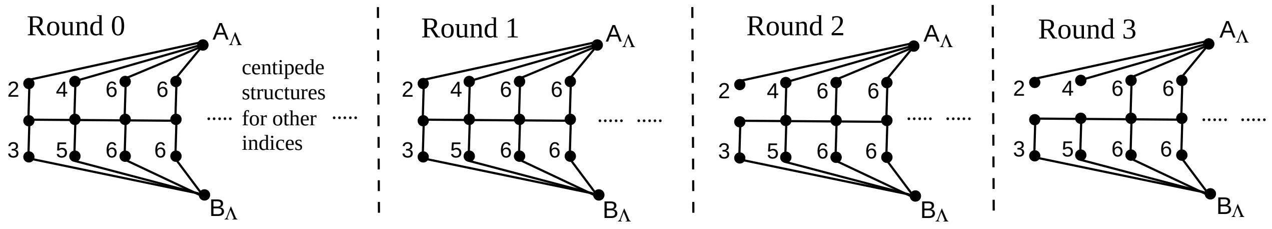

Figure 3: The ith centipede structure in the type-Λ subnetwork under the reference adversary, for $x_i = 2$, $y_i = 3$, and $q = 7$, assuming all middle nodes on all chains are sending.

is spoiled from the beginning of round 1 for both Alice and Bob. To prevent it from causally affecting A_Λ and B_Λ, we remove the two edges on the $|_0^0$ chain at the beginning of round 1, remove the edges on the $|_2^2$ chain at the beginning of round 2, and so on.

One may wonder why we cannot simply remove the edges on all these chains at the same time. To understand, imagine that the two edges on the $|_4^4$ chain in Figure 2 are removed in round 1 instead of in round 3. Once we do this in the reference adversary, Alice (Bob) will no longer be able to simulate the middle node V on this chain, since based on $\mathbf{x}$ ($\mathbf{y}$), Alice (Bob) cannot tell whether both edges on the chain has been removed. Intuitively, V becomes spoiled. V may now causally affect A_Λ and B_Λ, via the $|_6^6$ chain. This can happen rather quickly since V is just next to the $|_6^6$ chain.

Finally, we will also need to remove edges in the ith centipede structure when $x_i + y_i > 0$, as illustrated in Figure 3 where $x_i = 2$ and $y_i = 3$. While not immediately obvious, cascading edge removals play a critical role here as well: The middle node V on the $|_3^2$ chain becomes spoiled for Alice at the beginning of round 2, and cascading edge removals prevent V from causally affecting A_Λ via the $|_5^4$ chain.

Correctness arguments. The following lemma (whose proof is in Appendix A) is the same as Lemma 3 except that it is now for the type-Λ subnetwork:

LEMMA 4. *Consider any round r where $1 \le r \le \frac{q-1}{2}$ in the type-Λ subnetwork. For any non-spoiled node Z for Alice (Bob) in round r that is receiving in round r, let $\mathcal{S}$ be the set of Z's neighbors under the reference adversary in round r, and $\mathcal{S}'$ be the set of Z's neighbors under the adversary simulated by Alice (Bob) in round r. Then i) all nodes in $(\mathcal{S} \setminus \mathcal{S}') \cup (\mathcal{S}' \setminus \mathcal{S})$ are receiving in round r, and ii) a node in $\mathcal{S}'$ is either B_Λ (A_Λ) or a non-spoiled node for Alice (Bob) in round $r-1$.*

Type-Υ subnetwork. So far we have only described the type-Λ subnetwork, and we now move on to the type-Υ subnetwork.. Under the reference adversary, the type-Υ subnetwork is the same as the type-Λ subnetwork when DISJOINTNESSCP$(\mathbf{x}, \mathbf{y}) = 0$. To avoid confusion, we rename A_Λ (B_Λ) to be A_Υ (B_Υ) in the type-Υ subnetwork. When DISJOINTNESSCP$(\mathbf{x}, \mathbf{y}) = 1$, the type-$\Upsilon$ subnetwork is an empty network with no nodes. Under Alice's and Bob's simulated adversary, the type-Υ subnetwork is always empty (even when DISJOINTNESSCP$(\mathbf{x}, \mathbf{y}) = 0$). We define all nodes (if any) in the type-Υ subnetwork as always spoiled for both Alice and Bob. In the simulation, Alice (Bob) does not need to forward messages sent by A_Υ (B_Υ) to the other party.

6. LOWER BOUNDS FOR CFLOOD AND CONSENSUS

The composition lemma. We first present a lemma to enable the composition of multiple subnetworks. While this lemma can be extremely general, to simplify discussion, we only present a restricted form. Given a dynamic network $\mathcal{G}$ and a round, we use $\mathcal{G}^N$ and $\mathcal{G}^E$ to denote the set of vertices and edges in that round, respectively. A dynamic network $\mathcal{G}$ is the *composition network* of two dynamic networks $\mathcal{G}_1$ and $\mathcal{G}_2$ via *bridging edge set* $\mathcal{E}$ if for every round, $\mathcal{G}^N = \mathcal{G}_1^N \cup \mathcal{G}_2^N$ and $\mathcal{G}^E = \mathcal{G}_1^E \cup \mathcal{G}_2^E \cup \mathcal{E}$. All edges in the bridging edge set are required to span $\mathcal{G}_1$ and $\mathcal{G}_2$. Note that $\mathcal{E}$ does not change from round to round. A mapping from DISJOINTNESSCP instances to dynamic networks is called a *composition mapping* of type-φ_1 and type-φ_2 subnetworks ($\varphi_1, \varphi_2 \in \{\Gamma, \Lambda, \Upsilon\}$), if the mapped dynamic network $\mathcal{G}$ is always a composition network of the corresponding type-φ_1 subnetwork and type-φ_2 subnetwork for the given DISJOINTNESSCP instance. Note that the bridging edge set is allowed to differ under different DISJOINTNESSCP instances. A bridging edge in a dynamic network appeared in a composition mapping is *sensitive for Alice (Bob)* if at least one of its end points is a non-spoiled node for Alice (Bob) in that dynamic network in the first round. A composition mapping is called a *simple composition mapping* if i) both end points of every sensitive bridging edge for Alice (Bob) are always non-spoiled for Alice (Bob) up to round $\frac{q-1}{2}$, and ii) every sensitive bridging edge for Alice (Bob) appears in every dynamic network in the composition mapping.

LEMMA 5. *Consider any given simple composition mapping of type-φ_1 and type-φ_2 subnetworks ($\varphi_1, \varphi_2 \in \{\Gamma, \Lambda, \Upsilon\}$), and any given inputs $\mathbf{x}$ and $\mathbf{y}$ to the* DISJOINTNESSCP *problem. Let $\mathcal{G}$ be the dynamic network corresponding to $\mathbf{x}$ and $\mathbf{y}$, under this simple composition mapping. Consider the execution of any given oracle protocol over $\mathcal{G}$, under certain inputs to the nodes and certain public coin flip outcomes. Then for all round r where $0 \le r \le \frac{q-1}{2}$, Alice (Bob) can determine both the incoming and the outgoing messages of a node V in round r of that execution, based on only $\mathbf{x}$ ($\mathbf{y}$), if:*

- *V is non-spoiled for Alice (Bob) in round r.*
- *Alice (Bob) knows the oracle protocol and the public coin flip outcomes.*
- *Alice (Bob) knows the inputs to all her (his) corresponding non-spoiled nodes in the first round.*
- *Bob (Alice) always forwards to the other party all messages sent by B_Γ and B_Λ (A_Γ and A_Λ) in all rounds, as long as those nodes exist and if he (she) can determine those messages.*

PROOF. See Appendix A. $\square$

Lower bounds for CFLOOD **and** CONSENSUS**.** Using this lemma and following the intuition described in Section 3, we can eventually obtain the following two main theorems, whose proofs are in Appendix A.

THEOREM 6. *If the diameter D is unknown to the protocol beforehand, then a $\frac{1}{6}$-error Monte Carlo protocol for* CFLOOD *must have a time complexity of $\Omega(\sqrt[4]{N/\log N})$ flooding rounds. Furthermore, this holds even if the protocol knows N and the nodes' ids are from 1 to N.*

THEOREM 7. *If the diameter D is unknown to the protocol beforehand, then a $\frac{1}{18}$-error Monte Carlo protocol for* CONSENSUS *must have a time complexity of $\Omega(\sqrt[4]{N/\log N})$ flooding rounds. Furthermore, this holds even if the protocol knows an estimate N' for N that guarantees $|\frac{N'-N}{N}| \le \frac{1}{3}$.*

7. UPPER BOUND FOR CONSENSUSAND LEADERELECT— EFFECT OF A GOOD ESTIMATE OF N

Our lower bounds on CONSENSUS and LEADERELECT no longer hold when N' promises to satisfy $|\frac{N'-N}{N}| \le \frac{1}{3} - c$, for any positive constant c. For such N', this section presents a novel LEADERELECT protocol that does not require any prior knowledge of D and yet has a time complexity of only $O(\log N)$ flooding rounds.[5] Since CONSENSUS can be trivially reduced to LEADERELECT (see Appendix C), such an upper bound applies to CONSENSUS as well. The following provides the intuition for the protocol, with the pseudo-code and the proofs deferred to Appendix B. At the high level, the protocol proceeds in phases and keeps a guess D' for D, with D' doubling in each phase.

Majority counting. Our LEADERELECT protocol needs the following *majority counting* subroutine. Imagine that each node holds an input value. The subroutine uses well-known techniques [18] to count the total number of nodes holding a given node's input, and determines whether it is a majority of all the nodes in the system, within $O(D' \log N)$ rounds. There may be many distinct input values in the system, in which case the subroutine counts all these input values in parallel, while still having $O(\log N)$ message sizes. When $D' < D$ or when there are multiple distinct input values, the subroutine may under-count. With high probability, the subroutine does not over-count. If $D' \ge D$ and if there is a unique input value in the system, the subroutine will claim a majority with high probability. In other words, the subroutine is rather conservative in claiming a majority, and has one-sided error (with high probability). When determining the majority, we will need to use the condition of $|\frac{N'-N}{N}| \le \frac{1}{3} - c$.

Locking a majority. Given a D', our LEADERELECT protocol tries to elect the node with the largest id as the leader. Conceptually, it does so by simply flooding the ids of all the nodes for D' rounds, with only the largest id seen so far surviving in the flooding. If a node V finds that its id is the largest id it has seen after D' rounds, it tries to become a leader. (There can be multiple such V's when $D' < D$.) To do so, V will try to lock at least a majority of all the nodes, by flooding V's lock message for D' rounds. Whoever receiving this message will get locked, unless it has already been locked by someone else. Next V uses majority counting to see whether it has locked a majority. If so, V declares itself as the leader, and floods its id in future phases to notify all other nodes of the leader's id. Otherwise V floods an unlock message in future phases to roll back.

Avoid excessive lock roll back. If there are many such V's that fail to lock a majority, there will be many unlock messages to be propagated, which would result in excessive communication overhead and hence time complexity. Our key technique to overcome this is to have a separate stage before V actually acquires locks. Specifically, if V find that its id is the largest id it has seen after the D' rounds of initial flooding, before V acquires locks, V uses majority counting to check whether a majority of nodes have seen V's id. V will only acquire the locks if it finds a majority. This separate stage ensures (with high probability) that in a given phase, there is only at most one node that tries to acquire locks and hence may potentially need to unlock later.

It is worth noting that our protocol invokes majority counting twice — once for counting the nodes seeing V's id, and once for counting the nodes locked by V. These two invocations cannot be replaced by one: The largest id seen by a node may change from phase to phase. But once a node is locked, it will remain locked in all future phases unless it is explicitly unlocked.

Correctness. Intuitively, the correctness of the protocol comes from two facts. First, once a node declares itself as the leader, it must have locked a majority of the nodes (with

[5]Kuhn et al. [15] also notice that their reduction from HEAR-FROM-N-NODES to CONSENSUS no longer works when $|\frac{N'-N}{N}| \le \frac{1}{3} - c$, but their claim does not rule out the possibility of obtaining a good lower bound on CONSENSUS via other means.

high probability). This prevents other nodes from becoming leaders later. Second, once D' reaches D, all unsuccessful lockings done in previous phases will be fully rolled back, and also all nodes will now see the largest id in the system. The node with the largest id will then get the majority (with high probability) in both steps, and become the leader. For space limitations, we leave the pseudo-code and the proof of the following theorem to Appendix B:

THEOREM 8. *Let $c \in (0, \frac{1}{3}]$ be any given constant and N' be any value such that $|\frac{N'-N}{N}| \leq \frac{1}{3} - c$. Consider the* LEADERELECT *problem where c and N' are known to the protocol. Then even if the diameter D is unknown to the protocol beforehand, there exists a $\frac{1}{N}$-error Monte Carlo* LEADERELECT *protocol with a time complexity of $O(\log N)$ flooding rounds.*

8. ACKNOWLEDGMENTS

We thank Kartik Sankaran and the SPAA anonymous reviewers for helpful feedbacks. This work is partly supported by the research grant MOE2014-T2-2-030 from Singapore Ministry of Education Academic Research Fund Tier-2.

9. REFERENCES

[1] Y. Afek and D. Hendler. On the complexity of global computation in the presence of link failures: the general case. *Distributed Computing*, 8(3):115–120, 1995.

[2] C. Avin, M. Koucký, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *ICALP*, July 2008.

[3] K. Censor-Hillel, S. Gilbert, F. Kuhn, N. Lynch, and C. Newport. Structuring unreliable radio networks. In *PODC*, June 2011.

[4] B. Chen, H. Yu, Y. Zhao, and P. B. Gibbons. The cost of fault tolerance in multi-party communication complexity. *Journal of the ACM*, 61(3), May 2014.

[5] A. Cornejo, S. Gilbert, and C. Newport. Aggregation in dynamic networks. In *PODC*, July 2012.

[6] A. Drucker, F. Kuhn, and R. Oshman. The communication complexity of distributed task allocation. In *PODC*, July 2012.

[7] C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In *SODA*, Jan. 2013.

[8] E. Fusco and A. Pelc. Knowledge, level of symmetry, and time of leader election. In *ESA*, Sept. 2012.

[9] M. Ghaffari, N. Lynch, and C. Newport. The cost of radio network broadcast for different models of unreliable links. In *PODC*, July 2013.

[10] O. Goldreich and L. Shrira. On the complexity of computation in the presence of link failures: the case of a ring. *Distributed Computing*, 5(3), 1991.

[11] B. Haeupler and D. Karger. Faster information dissemination in dynamic networks via network coding. In *PODC*, June 2011.

[12] B. Haeupler and F. Kuhn. Lower bounds on information dissemination in dynamic networks. In *DISC*, Oct. 2012.

[13] F. Kuhn, N. Lynch, C. Newport, R. Oshman, and A. Richa. Broadcasting in unreliable radio networks. In *PODC*, July 2010.

[14] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *STOC*, June 2010.

[15] F. Kuhn, Y. Moses, and R. Oshman. Coordinated consensus in dynamic networks. In *PODC*, June 2011.

[16] F. Kuhn and R. Oshman. The complexity of data aggregation in directed networks. In *DISC*, Sept. 2011.

[17] F. Kuhn and R. Oshman. Dynamic networks: models and algorithms. *SIGACT News*, 42(1):82–96, Mar. 2011.

[18] D. Mosk-Aoyama and D. Shah. Fast distributed algorithms for computing separable functions. *IEEE Transactions on Information Theory*, 54(7):2997–3007, 2008.

[19] D. Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, 1987.

[20] A. Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. In *STOC*, 2011.

[21] Y. Zhao, H. Yu, and B. Chen. Near-optimal communication-time tradeoff in fault-tolerant computation of aggregate functions. *Distributed Computing*, 29(1):17–38, Feb 2016.

APPENDIX

A. OMITTED PROOFS

Corollary 2 (Restated). *For any $\frac{1}{6}$-error public coin Monte Carlo protocol for* $\textsc{DisjointnessCP}_{n,q}$*, there exist* $\mathbf{x}_0$ *and* $\mathbf{y}_0$ *such that* $\textsc{DisjointnessCP}_{n,q}(\mathbf{x}_0, \mathbf{y}_0) = 1$ *and the protocol incurs at least* $\Omega(\frac{n}{q^2}) - O(\log n)$ *bits over* $\mathbf{x}_0$*,* $\mathbf{y}_0$*, and average coin flips.*

Proof. Let $\mathcal{P}$ be any given $\frac{1}{6}$-error protocol with the following property: There does not exist $\mathbf{x}_0$, $\mathbf{y}_0$ where $\textsc{DisjointnessCP}_{n,q}(\mathbf{x}_0, \mathbf{y}_0) = 1$, such that $\mathcal{P}$ incurs more than a bits over the average coin flips. We construct a new protocol $\mathcal{Q}$ by invoking $\mathcal{P}$. $\mathcal{Q}$ outputs $\mathcal{P}$'s output if $\mathcal{P}$ terminates within $30a$ bits of communication. Otherwise $\mathcal{Q}$ outputs 0. It is easy to verify that the error probability of $\mathcal{Q}$ is at most $\frac{1}{6} + \frac{1}{30} = \frac{1}{5}$, and $\mathcal{Q}$ incurs at most $30a$ bits over worst-case $\mathbf{x}$, $\mathbf{y}$, and worst-case coin flips. Applying Theorem 1 immediately gives us the corollary. $\square$

Lemma 3 (Restated). *Consider any round r where $1 \le r \le \frac{q-1}{2}$ in the type-Γ subnetwork. For any non-spoiled node Z for Alice (Bob) in round r that is receiving in round r, let $\mathcal{S}$ be the set of Z's neighbors under the reference adversary in round r, and $\mathcal{S}'$ be the set of Z's neighbors under the adversary simulated by Alice (Bob) in round r. Then i) all nodes in $(\mathcal{S} \setminus \mathcal{S}') \cup (\mathcal{S}' \setminus \mathcal{S})$ are receiving in round r, and ii) a node in $\mathcal{S}'$ is either B_Γ (A_Γ) or a non-spoiled node for Alice (Bob) in round $r - 1$.*

Proof. It suffices to prove the lemma for Alice. In the following, the notions of spoiled and non-spoiled nodes are always with respect to Alice. First, B_Γ is always spoiled in all rounds, and A_Γ is always non-spoiled. For A_Γ, we have $\mathcal{S} = \mathcal{S}'$ and they contain all the top nodes on all the chains. None of these top nodes are ever spoiled for Alice. Next we consider the non-spoiled nodes on all the chains. Consider any given chain, and let U, V, and W be the nodes on the chain from the top to the bottom. We will enumerate all 6 kinds of chains. Let t be any non-negative integer.

For a $|^{2t}_{2t+1}$ chain, U is always non-spoiled, and V and W are non-spoiled iff $r \le t$:

- For node U, we exhaustively enumerate all scenarios: i) If $r \le t$, then $\mathcal{S} = \mathcal{S}' = \{A_\Gamma, V\}$. By definition, both A_Γ and V are non-spoiled in round $r-1$. ii) If $r \ge t+2$, then $\mathcal{S} = \mathcal{S}' = \{A_\Gamma\}$. By definition, A_Γ is non-spoiled in round $r - 1$. iii) If $r = t + 1$ and V is sending in round $t+1$, then $\mathcal{S} = \mathcal{S}' = \{A_\Gamma\}$ and A_Γ is non-spoiled in round $r - 1$. iv) If $r = t + 1$ and V is receiving in round $t + 1$, then $\mathcal{S} = \{A_\Gamma, V\}$ and $\mathcal{S}' = \{A_\Gamma\}$. Again, A_Γ is non-spoiled in round $r - 1$.
- For node V and $r \le t$, we have $\mathcal{S} = \mathcal{S}' = \{U, W\}$, and both nodes are non-spoiled in round $r - 1$.
- For node W and $r \le t$, we have $\mathcal{S} = \mathcal{S}' = \{V, B_\Gamma\}$, where V is non-spoiled in round $r - 1$.

For a $|^{2t}_{2t-1}$ chain, U is always non-spoiled, and V and W are non-spoiled iff $r \le t$:

- For node U, we exhaustively enumerate all scenarios: i) If $r \le t$, then $\mathcal{S} = \mathcal{S}' = \{A_\Gamma, V\}$. By definition, both A_Γ and V are non-spoiled in round $r-1$. ii) If $r \ge t+1$, then $\mathcal{S} = \mathcal{S}' = \{A_\Gamma\}$. By definition, A_Γ is non-spoiled in round $r - 1$.
- For node V and $r \le t$, we have $\mathcal{S} = \mathcal{S}' = \{U, W\}$, and both nodes are non-spoiled in round $r - 1$.
- For node W and $r \le t$, we have $\mathcal{S} = \mathcal{S}' = \{V, B_\Gamma\}$, where V is non-spoiled in round $r - 1$.

For a $|^{2t+1}_{2t}$ chain, U and V are always non-spoiled, and W is non-spoiled iff $r \le t$:

- For node U, $\mathcal{S} = \mathcal{S}' = \{A_\Gamma, V\}$. By definition, both A_Γ and V are non-spoiled in round $r - 1$.
- For node V, we exhaustively enumerate all scenarios: i) If $r \le t$, we have $\mathcal{S} = \mathcal{S}' = \{U, W\}$, and both nodes are non-spoiled in round $r - 1$. ii) If $r \ge t + 2$, then $\mathcal{S} = \mathcal{S}' = \{U\}$. By definition, U is non-spoiled in round $r-1$. iii) If $r = t+1$, recall that we only need to consider the case where V is receiving in round $t + 1$. We have $\mathcal{S} = \mathcal{S}' = \{U, W\}$, and both U and W are non-spoiled in round $r - 1$.
- For node W and $r \le t$, we have $\mathcal{S} = \mathcal{S}' = \{V, B_\Gamma\}$, where V is non-spoiled in round $r - 1$.

For a $|^{2t-1}_{2t}$ chain, U and V are always non-spoiled, and W is non-spoiled iff $r \le t - 1$:

- For node U, $\mathcal{S} = \mathcal{S}' = \{A_\Gamma, V\}$. By definition, both A_Γ and V are non-spoiled in round $r - 1$.
- For node V, we enumerate all scenarios: i) If $r \le t$, we have $\mathcal{S} = \mathcal{S}' = \{U, W\}$, and both nodes are non-spoiled in round $r - 1$. ii) If $r \ge t + 1$, then $\mathcal{S} = \mathcal{S}' = \{U\}$. By definition, U is non-spoiled in round $r - 1$.
- For node W and $r \le t - 1$, we have $\mathcal{S} = \mathcal{S}' = \{V, B_\Gamma\}$, where V is non-spoiled in round $r - 1$.

For a $|^{q-1}_{q-1}$ chain, U, V, and W are always non-spoiled for all $r \le \frac{q-1}{2}$:

- For node U , we have $\mathcal{S} = \mathcal{S}' = \{A_\Gamma, V\}$. By definition, both A_Γ and V are non-spoiled in round $r - 1$.
- For node V, we have $\mathcal{S} = \mathcal{S}' = \{U, W\}$. By definition, both U and W are non-spoiled in round $r - 1$.
- For node W, we have $\mathcal{S} = \mathcal{S}' = \{B_\Gamma, W\}$. By definition, W is non-spoiled in round $r - 1$.

Finally, for a $|^{0}_{0}$ chain, only U can be non-spoiled for $r \ge 1$. We have $\mathcal{S} = \mathcal{S}' = \{A_\Gamma\}$, where A_Γ is always non-spoiled. $\square$

Lemma 4 (Restated). *Consider any round r where $1 \le r \le \frac{q-1}{2}$ in the type-Λ subnetwork. For any non-spoiled node Z for Alice (Bob) in round r that is receiving in round r, let $\mathcal{S}$ be the set of Z's neighbors under the reference adversary in round r, and $\mathcal{S}'$ be the set of Z's neighbors under the adversary simulated by Alice (Bob) in round r. Then i) all nodes in $(\mathcal{S} \setminus \mathcal{S}') \cup (\mathcal{S}' \setminus \mathcal{S})$ are receiving in round r, and ii) a node in $\mathcal{S}'$ is either B_Λ (A_Λ) or a non-spoiled node for Alice (Bob) in round $r - 1$.*

Proof. It suffices to prove the lemma for Alice. In the following, the notions of spoiled and non-spoiled nodes are always with respect to Alice. First, B_Λ is always spoiled in all rounds, and A_Λ is always non-spoiled in all rounds. For A_Λ, we have $\mathcal{S} = \mathcal{S}'$ and they contain all the top nodes on all the chains. None of these top nodes are ever spoiled for Alice. Next we consider the non-spoiled nodes on all the chains. Consider any given chain, and let U, V, and W be the nodes on the chain from top to the bottom. We will enumerate all 5 kinds of chains. Let t be any non-negative integer.

For a $|^{2t}_{2t}$ chain, U is always non-spoiled, and V and W are non-spoiled iff $r \le t$:

- For node U, we exhaustively enumerate all scenarios: i) If $r \le t$, then $\mathcal{S} = \mathcal{S}' = \{A_\Lambda, V\}$. By definition, both A_Λ and V are non-spoiled in round $r-1$. ii) If $r \ge t+1$ and since $r \le \frac{q-1}{2}$, we must have $t \le \frac{q-3}{2}$. For such t, $\mathcal{S} = \mathcal{S}' = \{A_\Lambda\}$. By definition, A_Λ is non-spoiled in round $r-1$.
- For node W and $r \le t$, we have $\mathcal{S} = \mathcal{S}' = \{V, B_\Lambda\}$, where V is non-spoiled in round $r-1$.
- Next consider node V and $r \le t$. Let V_1 (V_2) be the middle node on the chain to the left (right) of V's chain in its centipede structure. If V_1 does not exist, then $\mathcal{S} = \mathcal{S}' = \{U, W, V_2\}$. If V_2 does not exist, then $\mathcal{S} = \mathcal{S}' = \{U, W, V_1\}$. If they both exist, then $\mathcal{S} = \mathcal{S}' = \{U, W, V_1, V_2\}$. All these 4 nodes (if exist) are non-spoiled in round $r-1$.

For nodes U and W on chains in the form of $|^{2t}_{2t+1}$, $|^{2t}_{2t-1}$, $|^{2t+1}_{2t}$, or $|^{2t-1}_{2t}$, applying the exact same arguments as in the proof of Lemma 3 will show that they satisfy this lemma. The arguments remain exactly the same since for these nodes, they neighbors and the relevant adversaries' behavior is exactly the same as in a type-Γ subnetwork. So we only need to consider node V on all those kinds of chains. For any such chain, let V_1 (V_2) be the middle node on the chain to the left (right) of V's chain in its centipede structure. We first consider the case where both V_1 and V_2 exist:

- For a $|^{2t}_{2t+1}$ chain, V is non-spoiled iff $r \le t$. In such a case we have $\mathcal{S} = \mathcal{S}' = \{U, W, V_1, V_2\}$, and all these nodes are non-spoiled in round $r-1$.
- All above claims, without any modification, hold for a $|^{2t}_{2t-1}$ chain as well.
- For a $|^{2t+1}_{2t}$ chain, V is always non-spoiled. We exhaustively enumerate all scenarios: i) If $r \le t$, we have $\mathcal{S} = \mathcal{S}' = \{U, W, V_1, V_2\}$, and all these nodes are non-spoiled in round $r-1$. ii) If $r \ge t+2$, then $\mathcal{S} = \mathcal{S}' = \{U, V_1, V_2\}$, and all these nodes are non-spoiled in round $r-1$. iii) If $r = t+1$ and since V is receiving in round r, we have $\mathcal{S} = \mathcal{S}' = \{U, W, V_1, V_2\}$, and all these nodes are non-spoiled in round $r-1$.
- For a $|^{2t-1}_{2t}$ chain, V is always non-spoiled. We exhaustively enumerate all scenarios: i) If $r \le t$, we have $\mathcal{S} = \mathcal{S}' = \{U, W, V_1, V_2\}$, and all these nodes are non-spoiled in round $r-1$. ii) If $r \ge t+1$, then $\mathcal{S} = \mathcal{S}' = \{U, V_1, V_2\}$, and all these nodes are non-spoiled in round $r-1$.

The cases where V_1 or V_2 does not exist are simpler and can be proved similarly. $\square$

Lemma 5 (Restated). *Consider any given simple composition mapping of type-φ_1 and type-φ_2 subnetworks ($\varphi_1, \varphi_2 \in \{\Gamma, \Lambda, \Upsilon\}$), and any given inputs $\mathbf{x}$ and $\mathbf{y}$ to the* DisjointnessCP *problem. Let $\mathcal{G}$ be the dynamic network corresponding to $\mathbf{x}$ and $\mathbf{y}$, under this simple composition mapping. Consider the execution of any given oracle protocol over $\mathcal{G}$, under certain inputs to the nodes and certain public coin flip outcomes. Then for all round r where $0 \le r \le \frac{q-1}{2}$, Alice (Bob) can determine both the incoming and the outgoing messages of a node V in round r of that execution, based on only $\mathbf{x}$ ($\mathbf{y}$), if:*

- *V is non-spoiled for Alice (Bob) in round r.*
- *Alice (Bob) knows the oracle protocol and the public coin flip outcomes.*
- *Alice (Bob) knows the inputs to all her (his) corresponding non-spoiled nodes in the first round.*
- *Bob (Alice) always forwards to the other party all messages sent by B_Γ and B_Λ (A_Γ and A_Λ) in all rounds, as long as those nodes exist and if he (she) can determine those messages.*

Proof. We prove via an induction on r. The lemma trivially holds for round 0. Assume that the lemma holds for all rounds before round r, and we prove the lemma for round r. It suffices to prove for Alice. Consider any non-spoiled node V for Alice in round r. Since knowing a node's initial input and its incoming messages up to a certain round immediately allows Alice to generate all its outgoing messages by simulating the oracle protocol, it suffices to prove that Alice can determine the incoming messages for V in round r if V is receiving in that round. By definition of a composition mapping, we know that V's incoming messages are either from V's neighbors in V's subnetwork or from nodes in the other subnetwork via bridging edges.

We first consider V's incoming messages from V's own subnetwork. Note that V must not be in a type-Υ subnetwork since V is non-spoiled. Lemma 3 and 4 tell us that if V is receiving in a round, then Alice can determine a set $\mathcal{S}'$ (by simulating her adversary) such that all the messages sent in that round by nodes in $\mathcal{S}'$ will exactly be those incoming messages to V in its subnetwork in that round. Also by Lemma 3 and 4, a node W in $\mathcal{S}'$ is either node B_Γ or B_Λ, or a non-spoiled node for Alice in the previous round. If W is B_Γ or B_Λ, note that B_Γ and B_Λ are non-spoiled for Bob. By inductive hypothesis, Bob can determining all the incoming messages to B_Γ and B_Λ up to and including round $r-1$. Hence Bob can generate the outgoing message from B_Γ and B_Λ in round r, if B_Γ and B_Λ are sending in round r. By condition in the lemma, Bob will have already forwarded these messages to Alice, so Alice knows these messages (as incoming messages to V). Second, if W is a non-spoiled node for Alice in the previous round, by inductive hypothesis, Alice can determine all incoming messages to W up to and including round $r-1$. Alice hence can also generate the message sent by W in round r if W is sending.

We next consider V's incoming messages via the bridging edges. Since V is non-spoiled in round r, it must be non-spoiled in the first round. Thus any bridging edge incidental on V must be sensitive for Alice. Let the other end of the bridging edge be W. By definition of a simple composition mapping, we know that W must be V's neighbor regardless of the DisjointnessCP instance, and W must always be non-spoiled for Alice. By inductive hypothesis, Alice can determine all the incoming messages to W up to and including round $r-1$. Based on these messages and W's input, Alice can determine whether W is sending in round r, by simulating the oracle protocol on W. If W is sending, Alice can further determine W's outgoing message. Thus Alice can determine the incoming messages (if any) to V via the bridging edges. $\square$

Theorem 6 (Restated). *If the diameter D is unknown to the protocol beforehand, then a $\frac{1}{6}$-error Monte Carlo protocol for* CFlood *must have a time complexity of* $\Omega(\sqrt[4]{N/\log N})$ *flooding rounds. Furthermore, this holds even if the protocol knows N and the nodes' ids are from 1 to N.*

Proof. Consider any $\frac{1}{6}$-error CFlood protocol that promises to terminate within s flooding rounds over average coin flips and over every dynamic network with no more than N nodes. Let $q = 120s+1$ and $n = \frac{N-4}{3q}$. Alice and Bob in the two-party DisjointnessCP$_{n,q}(\mathbf{x},\mathbf{y})$ problem will use this oracle CFlood protocol to solve DisjointnessCP with error probability of at most $\frac{1}{6}$, which will eventually lead to the lower bound. The proof is obtained via the following steps:

- Constructing a simple composition mapping. We first construct a mapping from DisjointnessCP instances to dynamic networks, and then show that this mapping is a simple composition mapping. Given $\mathbf{x}$ and $\mathbf{y}$, we start from the corresponding type-Γ subnetwork and type-Λ subnetwork. It is easy to verify that the type-Γ subnetwork and the type-Λ subnetwork have $\frac{3}{2}n(q-1)+2$ and $\frac{3}{2}n(q+1)+2$ nodes, respectively. Hence the total number of nodes is N. If DisjointnessCP$_{n,q}(\mathbf{x},\mathbf{y}) = 1$, we let the bridging edge set be $\{(A_\Gamma, A_\Lambda), (B_\Gamma, B_\Lambda)\}$. Here A_Γ and B_Γ (A_Λ and B_Λ) are the two special nodes in the type-Γ subnetwork (type-Λ subnetwork), as described in Section 4 (Section 5). If DisjointnessCP$_{n,q}(\mathbf{x},\mathbf{y}) = 0$, then the type-$\Gamma$ subnetwork must have $\Omega(q)$ disconnected nodes that are arranged into a line. Let one end of this line be node L_Γ. The type-Λ subnetwork must have at least one $|_0^0$ chain. Let L_Λ be the middle node of an arbitrary $|_0^0$ chain in the type-Λ subnetwork. We connect the two subnetworks using the bridging edge set $\{(A_\Gamma, A_\Lambda), (B_\Gamma, B_\Lambda), (L_\Gamma, L_\Lambda)\}$.

  It is obvious that this mapping is a composition mapping. We next show that it is a simple composition mapping. For Alice, the bridging edge (A_Γ, A_Λ) is sensitive, and both end points of this edge are always non-spoiled for Alice up to round $\frac{q-1}{2}$. Furthermore, this edge is present in every dynamic network appeared in the mapping. The other two bridging edges, (B_Γ, B_Λ) and (L_Γ, L_Λ), are not sensitive to Alice. Similar arguments apply to Bob.
- Simulating the CFlood oracle protocol. Given $\mathbf{x}$ and $\mathbf{y}$, Alice and Bob will simulate the execution of the given CFlood protocol over the dynamic network obtained from the above simple composition mapping, while feeding public coin flips into this oracle protocol. The nodes in the dynamic network will have ids from 1 to N, and A_Γ will be the special node that needs to flood the token in the CFlood problem. Since the ids do not depend on $\mathbf{x}$ and $\mathbf{y}$, Alice and Bob knows all such information. Alice (Bob) will always forward to the other party all messages sent by A_Γ and A_Λ (B_Γ and B_Λ) in all rounds, as long as she (he) can determine those messages.

  Based on these conditions, Lemma 5 tells us that Alice will be able to determine the incoming and outgoing messages of all her non-spoiled nodes in all round r where $0 \le r \le \frac{q-1}{2}$. Since A_Γ is always non-spoiled for Alice, Alice can determine all the incoming message to A_Γ. Using all these incoming messages and by simulating the CFlood protocol on A_Γ, Alice monitors whether A_Γ outputs by round $\frac{q-1}{2}$ (which would indicate that the CFlood protocol has terminated). If yes, Alice claims that DisjointnessCP$_{n,q}(\mathbf{x},\mathbf{y}) = 1$. Otherwise Alice claims that DisjointnessCP$_{n,q}(\mathbf{x},\mathbf{y}) = 0$.
- Correctness of Alice's output. We next prove that Alice's claim is correct with probability at least $\frac{5}{6}$, making the above a valid reduction. If DisjointnessCP$(\mathbf{x},\mathbf{y}) = 1$, then there are no $|_0^0$ chains in the two subnetworks, and one can easily verify that the diameter of the entire dynamic network is at most 10. Since the oracle CFlood protocol promises to terminate (over average coin flips) within s flooding rounds, it should terminate (over average coin flips) within $10s$ rounds on this dynamic network. By Markov's inequality, with probability at least $\frac{5}{6}$, it terminates by round $60s = \frac{q-1}{2}$, making Alice claim that DisjointnessCP$(\mathbf{x},\mathbf{y}) = 1$.

  If DisjointnessCP$(\mathbf{x},\mathbf{y}) = 0$, then the type-$\Gamma$ subnetwork has at least $\frac{q-1}{2}$ chains in the form of $|_0^0$. The middle nodes of all these chains are arranged into a line, and the line is connected to the type-Λ subnetwork via the (L_Γ, L_Λ) edge. It is impossible for the farthest node on the line to receive A_Γ's token in the CFlood protocol within $\frac{q-1}{2} = 60s$ rounds. Since the CFlood protocol has an error probability at most $\frac{1}{6}$, with probability at least $\frac{5}{6}$, the CFlood oracle protocol cannot terminate within $60s$ rounds. Hence Alice's claim is correct with probability at least $\frac{5}{6}$.
- From communication complexity to time complexity. We have shown above that by simulating the CFlood oracle protocol, Alice and Bob can solve any DisjointnessCP$_{n,q}$ instance. During such simulation, Alice (Bob) only needs to forward to the other party all messages sent by A_Γ and A_Λ (B_Γ and B_Λ). By Corollary 2, there exist $\mathbf{x}_0$ and $\mathbf{y}_0$ such that i) DisjointnessCP$_{n,q}(\mathbf{x}_0,\mathbf{y}_0) = 1$, and ii) Alice and Bob incur $\Omega(\frac{n}{q^2}) - O(\log n)$ bits over $\mathbf{x}_0$, $\mathbf{y}_0$, and average coin flips. Hence under such $\mathbf{x}_0$ and $\mathbf{y}_0$, at least one node out of A_Γ, A_Λ, B_Γ, and B_Λ must have sent $\Omega(\frac{n}{q^2}) - O(\log n)$ bit. The dynamic network corresponding to $\mathbf{x}_0$ and $\mathbf{y}_0$ has $O(1)$ diameter. This means that under this dynamic network and under the $\mathcal{CONGEST}$ model, the oracle protocol must take $\Omega(\frac{n}{q^2 \log N})$ rounds and also $\Omega(\frac{n}{q^2 \log N})$ flooding rounds to terminate over average coin flips. Since the oracle protocol promised to terminate within s flooding rounds, we have $s = \Omega(\frac{n}{q^2 \log N}) = \Omega(\frac{N}{s^3 \log N})$, implying $s = \Omega(\sqrt[4]{N/\log N})$.

□

Theorem 7 (Restated). *If the diameter D is unknown to the protocol beforehand, then a $\frac{1}{18}$-error Monte Carlo protocol for* Consensus *must have a time complexity of* $\Omega(\sqrt[4]{N/\log N})$ *flooding rounds. Furthermore, this holds even if the protocol knows an estimate N' for N that guarantees $|\frac{N'-N}{N}| \le \frac{1}{3}$.*

Proof. Consider any $\frac{1}{18}$-error Consensus protocol that promises to terminate within s flooding rounds over average coin flips and over every dynamic network with no more than N nodes. Let $q = 192s + 1$ and $n = \frac{N-4}{3(q+1)}$. Alice and Bob in the two-party $\textsc{DisjointnessCP}_{n,q}(\mathbf{x}, \mathbf{y})$ problem will use this oracle CONSENSUS protocol to solve DISJOINTNESSCP with error probability of at most $\frac{1}{6}$, which will eventually lead to the lower bound. The proof is obtained via the following steps:

- Constructing a simple composition mapping. We first construct a mapping from DISJOINTNESSCP instances to dynamic networks, and then show that this mapping is a simple composition mapping. Given $\mathbf{x}$ and $\mathbf{y}$, we start from the corresponding type-Λ subnetwork and type-Υ subnetwork. If $\textsc{DisjointnessCP}_{n,q}(\mathbf{x}, \mathbf{y}) = 1$, then the type-$\Upsilon$ subnetwork is empty. We let the bridging edge set be an empty set. If $\textsc{DisjointnessCP}_{n,q}(\mathbf{x}, \mathbf{y}) = 0$, then both the type-$\Lambda$ and type-Υ subnetwork have at least one $|_0^0$ chain. Let L_Λ and L_Υ be the middle node of an arbitrary $|_0^0$ chain in the type-Λ subnetwork and the type-Υ subnetwork, respectively. We connect the two subnetworks using the bridging edge set $\{(L_\Lambda, L_\Upsilon)\}$. It is obvious that this mapping is a simple composition mapping, since the bridging edge (L_Λ, L_Υ) is sensitive to neither Alice nor Bob.

  The type-Λ subnetwork has $\frac{3}{2}n(q+1)+2$ nodes, while the type-Υ subnetwork has either $\frac{3}{2}n(q+1)+2$ or 0 nodes. The total number of nodes in the dynamic network is thus either $3n(q+1)+4 = N$ or $\frac{3}{2}n(q+1)+2 = \frac{1}{2}N$. Alice and Bob set $N' = \frac{2}{3}N = \frac{2}{3}(3n(q+1)+4)$, and will feed such an N' to the CONSENSUS protocol. Obviously, such an N' satisfies both $|\frac{N'-N}{N}| \le \frac{1}{3}$ and $|\frac{N'-N/2}{N/2}| \le \frac{1}{3}$.

- Construct two CONSENSUS instances. Different from the proof of Theorem 7, here Alice and Bob will need to simulate the oracle CONSENSUS protocol twice over the above dynamic network. For our discussion here, it will be convenient to imagine that the id of each node in the network is fed into the oracle protocol running on that node, as part of the input to the protocol. While the two simulations invoke the same oracle protocol over the same dynamic network, the inputs to the nodes (and hence the ids of the nodes) will be different. In other words, the two simulations correspond to two difference instances of the CONSENSUS problem.

  Let $\mathcal{I}_{01}$ be the first CONSENSUS instance. In this instance, the nodes in the type-Λ subnetwork have initial values 0 and have ids from 1 to N. If the type-Υ subnetwork is not empty (i.e., if $\textsc{DisjointnessCP}_{n,q}(\mathbf{x}, \mathbf{y}) = 0$), then the nodes there will have initial values 1 and have ids from $N+1$ to $2N$. Here note that the ids and initial values of nodes in the type-Λ subnetwork are always fixed, and do not depend on $\mathbf{x}$ and $\mathbf{y}$. A non-spoiled node for Alice (Bob) must be in the type-Λ subnetwork. Hence Alice and Bob know the inputs to all their corresponding non-spoiled nodes in the first round.

  Let $\mathcal{I}_{10}$ be the second CONSENSUS instance. Here the nodes in the type-Λ subnetwork have initial values 1, and have the exactly same ids as the corresponding nodes in the type-Υ subnetwork (when it is not empty) in $\mathcal{I}_{01}$. Next, if the type-Υ subnetwork in $\mathcal{I}_{10}$ is not empty (i.e., if $\textsc{DisjointnessCP}_{n,q}(\mathbf{x}, \mathbf{y}) = 0$), then the nodes there will have initial values 0, and have the exactly same ids as the corresponding nodes in the type-Λ subnetwork in $\mathcal{I}_{01}$. Doing so implies that when $\textsc{DisjointnessCP}_{n,q}(\mathbf{x}, \mathbf{y}) = 0$, $\mathcal{I}_{01}$ and $\mathcal{I}_{10}$ are actually the same. By same argument as earlier, here in $\mathcal{I}_{10}$ Alice and Bob know the inputs to all their corresponding non-spoiled nodes in the first round.

  Finally, in each simulation, Alice (Bob) will always forward to the other party all messages sent by A_Λ (B_Λ) in all rounds, as long as she (he) can determine those messages.

- Simulating the oracle CONSENSUS protocol twice. Given $\mathbf{x}$ and $\mathbf{y}$, Alice and Bob will simulate the execution of the CONSENSUS oracle protocol over instances $\mathcal{I}_{01}$ and $\mathcal{I}_{10}$. When doing so, Alice and Bob will first generate all public coin flip outcomes (denoted as z), and then feed the same z into both simulations.

  Lemma 5 tells us that in each simulation, Alice can determine the incoming and outgoing messages of all her non-spoiled nodes in all round r where $0 \le r \le \frac{q-1}{2}$. Since A_Λ is always non-spoiled for Alice, Alice can determine all the incoming messages to A_Λ. Using all these messages and by simulating the oracle CONSENSUS protocol on A_Λ, Alice monitors when A_Λ decides in each of the two simulations. If A_Λ decides in *both* $\mathcal{I}_{01}$ *and* $\mathcal{I}_{10}$ by the end of round $\frac{q-1}{2}$, Alice claims that $\textsc{DisjointnessCP}_{n,q}(\mathbf{x}, \mathbf{y}) = 1$. Otherwise Alice claims that $\textsc{DisjointnessCP}_{n,q}(\mathbf{x}, \mathbf{y}) = 0$.

- Correctness of Alice's output. We next prove that Alice's claim is correct with probability at least $\frac{5}{6}$, which would make the above a valid reduction. If $\textsc{DisjointnessCP}(\mathbf{x}, \mathbf{y}) = 1$, the type-$\Upsilon$ subnetwork is empty. One can easily verify that the diameter of the overall dynamic network is at most 8. Since the oracle CONSENSUS protocol promises to terminate (over average coin flips) within s flooding rounds, it should terminate (over average coin flips) within $8s$ rounds on this dynamic network in instance $\mathcal{I}_{01}$. By Markov's inequality, with probability at least $\frac{11}{12}$, A_Λ decides by round $96s = \frac{q-1}{2}$ in instance $\mathcal{I}_{01}$. Similarly, with probability at least $\frac{11}{12}$, A_Λ decides by round $96s = \frac{q-1}{2}$ in instance $\mathcal{I}_{10}$. By union bound, we know that with probability at least $\frac{5}{6}$, A_Λ decides by round $\frac{q-1}{2}$ in both instances, making Alice claim that $\textsc{DisjointnessCP}(\mathbf{x}, \mathbf{y}) = 1$.

  If $\textsc{DisjointnessCP}(\mathbf{x}, \mathbf{y}) = 0$, then the type-$\Upsilon$ subnetwork is not empty, and the entire dynamic network's topology is symmetric along the edge (L_Λ, L_Υ). It suffices to prove that with probability at most $\frac{1}{6}$, A_Λ decides in both $\mathcal{I}_{01}$ and $\mathcal{I}_{10}$ within $\frac{q-1}{2}$ rounds. To prove this, first consider A_Λ and A_Υ in $\mathcal{I}_{01}$. It is easy to verify that under all possible coin flip outcomes of the CONSENSUS protocol, for all node U where $(U, 0) \rightsquigarrow (A_\Lambda, \frac{q-1}{2})$, U has initial value of 0 in $\mathcal{I}_{01}$. Similarly, all U's where $(U, 0) \rightsquigarrow (A_\Upsilon, \frac{q-1}{2})$ have initial value of 1. By Lemma 9 (described and proved next), with probability at most $3 \times \frac{1}{18} = \frac{1}{6}$, A_Λ and A_Υ in $\mathcal{I}_{01}$ both decide within $\frac{q-1}{2}$ rounds. Next recall that Alice and Bob feed the same public coin flip outcomes z into both simulations. Under a given z, the two instances $\mathcal{I}_{01}$ and $\mathcal{I}_{10}$ are actually exactly the same — $\mathcal{I}_{10}$ can be viewed as flipping $\mathcal{I}_{01}$ along the edge (L_Λ, L_Υ). This means that the behavior A_Υ in $\mathcal{I}_{01}$ and the behavior of A_Λ in $\mathcal{I}_{10}$ must be exactly the same. Hence A_Λ in $\mathcal{I}_{10}$ decides iff A_Υ in $\mathcal{I}_{01}$ decides. Thus with probability at most $\frac{1}{6}$, A_Λ decides in both $\mathcal{I}_{01}$ and $\mathcal{I}_{10}$ within $\frac{q-1}{2}$ rounds.

- From communication complexity to time complexity. We have shown above that by simulating the CONSENSUS oracle protocol twice, Alice and Bob can solve any DISJOINTNESSCP$_{n,q}$ instance. During each simulation, Alice (Bob) only needs to forward to the other party all messages sent by A_Λ (B_Λ). By Corollary 2, there exist $\mathbf{x}_0$ and $\mathbf{y}_0$ such that i) DISJOINTNESSCP$_{n,q}(\mathbf{x}_0, \mathbf{y}_0) = 1$, and ii) Alice and Bob incur $\Omega(\frac{n}{q^2}) - O(\log n)$ bits over $\mathbf{x}_0$, $\mathbf{y}_0$, and average coin flips. Hence under such $\mathbf{x}_0$ and $\mathbf{y}_0$, at least one node in the two CONSENSUS instances must have sent $\Omega(\frac{n}{q^2}) - O(\log n)$ bit. The dynamic network corresponding to $\mathbf{x}_0$ and $\mathbf{y}_0$ has $O(1)$ diameter. This means that under this dynamic network and under the $\mathcal{CONGEST}$ model, the oracle protocol must take $\Omega(\frac{n}{q^2 \log N})$ rounds and also $\Omega(\frac{n}{q^2 \log N})$ flooding rounds to terminate over average coin flips. Since the oracle protocol promised to terminate within s flooding rounds, we have $s = \Omega(\frac{n}{q^2 \log N}) = \Omega(\frac{N}{s^3 \log N})$, implying that $s = \Omega(\sqrt[4]{N/\log N})$.

□

LEMMA 9. *Consider any given δ-error* CONSENSUS *protocol for dynamic networks, and any given adversary for determining the dynamic network. Let r be any given round, and let z denote all the (public or private) coin flip outcomes of the protocol. Consider any given initial values to the nodes in the dynamic network. Let nodes V_0 and V_1 be any two nodes such that under the given initial values and under every z, all nodes in $\Psi(V_i)$ in the dynamic network as determined by the adversary have initial values i for $i \in \{0, 1\}$, where $\Psi(V_i) = \{U | (U, 0) \rightsquigarrow (V_i, r)\}$. Then under the given initial values, with probability at most 3δ, both V_0 and V_1 decide by the end of round r.*

PROOF. Let the adversary given in the lemma be $\mathcal{A}$. We first construct another adversary $\mathcal{B}$. To fully specify $\mathcal{B}$, we need to specify the dynamic network (i.e., the sequence of graphs) constructed by $\mathcal{B}$, for all possible initial values to the nodes and all possible coin flip outcomes of the CONSENSUS protocol. Consider any given outcome z of all the coin flips. Under the given initial values in the lemma and under such z, let $\mathcal{G}$ be the the unique dynamic network as determined by $\mathcal{A}$. $\mathcal{G}$ may be different for different z's if $\mathcal{A}$ is adaptive. We design $\mathcal{B}$ such that $\mathcal{B}$ constructs the dynamic network to be exactly the same as $\mathcal{G}$, under z and under *all* possible initial values to the nodes.

Let the CONSENSUS instance as described in the lemma be $\mathcal{I}$, where for all z, all nodes in $\Psi(V_0)$ and all nodes in $\Psi(V_1)$ have initial value of 0 and 1, respectively. We construct two additional CONSENSUS instances, $\mathcal{I}_0$ and $\mathcal{I}_1$, under the adversary $\mathcal{B}$. All nodes have initial values 0 in $\mathcal{I}_0$, while all nodes have initial values 1 in $\mathcal{I}_1$. The ids of the corresponding nodes in the three CONSENSUS instances are always the same.

Consider a given outcome z of all the coin flips of the protocol. We will prove that if both V_0 and V_1 decide by the end of round r in $\mathcal{I}$ under z, then the CONSENSUS protocol must err in either $\mathcal{I}$ or $\mathcal{I}_0$ or $\mathcal{I}_1$ under z. We consider two cases. If the protocol errs in the $\mathcal{I}$ instance, we are done. If the protocol does not err in $\mathcal{I}$, without loss of generality, assume that both V_0 and V_1 decide on 1. Consider V_0's behavior in the instance $\mathcal{I}_0$. Note that in $\mathcal{I}_0$, the initial values of all nodes in $\Psi(V_0)$ are 0, which is exactly the same as the initial values of the nodes in $\Psi(V_0)$ in $\mathcal{I}$. In any instance, only nodes in $\Psi(V_0)$ can potentially influence V_0's behavior in round r. Hence V_0's behavior in $\mathcal{I}_0$ and $\mathcal{I}$ must be exactly the same up to round r. Since V_0 decides on 1 by the end of round r in $\mathcal{I}$, V_0 must decide on 1 in $\mathcal{I}_0$ as well. But this decision is wrong for $\mathcal{I}_0$.

Finally, the lemma follows since for each of $\mathcal{I}$ (under adversary $\mathcal{A}$), $\mathcal{I}_0$ (under adversary $\mathcal{B}$), and $\mathcal{I}_1$ (under adversary $\mathcal{B}$), the probability that the protocol errs is at most δ. □

B. DETAILS OF OUR LEADERELECT PROTOCOL

Pseudo-code. Algorithm 1 presents the pseudo-code of our LEADERELECT protocol. It uses three subroutines, **Flood** (Algorithm 2), **FindLargestId** (Algorithm 3), and **HasMajority** (Algorithm 4). In **Flood**, each node has an initial non-negative input. If a node's input is positive, it will keep sending this input to its neighbors. If a node's input is 0, then it will keep waiting for incoming messages, and adopt the first value received. After that, it will keep sending the value adopted. **FindLargestId** allows every node to find out the largest id within a certain number of hops of that node. Since a node cannot simultaneously send and receive, the flooding of the largest id is done implicitly via a binary search instead of via a naive flooding. In **HasMajority**, each node has an initial input. The subroutine enables a node to tell, under certain conditions, whether its input is also held by a majority of the nodes.

Our LEADERELECT protocol has total $\lceil \log 4N' \rceil$ phases. It maintains a guess D' for D, with D' doubling in each phase. Each phase has 5 sub-phases (see Algorithm 1). The first sub-phase tries to roll back unsuccessful locking done in previous phases. (We say "tries" since the rolling back may not be fully successful if $D' < D$.) The second sub-phase tries to first find the largest id throughout the network, and then let each node test whether the id found has been adopted (i.e., believed to be the largest id) by a majority of the nodes in the system. If a node finds that its own id has been adopted by a majority, in the third sub-phase, it tries to lock all the nodes in the system and then checks whether it successfully locks at least a majority. If yes, it becomes the leader in the fourth sub-phase, and tries to notify other nodes in the last sub-phase. If no, it sets a flag, which will later unlock the nodes in future phases.

Properties of the three subroutines. In the following, we first prove the properties of the three subroutines.

LEMMA 10. *Consider any $D' > 0$ and $z \geq 0$ (note that z can be 0).*

- *If every node either does nothing in the next D' rounds or invokes* **Flood**(z, D') *or invokes* **Flood**$(0, D')$*, then after exactly D' rounds, a* **Flood**(z, D') *invocation must return z and a* **Flood**$(0, D')$ *invocation must return either z or 0.*

Algorithm 1 Our LeaderElect protocol. // Elect a leader and output the leader's id.

```
1: // $N'$ and $c$ are input parameters.
2: $D' = 1$; $leader\_id = 0$; $phase\_locked = 0$; $unlock[i] = 0$ for $i \in [1, \lceil \log(4N') \rceil]$;
3: for $current\_phase = 1$ to $\lceil \log(4N') \rceil$ do
4:     $D' = 2D'$;

5:     /* Unlock sub-phase ($O(D' \log N)$ rounds) */
6:     // for rolling back unsuccessful locking done in all previous phases
7:     $unlock[i]$ = Flood($unlock[i], D'$) for $i \in [1, current\_phase]$;
8:     if ($phase\_locked > 0$ and $unlock[phase\_locked] \neq 1$) then
9:         wait for $(D' \cdot (1 + id\_length + 2\lceil \frac{100}{c^2} \ln(10N') \rceil))$ rounds and then jump to Line 30;
10:        // locked nodes only execute the notification sub-phase; here $id\_length$ is the length of the ids
11:    end if

12:    /* Finding largest id sub-phase ($O(D' \log N)$ rounds) */
13:    $largest\_id$ = FindLargestId($D'$);
14:    $id\_majority$ = HasMajority($largest\_id, D'$); // whether $largest\_id$ is the majority
15:    if $my\_id \neq largest\_id$ then $id\_majority = 0$; // here $my\_id$ is my own id

16:    /* Lock acquiring sub-phase ($O(D' \log N)$ rounds) */
17:    $acquire\_lock = id\_majority$;
18:    $acquire\_lock$ = Flood($acquire\_lock, D'$);
19:    if $acquire\_lock = 1$ then
20:        $phase\_locked = current\_phase$;
21:    end if
22:    $lock\_majority$ = HasMajority($acquire\_lock, D'$);

23:    /* Decision sub-phase (0 round) */
24:    if ($id\_majority = 1$ and $lock\_majority = 0$) then
25:        $unlock[current\_phase] = 1$;
26:    end if
27:    if ($id\_majority = 1$ and $lock\_majority = 1$) then
28:        $leader\_id = my\_id$;
29:    end if

30:    /* Notification sub-phase ($O(D')$ rounds) */
31:    $leader\_id$ = Flood($leader\_id, D'$);
32:    if ($leader\_id > 0$ and has not already output) then output $leader\_id$;
33: end for
34: if has not already output then output $my\_id$; // probability of doing this is vanishingly small
```

Algorithm 2 Flood (z, D'). // Flood the input z and overwrite nodes with input 0 (but not other nodes).

```
1: for $i = 1$ to $D'$ do
2:     if $z > 0$ then send $z$;
3:     else
4:         receive messages;
5:         if message with value $z'$ is received then $z = z'$;
6:         (If multiple messages are received,
7:         simply pick an arbitrary one.)
8:     end if
9: end for
10: return $z$;
```

Algorithm 3 FindLargestId(D'). // Find out the largest id within a range of D'.

```
1: $tmp = my\_id$;    // here $my\_id$ is my own id
2: for $i = 1$ to $id\_length$ do    // here $id\_length$ is the length of the ids, which is $O(\log N)$
3:     $tmp[i]$ = Flood($tmp[i], D'$);
4:     // here $tmp[i]$ is the $i$th bit in $tmp$, with $tmp[1]$ being the most significant bit
5:     if $tmp[i] \neq my\_id[i]$ then $tmp[j] = 0$ for $j \in [i+1, id\_length]$
6: end for
7: return $tmp$;
```

Algorithm 4 HasMajority(z, D'). // Check whether majority of the nodes' inputs are z.

```
1: $a = \lceil \frac{100}{c^2} \ln(10N') \rceil$; $b = 1 - (1 - \frac{1}{N'})^{3N'/4}$; $sum = 0$;
2: for $i = 1$ to $a$ do
3:     set $tmp = z$ with probability $\frac{1}{N'}$, and $tmp = 0$ with probability $1 - \frac{1}{N'}$;
4:     $tmp$ = Flood($tmp, D'$);
5:     if $tmp = z$ then $sum = sum + 1$;
6: end for
7: if $z = 0$ then return 0;
8: if $N' < \frac{4}{3}$ then return 1;
9: if $sum \geq ab$ then return 1;
10: return 0;
```

- *If i) $D' \geq D$, ii) there is at least one node invoking* **Flood**(z, D')*, and iii) all other nodes invoke* **Flood**$(0, D')$*, then all* **Flood** *invocations return z after exactly D' rounds.*

PROOF.

- Trivial since a positive value never gets overwritten in the protocol and the return value of **Flood** must be some node's input.
- Directly from $D' \geq D$ and the fact that D is the diameter of the dynamic network.

□

LEMMA 11. *If $D' \geq D$ and every node invokes* **FindLargestId**(D')*, then all such invocations return the largest id among of all the nodes after $O(D' \log N)$ rounds.*

PROOF. All line numbers below refer to Algorithm 3. We first show that the return values on all the nodes must be identical. For any given node, consider the ith bit in the *tmp* variable. This bit is last set by **Flood** at Line 3 in the ith iteration. By Lemma 10, all nodes must assign the same value to *tmp*$[i]$ at Line 3 in the ith iteration.

We next prove that the return value must be the largest id among all the nodes. Let V be the node with the largest id. It suffices to prove that the **FindLargestId** invocation on V returns V's own id. We prove by contradiction and assume that on V, *my_id* $\neq$ *tmp* when **FindLargestId** returns at Line 7. Let k be the smallest index such that *my_id*$[k] \neq$ *tmp*$[k]$ at Line 7. Since *tmp*$[k]$ = *my_id*$[k]$ immediately after Line 1, we know that *tmp*$[k]$ must have been changed at Line 3 of the kth iteration, or at Line 5 in earlier iterations. We next show that Line 5 in earlier iterations will not modify *tmp*$[k]$: For all $k' < k$, we know that *my_id*$[k']$ = *tmp*$[k']$ at Line 7. The last time that *my_id*$[k']$ can be modified is at Line 3 of the k'th iteration. Since *my_id*$[k']$ = *tmp*$[k']$ at Line 7, we know that at Line 5 of the k'th iteration, *my_id*$[k']$ = *tmp*$[k']$ must also hold. This implies that in the k'th iteration, the condition at Line 5 will not be met and *tmp*$[k]$ will not be modified at Line 5.

Hence *tmp*$[k]$ must have been modified at Line 3 of the kth iteration. There are only two cases after that line: Either *my_id*$[k] = 1$ and *tmp*$[k] = 0$, or *my_id*$[k] = 0$ and *tmp*$[k] = 1$. By Lemma 10 on the property of **Flood**, the first case is not possible. Hence we must have *my_id*$[k] = 0$ and *tmp*$[k] = 1$. We also know that there must be some other node U, which invoked **Flood** at Line 3 of the kth iteration with *tmp*$[k] = 1$. Since *tmp*$[k] = 1$ on U in the kth iteration at that time, we immediately know that *my_id*$[k]$ must be 1 on U since otherwise *tmp*$[k]$ can never be 1. We also know that the condition at Line 5 on U must have never been satisfied in all earlier iterations on U — otherwise U would have set *tmp*$[k] = 0$ in one of those earlier iterations. In other words, for all $k' < k$, *tmp*$[k']$ = *my_id*$[k']$ on U at the end of the k'th iteration. Since *tmp*$[k']$ is never modified after the k'th iteration, we know that for all $k' < k$, *tmp*$[k']$ = *my_id*$[k']$ on U at Line 7.

Recall that we showed earlier that *tmp* must be identical at Line 7 on all nodes. We also proved earlier that for all $k' < k$, we have *tmp*$[k']$ = *my_id*$[k']$ on both V and U, implying that the first k' bits in U's and V's id are identical. Finally, we know that *my_id*$[k] = 0$ on V and *my_id*$[k] = 1$ on U, implying that U's id is larger than V's. This contradicts with V having the largest id. □

LEMMA 12. *Let $b = 1 - (1 - \frac{1}{N'})^{3N'/4}$. For any $N' > \frac{4}{3}$, we have:*

$$1 - e^{-3/4} < b < \frac{3}{4}$$

For any $b \in (1 - e^{-3/4}, \frac{3}{4})$ and $c \in (0, \frac{1}{3}]$, we have:

$$\begin{aligned} 1 - (1-b)^{\frac{1}{1+\frac{3}{2}c}} &< b - 0.5e^{-\frac{1}{2}}c \\ 1 - (1-b)^{\frac{1}{1-\frac{3}{4}c}} &> b + 0.25c \end{aligned}$$

PROOF. We prove the three inequalities one by one.

- Consider b as a function of N', and we have:

$$b'(N') = -\frac{3(\frac{N'-1}{N'})^{3N'/4}((N'-1)\ln(\frac{N'-1}{N'}) + 1)}{4(N'-1)}$$

  Hence $b'(N') < 0$ for all $N' > \frac{4}{3}$, implying that $b(N')$ is monotonically decreasing. The inequality follows since $b(\frac{4}{3}) = \frac{3}{4}$ and $b(\infty) = 1 - e^{-3/4}$.

- Define $g(c) = 1 - (1-b)^{\frac{1}{1+\frac{3}{2}c}}$, and we have:

$$\begin{aligned} g'(c) &= \frac{6(1-b)^{\frac{2}{3c+2}}\ln(1-b)}{(3c+2)^2} \\ g''(c) &= -\frac{36(1-b)^{\frac{2}{3c+2}}\ln(1-b)(\ln(1-b)+3c+2)}{(3c+2)^4} \end{aligned}$$

  For $b \in (1 - e^{-3/4}, \frac{3}{4})$ and $c \in (0, \frac{1}{3}]$, we have $g''(c) > 0$ and hence $g'(c)$ is monotonically increasing. Using Taylor expansion with Lagrange's remainder, for any $c \in (0, \frac{1}{3}]$, there must exist $c' \in [0, c]$ such that $g(c) = g(0) + cg'(c') \leq b + cg'(\frac{1}{3})$. We next only need to prove that $g'(\frac{1}{3}) < -0.5e^{-\frac{1}{2}}$.

  Let $h(b) = g'(\frac{1}{3}) = \frac{2}{3}(1-b)^{\frac{2}{3}}\ln(1-b)$. We have $h'(b) = -\frac{2(2\ln(1-b)+3)}{9(1-b)^{\frac{1}{3}}}$, which is always negative for $b \in (1 - e^{-3/4}, \frac{3}{4})$. Hence $h(b)$ is monotonically decreasing, and we have $g'(\frac{1}{3}) = h(b) < h(1 - e^{-3/4}) = -0.5e^{-\frac{1}{2}}$.

- Define $g(c) = 1 - (1-b)^{\frac{1}{1-\frac{3}{4}c}}$, and we have:

$$\begin{aligned} g'(c) &= -\frac{12(1-b)^{\frac{4}{4-3c}}\ln(1-b)}{(4-3c)^2} \\ g''(c) &= -\left(\frac{\ln(1-b)}{2(1-\frac{3c}{4})} + 1\right)\left(\frac{9(1-b)^{\frac{1}{1-\frac{3c}{4}}}\ln(1-b)}{8(1-\frac{3c}{4})^3}\right) \end{aligned}$$

  For $b \in (1 - e^{-3/4}, \frac{3}{4})$ and $c \in (0, \frac{1}{3}]$, we have $g''(c) > 0$ and hence $g'(c)$ is monotonically increasing. Using Taylor expansion with Lagrange's remainder, for any $c \in (0, \frac{1}{3}]$, there must exists $c' \in [0, c]$ such that $g(c) = g(0) + cg'(c') \geq b + cg'(0)$. We next only need to prove that $g'(0) > 0.25$.

  Let $h(b) = g'(0) = -\frac{3}{4}(1-b)\ln(1-b)$, and thus $h'(b) = \frac{3}{4}(\ln(1-b)+1)$. We have $h'(b) > 0$ for $b \in (1-e^{-3/4}, 1-e^{-1})$, $h'(b) = 0$ for $b = 1 - e^{-1}$, and $h'(b) < 0$ for $b \in (1-e^{-1}, \frac{3}{4})$. Hence for all $b \in (1-e^{-3/4}, \frac{3}{4})$, $g'(0) = h(b) \geq \min(h(1 - e^{-3/4}), h(\frac{3}{4})) = h(\frac{3}{4}) = \frac{3}{16}\ln(4) > 0.25$.

□

Lemma 13. *Consider any given $D' > 0$. If every node either does nothing in the next D' rounds or invokes* **HasMajority**(z, D') *(where z can be different for different nodes), then all such* **HasMajority** *invocations return within $O(D' \log N)$ rounds. Furthermore, consider any given node V that invokes* **HasMajority**(z_0, D'). *With probability of at least $1 - \frac{1}{10N^3}$, V's invocation is* good *in the sense that:*

- *If total no more than $N/2$ nodes invoke* **HasMajority**(z_0, D')*, then V's invocation returns 0.*
- *If $z_0 > 0$, $D' \geq D$, and all N nodes invoke* **HasMajority**(z_0, D')*, then V's invocation returns 1.*

Proof. All line numbers below refer to Algorithm 4. It is obvious that **HasMajority** returns within $O(D' \log N') = O(D' \log N)$ rounds. If $z_0 = 0$, then the condition at Line 7 will be met and Line 7 will return 0, in which case V's invocation must be good. If $z_0 > 0$ and $N = 1$, we must have $N' \leq \frac{4}{3} - c < \frac{4}{3}$. Then the condition at Line 8 will be met and Line 8 will return 1, in which case V's invocation must be good as well. We next prove that V's invocation is good when $z_0 > 0$ and $N \geq 2$, which implies that $N' > \frac{4}{3}$. Define binary random variable X_i which equals 1 iff there exists at least one node having $tmp = z_0$ immediately after Line 3 in the ith iteration. We next prove the two claims that need to hold for V's invocation to be good:

- For the case where no more than $\frac{N}{2}$ nodes invoke **HasMajority**(z_0, D'), by Lemma 10, we know that the condition of $tmp = z_0$ in Line 5 of the ith iteration may hold only if $X_i = 1$. Hence at Line 9, we have $sum \leq \sum_{i=1}^{a} X_i$. We also have $E[X_i] \leq 1 - (1 - \frac{1}{N'})^{\frac{N}{2}} \leq 1 - (1 - \frac{1}{N'})^{\frac{N'}{\frac{4}{3}+2c}} = 1 - (1-b)^{\frac{1}{1+\frac{3}{2}c}} < b - 0.5e^{-\frac{1}{2}}c < 1$, where the second to the last step is from Lemma 12. Define binary random variable Y_i to be 1 with probability exactly $b - 0.5e^{-\frac{1}{2}}c$. By a simple coupling argument and by Chernoff bound, we have:

$$\begin{aligned}
&\Pr\left[\sum_{i=1}^{a} X_i \geq ab\right] \leq \Pr\left[\sum_{i=1}^{a} Y_i \geq ab\right] \\
= \;& \Pr\left[\sum_{i=1}^{a} Y_i \geq \left(1 + \frac{0.5e^{-\frac{1}{2}}c}{b - 0.5e^{-\frac{1}{2}}c}\right) \cdot (a(b - 0.5e^{-\frac{1}{2}}c))\right] \\
\leq \;& \exp\left(-\frac{(0.5e^{-\frac{1}{2}}c)^2 \cdot (a(b - 0.5e^{-\frac{1}{2}}c))}{3(b - 0.5e^{-\frac{1}{2}}c)^2}\right) \\
\leq \;& \exp\left(-\frac{4}{9}(0.5e^{-\frac{1}{2}}c)^2 \cdot \lceil \frac{100}{c^2} \ln(10N') \rceil\right) \\
& \text{(by the first inequality in Lemma 12)} \\
\leq \;& \exp\left(-3\ln(10N')\right) < \exp\left(-3\ln(\frac{20}{3}N)\right) < \frac{1}{10N^3}
\end{aligned}$$

  Hence the probability of V's invocation returning 1 is $\Pr[sum \geq ab] \leq \Pr[\sum_{i=1}^{a} X_i \geq ab] < \frac{1}{10N^3}$.

- For the case where $z_0 > 0$, $D' \geq D$, and all N nodes invoke **HasMajority**(z_0, D'), Lemma 10 tells us that the condition of $tmp = z_0$ in Line 5 of the ith iteration holds if and only if $X_i = 1$. Hence at Line 9, we must have $sum = \sum_{i=1}^{a} X_i$. We also have $E[X_i] = 1 - (1 - \frac{1}{N'})^N \geq 1 - (1 - \frac{1}{N'})^{\frac{N'}{\frac{4}{3}-c}} = 1 - (1-b)^{\frac{1}{1-\frac{3}{4}c}} > b + 0.25c$, where the last step is from Lemma 12. Define binary random variable Y_i to be 1 with probability exactly $b + 0.25c$. Note that $0 < b + 0.25c < 1$ since $0 < c \leq \frac{1}{3}$ and $1 - e^{-3/4} < b < \frac{3}{4}$ by Lemma 12. By a simple coupling argument and by Chernoff bound, we have:

$$\begin{aligned}
&\Pr\left[\sum_{i=1}^{a} X_i < ab\right] \\
\leq \;& \Pr\left[\sum_{i=1}^{a} Y_i \leq ab\right] \\
= \;& \Pr\left[\sum_{i=1}^{a} Y_i \leq \left(1 - \frac{0.25c}{b + 0.25c}\right) \cdot (a(b + 0.25c))\right] \\
\leq \;& \exp\left(-\frac{(0.25c)^2 \cdot (a(b + 0.25c))}{2(b + 0.25c)^2}\right) \\
< \;& \exp\left(-\frac{3}{100}c^2 \cdot \lceil \frac{100}{c^2} \ln(10N') \rceil\right) \\
& \text{(by the first inequality in Lemma 12)} \\
\leq \;& \exp\left(-3\ln(10N')\right) < \exp\left(-3\ln(\frac{20}{3}N)\right) < \frac{1}{10N^3}
\end{aligned}$$

  Hence the probability of V's invocation returning 0 is $\Pr[sum < ab] = \Pr[\sum_{i=1}^{a} X_i < ab] < \frac{1}{10N^3}$.

□

Time complexity and correctness of our LeaderElect **protocol.** Having proved the properties of the three subroutines, we are now ready to prove the properties of our LeaderElect protocol. All line numbers in the proofs below refer to Algorithm 1.

Lemma 14. *Conditioned upon the event that all* **HasMajority** *invocations (at Line 14 and Line 22 of Algorithm 1) are good, in each phase of Algorithm 1, at most one node has id_majority = 1 immediately after Line 15.*

Proof. The variable *id_majority* can only be modified in Line 14 and Line 15. If *id_majority* = 1 immediately after Line 15, we know that **HasMajority** returns 1 at Line 14 and *my_id* = *largest_id* at Line 15. We say that a node is *popular* iff its *largest_id* variable immediately before Line 14 equals the *largest_id* variable immediately before Line 14 on more than $\frac{N}{2}$ nodes.

For all nodes that are not popular, Lemma 13 tells us that **HasMajority** returns 0 on all such nodes and hence they will not have *id_majority* = 1 immediately after Line 15. For all nodes that are popular, the values of *largest_id* at Line 14 must all be the same. Let the value be z_0. Since every node has a unique id, there will be at most one node whose id is z_0. Only this one node may satisfy the condition of *my_id* = *largest_id* at Line 15, and hence only this node may have *id_majority* = 1 immediately after Line 15. □

Lemma 15. *Conditioned upon the event that all* **HasMajority** *invocations (at Line 14 and Line 22 of Algorithm 1) are good, at most one node executes Line 28 of Algorithm 1.*

Proof. Let k be the value of *current_phase* when some node V first executes Line 28. If V does not exist, we are done. If V exists, it means that we have *id_majority* $= 1$ and *lock_majority* $= 1$ on V immediately before Line 28. By Lemma 14, at most one node has *id_majority* $= 1$ immediately after Line 15 in the kth phase. Hence V must be this node, and no other nodes have *id_majority* $= 1$. Since reaching Line 28 requires *id_majority* $= 1$, no other nodes will execute Line 28 in the kth phase. We next prove that no node (not even V) will execute Line 28 in later phases.

To do so, we need to first examine further the nodes' behavior in phase k. Since *id_majority* $= 1$ and *lock_majority* $=$ 1 on V immediately before Line 28, the **HasMajority**(*acquire_lock*, D') invocation at Line 22 on V must have returned 1. The parameter *acquire_lock* used for this invocation must equal 1, since *acquire_lock* was set to 1 at Line 17 and by Lemma 10, it must be 1 even after Line 18. Since this **HasMajority**$(1, D')$ invocation on V is good, there must be more than $\frac{N}{2}$ nodes that invoke **HasMajority**$(1, D')$ at Line 22 of the kth phase. (Otherwise by the definition of "good" in Lemma 13, **HasMajority**$(1, D')$ on V would have returned 0.) Let us call all these nodes as *locked nodes*. All these locked nodes must have had *acquire_lock* $= 1$ at Line 19, and hence must have set *phase_locked* $= k$ at Line 20.

Next, because V is the only node with *id_majority* $= 1$ in the kth phase and V has *lock_majority* $= 1$, we know that no node will execute Line 25 in the kth phase and *unlock*$[k]$ will never be set to 1 at Line 25 in phase k. Lemma 10 in turn tells us that *unlock*$[k]$ will never be set to 1 at Line 7 in any phase on any node either. Now since *phase_locked* $= k$ on all locked nodes at Line 20 of the kth phase, and since *unlock*$[k]$ is always 0 on all nodes in all phases, all the locked nodes will satisfy the condition at Line 8 in the $(k+1)$th phase, and hence skip all sub-phases in the $(k+1)$th phase except the notification sub-phase. In particular, skipping the lock acquiring sub-phase during the $(k+1)$th phase means that *phase_locked* will remain k at the end of the $(k+1)$th phase. In turn, these locked nodes will skip all sub-phases except for the notification sub-phase in the $(k+2)$th phase. Repeating such arguments immediately shows that in all phases after phase k, all these locked nodes will skip all sub-phases except for the notification sub-phase .

Now since more than $\frac{N}{2}$ nodes will skip all sub-phases except for the notification sub-phase, there will be less than $\frac{N}{2}$ nodes left. This means that in all phases after phase k, when nodes invoke **HasMajority** at Line 14, all such invocations will return 0. This prevents any node from executing Line 28 in any phase after phase k. $\square$

LEMMA 16. *Conditioned upon the event that all* **HasMajority** *invocations (at Line 14 and Line 22 of Algorithm 1) are good, at least one node executes Line 28 of Algorithm 1 by the end of the first phase where D' has become larger than or equal to D.*

PROOF. Let k be the first phase where $D' \geq D$. If some node has executed Line 28 by the end of phase $k-1$, we are done. If no node has done so, consider phase k.

We first prove that no node will skip any sub-phase in phase k. At Line 4 of phase k, we claim that if *phase_locked* $= k' > 0$ for any k' on any node V, then $k' < k$ and *unlock*$[k']$ must be 1 on some (potentially different) node W at Line 4 of phase k. This claim holds since the only way to set *phase_locked* to k' is at Line 20 of phase k' where $k' < k$. This requires *acquire_lock* $= 1$ at Line 19 of phase k'. Some node W must have *acquire_lock* $= 1$ immediately before Line 18 since otherwise all node would have invoked **Flood**$(0, D')$ and by Lemma 10, all of them would return 0. Hence W has *id_majority* $= 1$ at Line 17 of phase k'. With *id_majority* $= 1$ and since W does not execute Line 28 in phase k', it must execute Line 25 in phase k' and will have *unlock*$[k'] = 1$. Once *unlock*$[k']$ becomes 1 on W, Lemma 10 tells us that *unlock*$[k']$ on W will never be changed to 0 again later at Line 7 in future phases. Hence *unlock*$[k']$ must be 1 on W at Line 4 of phase k. Now consider nodes V and W in phase k. Since *unlock*$[k'] = 1$ on W at Line 4 of phase k and since $D' \geq D$, by Lemma 10, *unlock*$[k']$ must be 1 on V immediately after Line 7. This means that V will not skip any of the sub-phases in phase k, or more precisely, the condition at Line 8 on V will not be met in phase k.

So we have shown that no node will skip any sub-phase in phase k. Since $D' \geq D$, Lemma 11 tells us that immediately after Line 13 in phase k, all nodes will have identical *largest_id*, which equals the largest id among all the nodes. All the nodes will then invoke **HasMajority** with the same *largest_id* at Line 14. Since we conditioned upon all **HasMajority** invocations being good, all such invocations will return 1. Then at Line 15, the node with the largest id will have *id_majority* $= 1$, while all other nodes will have *id_majority* $= 0$. The node with the largest id will then set *acquire_lock* to be 1. After all the nodes invoke Line 18, again by Lemma 10, all of them will have *acquire_lock* $= 1$. Next, all nodes will have *lock_majority* $= 1$ immediately after Line 22. Finally, the node with *id_majority* $= 1$ will execute Line 28. $\square$

THEOREM 8. *Let $c \in (0, \frac{1}{3}]$ be any given constant and N' be any value such that $|\frac{N'-N}{N}| \leq \frac{1}{3} - c$. Consider the* LEADERELECT *problem where c and N' are known to the protocol. Then even if the diameter D is unknown to the protocol beforehand, there exists a $\frac{1}{N}$-error Monte Carlo* LEADERELECT *protocol with a time complexity of $O(\log N)$ flooding rounds.*

PROOF. We will prove that Algorithm 1 satisfies the properties needed by the theorem. Let $\mathcal{T}$ denote the event that all **HasMajority** invocations (at Line 14 and Line 22 of Algorithm 1) are good. There can be at most $2N\lceil\log(4N')\rceil < 2N\lceil\log(\frac{16}{3}N)\rceil < 10N^2$ such invocations. By Lemma 13 and by union bound, $\mathcal{T}$ occurs with probability at least $1 - \frac{1}{N}$.

Let k be the first phase in Algorithm 1 such that $D' \geq D$. Such k must exist since by the end of all phases, $D' \geq 4N' > N \geq D$. Conditioned upon $\mathcal{T}$, Lemma 15 and Lemma 16 tell us that by the end of phase k, there will be exactly one node V executing Line 28 and set *leader_id* to V's id. Since there is no other node setting *leader_id* to its own id, V's id will be the only id that ever gets flooded in Line 31 in each phase. Next by Lemma 10, Line 31 in phase k will set *leader_id* to V's id on all nodes. This in turns causes all nodes to output V as the leader, if they have not yet done so. Hence the protocol is correct with probability at least $1 - \frac{1}{N}$.

For time complexity, conditioned upon $\mathcal{T}$, all nodes will output by the end of phase k and hence will output within total $O(D \log N)$ rounds. With the remaining probability of at most $\frac{1}{N}$, all nodes will output (albeit incorrectly) within $O(N \log N)$ rounds. Hence over expected coin flips, all nodes output within $O(D \log N)$ rounds, or $O(\log N)$ flooding rounds. □

C. TRIVIAL UPPER BOUNDS AND TRIVIAL REDUCTIONS

Trivial upper bounds when the diameter is known beforehand. If the diameter D of the dynamic network is known beforehand, the following describes some trivial and efficient protocols for CFlood, Consensus (in fact for simultaneous consensus as well), LeaderElect, Max, Hear-from-N-nodes, and estimating N.

For CFlood, recall that there is a special node V that needs to propagate a token. In the trivial protocol, a node that has seen this token will keep sending the token. Otherwise a node keeps receiving. Node V will claim that all nodes have received the token after exactly D rounds. For Consensus, each node has an initial value of 0 or 1. A node with initial value of 1 will keep sending 1. A node with initial value of 0 will keep receiving. Once it receives a 1, it will adopt 1 and keep sending 1. After exactly D rounds, each node outputs 1 if either its initial value is 1 or it has received 1. Otherwise the node outputs 0. For LeaderElect, invoking Algorithm 3 in Appendix B with $D' = D$ will enable all nodes to learn the largest id in the network in $O(\log N)$ flooding rounds. This largest id is then output as the leader's id. Max can be done in the same way by using the nodes' inputs as their ids. In the Hear-from-N-nodes problem [16], a node V can only terminate[6] in round r if for all nodes U in the network, $(U, 0) \rightsquigarrow (V, r)$. With D being known, the nodes do not need to do anything and can simply terminate after D rounds.

Finally, to estimate N, we go through successive phases. In the ith phase, each node chooses 1 with 2^{-i} probability, and 0 with the remaining probability. Next all the nodes flood the 1's, as in the previous Consensus protocol. The process stops when the nodes no longer see any 1 in a certain phase. It is easy to see that on expectation, it takes $O(\log N)$ phases (or $O(\log N)$ flooding rounds) for the process to stop. The nodes can then estimate N based on in which phase the process stops. For any constants $\epsilon > 0$ and $\delta > 0$, repeating this process constant number of times will be sufficient give us an estimate N' such that with probability at least $1 - \delta$, $|\frac{N'-N}{N}| \leq \epsilon$.

Trivial reduction from Consensus to LeaderElect. Given a LeaderElect oracle protocol $\mathcal{P}$, we construct a Consensus protocol $\mathcal{Q}$ in the following way: Each node appends its single bit initial value to its id, and then invokes $\mathcal{P}$ with its new id (which is one bit longer than its old id). When a node outputs the leader's id in $\mathcal{P}$, it decides on the last bit of the leader's id for $\mathcal{Q}$.

Trivial reduction from CFlood to Hear-from-N-nodes. Given an oracle $\frac{1}{3}$-error protocol $\mathcal{P}$ for Hear-from-N-nodes with time complexity of s flooding rounds, we will construct a $\frac{1}{3}$-error CFlood protocol $\mathcal{Q}$ with $O(s \log\log N)$ flooding rounds. To do so, we first define a modified protocol $\mathcal{P}'$ based on $\mathcal{P}$. In $\mathcal{P}'$, certain nodes may be *flagged* at the beginning of the protocol. A flagged node will keep sending a special flag message throughout the protocol. Any unflagged node receiving this message will become flagged itself (and will keep sending the flag message). Unflagged nodes in $\mathcal{P}'$ will run the protocol $\mathcal{P}$. This implies that if a node running $\mathcal{P}$ in a certain round happens to receive a flag message from some neighbor, then starting from the next round, the node will give up running $\mathcal{P}$ and will instead keep sending the flag message.

The protocol $\mathcal{Q}$ maintains a guess D' for D, where D' starts from 4 and keeps doubling. Recall that there is a special node V in CFlood that needs to propagate a token. For any given D', $\mathcal{Q}$ sequentially does the following steps:

1. All flagged nodes clear their flagged statuses. Next the system floods V's token for D' rounds.
2. All nodes that have not received the token become flagged. A flagged node will keep flooding a flag message. Any unflagged node receiving this message will become flagged itself (and will keep sending the flag message). Such process continues for exactly $10sD'$ rounds.
3. The nodes invokes protocol $\mathcal{P}'$ for $100 \log\log D'$ times, where each invocation continues execution for exactly $10sD'$ rounds. We say that an invocation of $\mathcal{P}'$ is *good* if at the end of those $10sD'$ rounds, V is not flagged and the protocol $\mathcal{P}$ on V has terminated.[7]
4. V claims that all nodes have received V's token if at least $\frac{2}{3} \cdot 100 \log\log D'$ invocations of $\mathcal{P}'$ are good.

The key point in the reduction is that for each D', if some node has not received V's token, then with probability at least $1 - \frac{1}{3\log^2 D'}$, V will not claim that all nodes have received V's token. The following provides some intuition behind this claim. If some node does not receive V's token, then immediately after Step 2, there will be at least $\min(10sD'+1, N)$ flagged nodes. If all N nodes are flagged, then V must be already flagged. Hence none of V's invocation of $\mathcal{P}'$ will be good. If there are $10sD'+1$ flagged nodes, let those nodes be V_1 through V_k. Given how $\mathcal{P}'$ is constructed, it is easy to show that in any round in $\mathcal{P}'$, for any node $U \neq V_i$ for all i where $1 \leq i \leq k$, if there are edges between U and some of the V_i's, then we can delete all those edges and add a new edge between U and V_1 without affecting the behavior of U and V_1 through V_k in $\mathcal{P}'$. After doing this for all such U's, we can arrange V_1 through V_k into a chain, again, without affecting behavior of the unflagged nodes in $\mathcal{P}'$.[8] Now under this modified dynamic network, if V is not flagged, one can show that V's behavior in $\mathcal{P}'$ (when everyone else runs $\mathcal{P}'$ as well) must be the same as if both V and all other nodes executed $\mathcal{P}$. Since it is impossible for $(V_k, 0) \rightsquigarrow (V, 10sD')$ to hold under this modified topology, with probability at most $\frac{1}{3}$, P can terminate on V within $10sD'$ rounds. Because we use $100 \log\log D'$ instances of $\mathcal{P}'$ and $\mathcal{Q}$ require $\frac{2}{3} \cdot 100 \log\log D'$ good instances, for $D' \geq 4$, the error probability is reduced to $\frac{1}{3\log^2 D'}$.

For the overall error probability of $\mathcal{Q}$, note that for each D', $\mathcal{Q}$ may terminate incorrectly with probability at most $\frac{1}{3\log^2 D'}$. Hence the error probability of $\mathcal{Q}$ is at most

[6]Termination in the Hear-from-N-nodes problem is defined as halting in [16], while in this paper, we usually define termination as outputting. None of our discussions here depend on this difference.

[7]Note that since V is not flagged, V will run $\mathcal{P}$.

[8]More rigorously, we are constructing a new dynamic network by modifying the topology in each round of the original dynamic network.

$\frac{1}{3}\sum_{i=2}^{\infty}\frac{1}{i^2} = \frac{1}{3}(\frac{\pi^2}{6} - 1) < \frac{1}{3}$. For time complexity, one can verify that once D' ($D' \geq 4$) reaches D, the protocol will terminate within $O(s \log \log N)$ flooding rounds on expectation. $\mathcal{Q}$'s total time complexity is $O(s \log \log N)$ flooding rounds as well — note that there is no extra $\Theta(\log D)$ factor since the overhead is smaller for smaller D' values.