The Cost of Unknown Diameter in Dynamic Networks[*]

[Extended Abstract] [†]

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. Here N is the number of nodes in the network. 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 of 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 has been a growing interest in *dynamic networks* [17], where the network topology may change over time. 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 protocols 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 an input, but internally implicitly use N as 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

[*]The first two authors of this paper are alphabetically ordered.
[†]A full version of this paper is available at `http://www.comp.nus.edu.sg/%7Eyuhf/TRA4-16.pdf`

SPAA '16, July 11 - 13, 2016, Pacific Grove, CA, USA
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4210-0/16/07...$15.00
DOI: http://dx.doi.org/10.1145/2935764.2935781

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 unknown diameter. Our results also carry over to the HEAR-FROM-N-NODES problem [16] (see the full version [21] of this paper), which in turn, reduces to computing globally-sensitive functions [16] such as MAX.

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 of 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 size $O(\log N)$ 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 $s(N)$ flooding rounds on all dynamic networks of size no more than N. When the context is clear, we also write $s(N)$ as s.

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 the full version [21] of this paper 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' of 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 of 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, the full version [21] of this paper 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, 22]:

- 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 DIS-

[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.

[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 type-Γ subnetwork
N'	estimate of N		A_Λ, B_Λ	special nodes in the type-Λ subnetwork
D	diameter of the dynamic network		A_Υ, B_Υ	special nodes in the type-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.

jointnessCP problem to *disagree* on the dynamic network that they simulate, which is quite different from reductions in other efforts [4, 6, 16, 20, 22].

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 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 reductions need 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 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 reductions in this paper are quite different from and are 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 of N makes some problems no longer sensitive to unknown diameter.

2. MODEL AND DEFINITIONS

Dynamic network. 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 $(U, r) \rightarrow (V, r+1)$ 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 "$\rightarrow$": 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

[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.

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 network $\mathcal{G}$, 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 $q \geq 3$, $\text{DisjointnessCP}_{n,q}$ [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 $\text{DisjointnessCP}(\mathbf{x}, \mathbf{y})$, 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 $\text{DisjointnessCP}_{n,q}$ 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 $\text{DisjointnessCP}_{n,q}$ more efficiently. The following lower bound is from [4]:

Theorem 1. (From [4].) *A $\frac{1}{5}$-error public coin Monte Carlo protocol for $\text{DisjointnessCP}_{n,q}$ 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 the full version [21] of this paper for the simple proof):

Corollary 2. *For any $\frac{1}{6}$-error public coin Monte Carlo protocol for $\text{DisjointnessCP}_{n,q}$, there exist $\mathbf{x}_0$ and $\mathbf{y}_0$ such that $\text{DisjointnessCP}_{n,q}(\mathbf{x}_0, \mathbf{y}_0) = 1$ 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 the full version [21] of this paper), 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 $\text{DisjointnessCP}(\mathbf{x}, \mathbf{y})$ 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 $\text{DisjointnessCP}(\mathbf{x}, \mathbf{y}) = 1$, and $\Omega(q)$ diameter if $\text{DisjointnessCP}(\mathbf{x}, \mathbf{y}) = 0$. Assume that the CFlood oracle protocol promises to terminate within s flooding rounds. We can then show that i) if $\text{DisjointnessCP}(\mathbf{x}, \mathbf{y}) = 1$, then the CFlood protocol under the corresponding network should terminate within $O(s)$ rounds, and ii) if $\text{DisjointnessCP}(\mathbf{x}, \mathbf{y}) = 0$, 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 $\text{DisjointnessCP}(\mathbf{x}, \mathbf{y}) = 1$. Otherwise they claim $\text{DisjointnessCP}(\mathbf{x}, \mathbf{y}) = 0$. In such a way, Alice and Bob successfully solve 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 need 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 us 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, each subnetwork has $\Theta(nq)$ nodes, and by itself does not need to be connected in each round. Section 4 and Section 5 will the present the 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 Bounds 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 the full version [21] of this paper). 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).

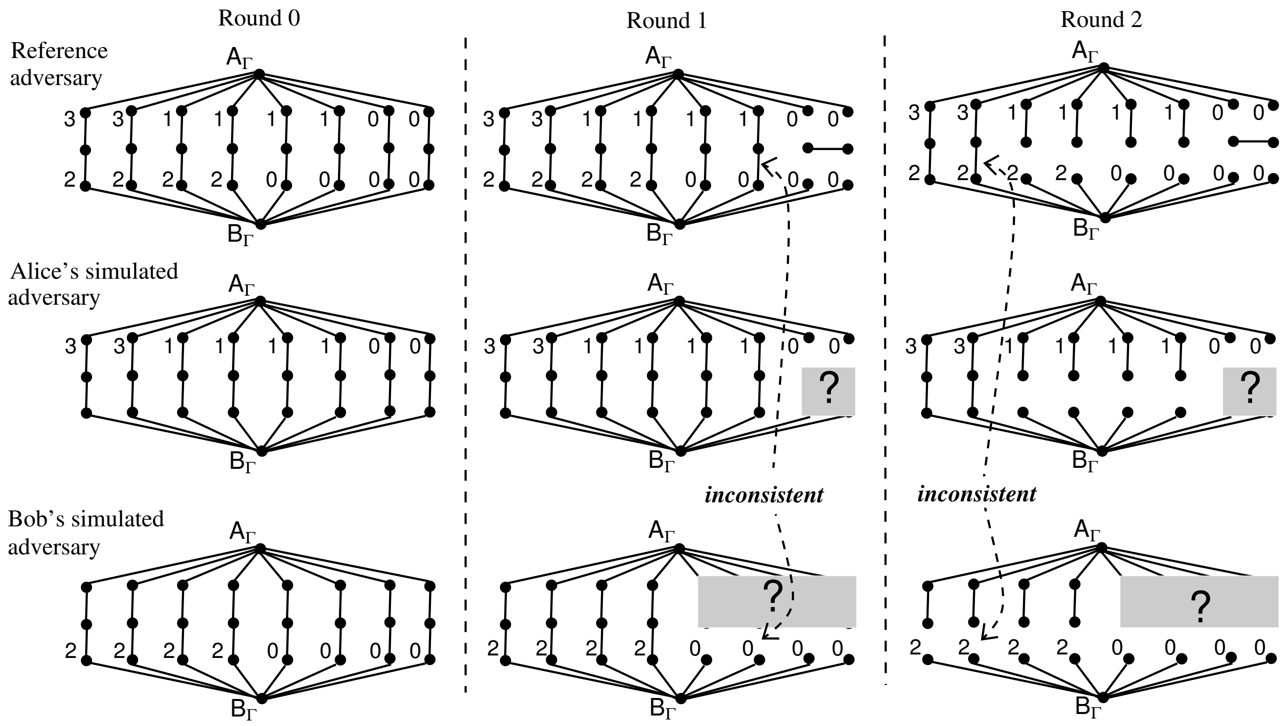

Figure 1: The three adversaries of the type-Γ subnetwork, for $n = 4$, $q = 5$, $\mathbf{x} = 3110$, and $\mathbf{y} = 2200$. The numbers beside the nodes are labels. This example assumes that the middle nodes on all the chains are receiving.

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 middle nodes into a line. Obviously, we will have at least $\frac{q-1}{2}$ chains in the form of $|_0^0$ if $\textsc{Disjointness}\text{CP}(\mathbf{x}, \mathbf{y}) = 0$. 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 every 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 even though 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 enable 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 ($r \leq \frac{q-1}{2}$), 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* for 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 claims that for any node Z that is non-spoiled for Alice in a round and is receiving in that round, Alice is able to determine which nodes can potentially send messages to Z in that round under the reference adversary, by simply simulating her own adversary based on $\mathbf{x}$. The lemma further claims that any node that is sending messages to Z must be either non-spoiled for Alice or the node B_Γ. All these 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 $1 \leq r \leq \frac{q-1}{2}$ 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 the node 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 $|_{2t+1}^{2t}$ chain, U is always non-spoiled, and V and W are non-spoiled iff $r \leq t$:

- For node U, we exhaustively enumerate all scenarios: i) If $r \leq t$, then $\mathcal{S} = \mathcal{S}' = \{A_\Gamma, V\}$. By definition, both A_Γ and V are non-spoiled in round $r-1$. ii) If $r \geq t+2$, 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 $r \leq t$, we have $\mathcal{S} = \mathcal{S}' = \{U, W\}$, and both nodes are non-spoiled in round $r-1$.
- For node W and $r \leq t$, 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 is always non-spoiled, and V and W are non-spoiled iff $r \leq t$:

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

- 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 $r \leq t$, we have $\mathcal{S} = \mathcal{S}' = \{U, W\}$, and both nodes are non-spoiled in round $r-1$. ii) If $r \geq t+2$,

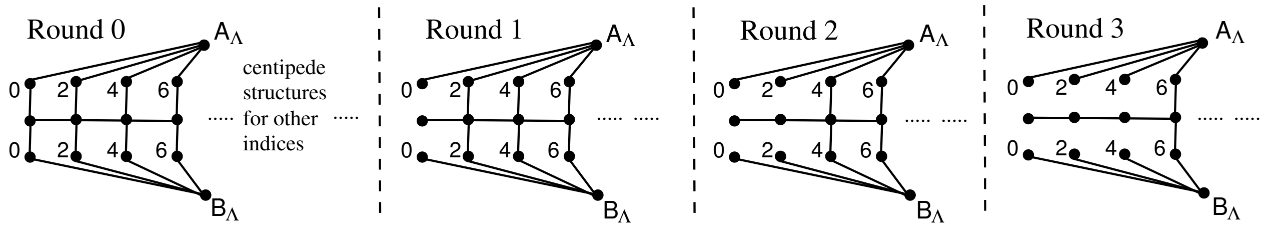

Figure 2: The ith centipede structure in the type-Λ subnetwork under the reference adversary, for $x_i = y_i = 0$ and $q = 7$.

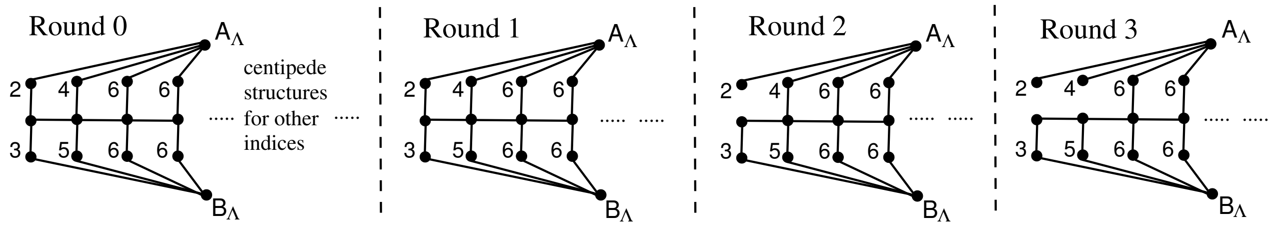

Figure 3: The ith centipede structure in the type-Λ subnetwork under the reference adversary, for $x_i = 2$, $y_i = 3$, and $q = 7$, assuming all middle nodes on all chains are sending.

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 $|_{2t}^{2t-1}$ 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. □

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 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 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 have 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 the full version [21] of this paper) 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 (called *bridging edges*) in the bridging edge set $\mathcal{E}$ 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. 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 determine 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 to 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. □

Lower bounds for CFlood **and** Consensus**.** Using Lemma 5 and following the intuition described in Section 3, we can eventually obtain the following two main theorems.

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.*

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 that 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 know 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 messages 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 $\textsc{DisjointnessCP}_{n,q}(\mathbf{x},\mathbf{y}) = 1$. 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 $\frac{5}{6}$, making the above a valid reduction. If $\textsc{DisjointnessCP}(\mathbf{x},\mathbf{y}) = 1$, 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 $\textsc{DisjointnessCP}(\mathbf{x},\mathbf{y}) = 1$.

  If $\textsc{DisjointnessCP}(\mathbf{x},\mathbf{y}) = 0$, 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 $\textsc{DisjointnessCP}_{n,q}$ 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) $\textsc{DisjointnessCP}_{n,q}(\mathbf{x}_0,\mathbf{y}_0) = 1$, 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. *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. See the full version [21] of this paper. □

7. UPPER BOUND FOR Consensus AND 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 the full version [21] of this paper), 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 the full version [21] of this paper. 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 some 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 receives 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 finds that

[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.

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 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 the full version [21] of this paper:

THEOREM 8. *Let $c \in (0, \frac{1}{3}]$ be any given constant and N' be any value such that $|\frac{N'-N}{N}| \le \frac{1}{3} - c$. 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 their 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] H. Yu, Y. Zhao, and I. Jahja. The Cost of Unknown Diameter in Dynamic Networks. Technical Report TRA4/16, School of Computing, National University of Singapore, April 2016. Also available at `http://www.comp.nus.edu.sg/%7Eyuhf/TRA4-16.pdf`.

[22] 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.