The Cost of Fault Tolerance in Multi-Party Communication Complexity

BINBIN CHEN, Advanced Digital Sciences Center
HAIFENG YU and YUDA ZHAO, National University of Singapore
PHILLIP B. GIBBONS, Intel Labs Pittsburgh

Multi-party communication complexity involves distributed computation of a function over inputs held by multiple distributed players. A key focus of distributed computing research, since the very beginning, has been to tolerate failures. It is thus natural to ask "If we want to compute a certain function in a fault-tolerant way, what will the communication complexity be?" For this question, this article will focus specifically on (i) tolerating node crash failures, and (ii) computing the function over general topologies (instead of, e.g., just cliques).

One way to approach this question is to first develop results in a simpler failure-free setting, and then "amend" the results to take into account failures' impact. Whether this approach is effective largely depends on how big a difference failures can make. This article proves that the impact of failures is significant, at least for the SUM aggregate function in general topologies: As our central contribution, we prove that there exists (at least) an exponential gap between the non-fault-tolerant and fault-tolerant communication complexity of SUM. This gap attests that fault-tolerant communication complexity needs to be studied separately from non-fault-tolerant communication complexity, instead of being considered as an "amended" version of the latter. Such exponential gap is not obvious: For some other functions such as the MAX aggregate function, the gap is only logarithmic.

Part of our results are obtained via a novel reduction from a new two-party problem UNIONSIZECP that we introduce. UNIONSIZECP comes with a novel *cycle promise*, which is the key enabler of our reduction. We further prove that this cycle promise and UNIONSIZECP likely play a fundamental role in reasoning about fault-tolerant communication complexity.

19

Categories and Subject Descriptors: F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms: Theory, Algorithms

Additional Key Words and Phrases: Aggregate functions, communication complexity, fault tolerance, promise problems

ACM Reference Format:
Binbin Chen, Haifeng Yu, Yuda Zhao, and Phillip B. Gibbons. 2014. The cost of fault tolerance in multi-party communication complexity. J. ACM 61, 3, Article 19 (May 2014), 64 pages.
DOI: http://dx.doi.org/10.1145/2597633

A preliminary conference version of this article [Chen et al. 2012] appeared in *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)* 2012.
The first three authors of this article are alphabetically ordered.
This work was partly supported by the research grant for the Human Sixth Sense Programme at the Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research (A*STAR), partly supported by Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2011-T2-2-042, and partly supported by the Intel Science and Technology Center for Cloud Computing (ISTC-CC).
Authors' addresses: B. Chen, Advanced Digital Sciences Center, Republic of Singapore; email: binbin.chen@adsc.com.sg; H. Yu and Y. Zhao, School of Computing, National University of Singapore, Republic of Singapore; email: {haifeng, zhaoyuda}@comp.nus.edu.sg; P. B. Gibbons, Intel Labs, Pittsburgh, PA; email: phillip.b.gibbons@intel.com.

DOI: http://dx.doi.org/10.1145/2597633

1. INTRODUCTION

1.1. Background and Motivation

Fault Tolerance in Communication Complexity and Our Focus. Multi-party communication complexity [Chandra et al. 1983] involves distributed computation of a function over inputs held by multiple distributed players. A key focus of distributed computing research, since the very beginning, has been to tolerate failures. It is thus natural to ask "If we want to compute a certain function in a fault-tolerant way, what will the communication complexity be?" Here the computation is allowed to ignore/omit the inputs held by those players that have failed. For this question, this paper will focus specifically on (i) tolerating node crash failures, and (ii) computing the function over general topologies (instead of for example, just cliques). Our focus is driven by the following key practical setting, which is also driving many other distributed computing research efforts today.

Practical Motivations. Our focus here on fault-tolerant communication complexity is motivated by emerging large-scale wireless sensor networks and large-scale wireless ad hoc networks—the rapid advance of hardware technologies in the past decade or so have made their large-scale deployment a reality. Such networks consist of many low-cost nodes (sensors or wireless routers) distributed over a large physical area. Due to the limited wireless transmission range of these nodes, only nearby nodes can directly communicate with each other. This results in a multi-hop network topology that is often beyond the control of the protocol designer. For example, wireless sensor networks may be deployed simply by airplanes dropping sensors onto a target region [McGlynn and Borbash 2001], or deployed according to the specific physical environment that they monitor. The physical nature of these networks thus naturally requires one to consider general topologies.

Distributed computation of functions is of fundamental importance in these wireless sensor networks and wireless ad hoc networks. For example, consider a sensor network for temperature monitoring in a forest. In such a setting, the temperature reading of a single sensor often bears limited importance. Instead, we often need aggregate information such as the average temperature in a certain region [Madden et al. 2002]. This then corresponds to the computation of certain aggregate functions [Gray et al. 1997; Madden et al. 2002] over the sensor readings.

Communication complexity also has significant practical relevance in this setting since (i) wireless communication usually consumes far more energy than local computation, and needs to be minimized for nodes operating on battery power or nodes relying on energy harvesting, and (ii) the (bandwidth) capacity of these wireless networks does not scale well [Gupta and Kumar 2000].

Finally, as in most large-scale distributed systems, failures are the norm instead of exception in these sensor networks and wireless ad hoc networks. In fact, this is particularly so because the nodes are often exposed to harsh physical environments.

Other Related Focuses on Fault-Tolerant Communication Complexity. Our focus on fault-tolerant communication complexity (i.e., tolerating crash failures in general topologies) has not been thoroughly studied by previous researchers. Related to our focus, there have been prior work on the communication complexity of secure multi-party computation [Beaver 1991; Beerliová-Trubíniová and Hirt 2006, 2008; Ben-Or et al. 1988, 1993; Chaum et al. 1988a, 1988b; Franklin and Yung 1992; Galil et al. 1987; Goldreich et al. 1987; Hirt et al. 2000; Hirt and Nielsen 2006; Rabin and Ben-Or 1989; Yao 1982, 1986] and fault-tolerant distributed consensus [Berman and Garay 1993; Chlebus et al. 2009; Gilbert and Kowalski 2010; King et al. 2006a, 2006b, 2010; King and Saia 2009, 2010; Pietro and Michiardi 2008]. While they do consider failures, their main focuses are on separate challenges such as privacy requirements or dealing with byzantine failures, which are different from our focus. Given such different emphases, most of them assume that the players form a clique. Section 2 provides a more detailed discussion on related work.

1.2. Our Exponential Gap Result

We have so far motivated our specific focus (i.e., tolerating crash failures in general topologies) on fault-tolerant communication complexity. One way to approach the subject is to first develop results in a simpler failure-free setting, and then "amend" the results to take into account failures' impact. Whether this approach is effective largely depends on how big a difference failures can make. This paper proves that the impact of failures is significant, at least for the SUM aggregate function. Here the SUM function is defined over a synchronous wireless network with N nodes and some arbitrary topology. Each node has a binary value, and the goal is simply to determine the sum of all the values. The function is allowed to freely ignore/omit the inputs held by failed or disconnected nodes. (See Section 3 for a more formal description.) Note that SUM can be easily reduced to and from many other interesting aggregate functions [Gray et al. 1997] such as SELECTION.[1]

As the central contribution of this work, we prove that there exists (at least) an exponential gap between the non-fault-tolerant (NFT) and fault-tolerant (FT) communication complexity of SUM. Here FT communication complexity is the smallest communication complexity among all fault-tolerant protocols that can tolerate an arbitrary number of failures, while NFT communication complexity corresponds to all protocols. To our knowledge, ours is the first such gap result between FT communication complexity and NFT communication complexity over general topologies. This exponential gap attests that FT communication complexity needs to be studied separately from NFT communication complexity, instead of being considered as an "amended" version of NFT communication complexity.

It is worth noting that such exponential gap is far from obvious: For some other functions such as the MAX function, there is no large gap—the gap is only logarithmic. See Appendix A for more details.

Existing Results on SUM*.* In a failure-free setting, by leveraging in-network processing, a trivial tree-aggregation protocol (see Section 4) can compute SUM with zero-error while requiring each node to send $O(\log N)$ bits. Since we consider general network topologies, we will naturally define communication complexity of a protocol as the number of bits sent by the bottleneck node (see Section 3 for formal discussion), instead of by all nodes combined. Hence, for zero-error results, the NFT communication complexity of SUM is upper bounded by $O(\log N)$. For (ϵ, δ)-approximate results, where with probability at least $1 - \delta$ the result has at most ϵ relative error, it is possible to further reduce the communication complexity to $O(\log \frac{1}{\epsilon} + \log\log N)$ bits per node for constant δ (see Section 4).

In comparison, to tolerate arbitrary failures, we are not aware of any zero-error SUM protocol that is better than trivially having every node flood its id together with its

[1]The SELECTION function returns the xth largest value among the N values, where x is an input parameter. Here each value is an integer whose domain is polynomial with N. One can reduce SUM to SELECTION by finding i, via a binary search over $[1, N]$, such that the ith largest value is 1 while the $i + 1$th largest value is 0. Hence, by solving a logarithmic number of SELECTION instances, we can solve SUM. On the other hand, one can also reduce SELECTION to SUM by doing a binary search over the value domain. At each step of the binary search, we count the number of nodes whose value is larger than the value we are examining. The process stops when the count is exactly x. Here again, by solving a logarithmic number of SUM instances, we can solve SELECTION.

value and thus requiring each node to send $O(N \log N)$ bits. For (ϵ, δ)-approximate results, researchers have proposed some protocols [Bawa et al. 2007; Considine et al. 2004; Mosk-Aoyama and Shah 2006; Nath et al. 2008; Yu 2011] where each node needs to send roughly $O(\frac{1}{\epsilon^2})$ bits for constant δ (after omitting logarithmic terms of $\frac{1}{\epsilon}$ and N). All these protocols conceptually map the value of each node to geometrically weighted positions in some bit vectors, and then estimate the sum from the bit vectors. Same as in one-pass distinct element counting algorithms in streaming databases [Alon et al. 1996; Flajolet and Martin 1985], doing so makes the whole process duplicate-insensitive. In turn, this allows each node to push its value along multiple directions to guard against failures. Note however, that duplicate-insensitive techniques do not need to be one-pass, and furthermore tolerating failures does not have to use duplicate-insensitive techniques. For example, one could repeatedly invoke the tree-aggregation protocol until one happens to have a failure-free run. There is also a large body of work [Aysal et al. 2009; Boyd et al. 2006; Chen et al. 2005; Chen and Pandurangan 2010; Jelasity et al. 2005; Kashyap et al. 2006; Kempe et al. 2003] on computing SUM via gossip-based averaging (also called average consensus protocols). They all rely on the mass conservation property [Kempe et al. 2003], and thus are vulnerable to node failures. There have been a few efforts [Eyal et al. 2011; Jesus et al. 2009] on making these protocols fault-tolerant. But they largely focus on correctness, without formal results on the protocol's communication complexity in the presence of failures. Despite all these efforts, no lower bounds on the FT communication complexity of SUM have ever been obtained, and thus it has been unknown whether the existing protocols can be improved.

Our Results. Our main results in this paper are the first lower bounds on the FT communication complexity (or *FT lower bounds* in short) of SUM, for public-coin randomized protocols with zero-error and with (ϵ, δ)-error. These FT lower bounds are (at least) exponentially larger than the corresponding upper bounds on the NFT communication complexity (or *NFT upper bounds* in short) of SUM, thus establishing an exponential gap. Private-coin protocols and deterministic protocols are also fully but implicitly covered, and our exponential gap still applies.

Specifically, since there is a tradeoff between communication complexity and time complexity, Figure 1 summarizes our FT lower bounds when the time complexity of the SUM protocol is within b *aggregation rounds*, for b from 1 to ∞. The notion of aggregation rounds is introduced to isolate the effects of the topology on the time complexity, and will be formally defined later in Section 3. Intuitively, each aggregation round consists of x rounds, where the value x captures factors such as the diameter of the topology. For $b \le N^{0.25-c}$ or $b \le \frac{1}{\epsilon^{0.5-c}}$ where c is any positive constant below 0.25, the NFT upper bounds are always at most logarithmic with respect to N or $\frac{1}{\epsilon}$, while the FT lower bounds are always polynomial.[2] For $b > N^{0.25-c}$ or $b > \frac{1}{\epsilon^{0.5-c}}$, the NFT upper bounds drop to $O(1)$, while the FT lower bounds are still at least logarithmic. Our results also imply that under small b values (i.e., $b \le 2 - c$), the existing fault-tolerant SUM protocols [Bawa et al. 2007; Considine et al. 2004; Mosk-Aoyama and Shah 2006; Nath et al. 2008; Yu 2011] (incurring $O(N \log N)$ or $O(\frac{1}{\epsilon^2})$ bits per node) are actually optimal within polylog factors.[3]

[2]Here for (ϵ, δ)-approximate results, we only considered terms containing ϵ. Even if we take the extra terms with N into account, our exponential gaps continue to exist as long as $\frac{1}{\epsilon^c} = \Omega(\log N)$.
[3]Note that prior efforts analyzed these protocols in a model where (i) wireless network communication collisions are ignored, and (ii) the failure adversary (for determining which nodes fail at what time) is oblivious. The optimality hence refers to the optimality under that model. This article actually uses a

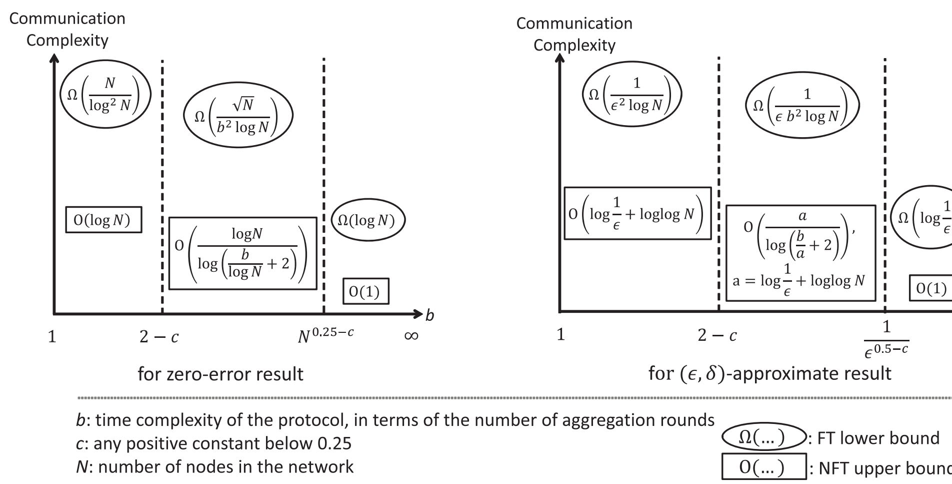

Fig. 1. Summary of our exponential gaps. All NFT upper bounds are either well-known or are obtained via standard tricks—they are described in Section 4. All FT lower bounds are novel and are our main contributions. They are obtained in Section 5 (for $1 \leq b \leq 2-c$), Section 6 (for $2-c < b \leq N^{0.25-c}$ or $2-c < b \leq \frac{1}{\epsilon^{0.5-c}}$), and Section 8 (for $b > N^{0.25-c}$ or $b > \frac{1}{\epsilon^{0.5-c}}$).

Our Approach. Our FT lower bounds for $b \leq 2-c$ are obtained via a simple but interesting reduction from a two-party communication complexity problem UnionSize, where Alice and Bob intend to determine the size of the union of two sets. In the reduction, without knowing Bob's input, Alice can only simulate the Sum oracle protocol's execution in part of the network. Furthermore, this part is continuously shrinking due to the spreading of such unknown information. Failures play a fundamental role in the reduction—they hinder the spreading of unknown information. The FT lower bounds under $b \leq N^{0.25-c}$ or $b \leq \frac{1}{\epsilon^{0.5-c}}$ are much harder to obtain. There we introduce a new two-party problem called UnionSizeCP, which is roughly UnionSize extended with a novel *cycle promise*. Identifying this promise is a key contribution of this work, which enables the continuous injection of failures to further hinder the spreading of unknown information. We then prove a lower bound on UnionSizeCP's communication complexity via information cost [Bar-Yossef et al. 2004]. This lower bound, coupled with our reduction, leads to FT lower bounds for Sum. We further prove a strong completeness result showing that UnionSizeCP is *complete* among the set of all two-party problems that can be reduced to Sum in the FT setting via *oblivious reductions* (defined in Section 7). Namely, we prove that every problem in that set can be reduced to UnionSizeCP. Our proof also implicitly derives the cycle promise, thus showing that it likely plays a fundamental role in reasoning about the FT communication complexity of many functions beyond Sum. Finally, our FT lower bounds under $b > N^{0.25-c}$ or $b > \frac{1}{\epsilon^{0.5-c}}$ are obtained by drawing a connection to an interesting probing game, and then proving a lower bound on the probing game.

1.3. Roadmap

Section 2 discusses related work. Section 3 defines the Sum problem and various complexity measures. Section 4 presents the upper bounds on the NFT communication

different model where (i) collision is considered, and (ii) the failure adversary is adaptive. However, our lower bounds for $b \leq 2-c$ continues to hold under their model. (In fact, all our lower bounds hold under their model except the lower bounds for $b > N^{0.25-c}$ or $b > \frac{1}{\epsilon^{0.5-c}}$, which require an adaptive adversary.)

complexity of SUM. Section 5 proves the lower bounds on the FT communication complexity of SUM for $b \leq 2 - c$, while Section 6 proves the corresponding lower bounds for $b \leq N^{0.25-c}$ and $b \leq \frac{1}{\epsilon^{0.5-c}}$. Next, Section 7 proves the completeness result for UNION-SIZECP, showing that the polynomial dependency on b in Section 6's lower bounds might be inherent in Section 6's overall approach. Section 8 proves the lower bounds on the FT communication complexity of SUM for all b. Finally, Section 9 discusses various extensions of our results, and Section 10 draws conclusions and proposes future work.

2. RELATED WORK

Section 1 already thoroughly discussed previous efforts on the communication complexity of computing SUM. In the following, we will discuss several prior efforts that are related to our work in the broader sense.

Secure Multi-Party Computation. Our fault-tolerant communication complexity is related to the topic of secure multi-party computation [Beaver 1991; Beerliová-Trubíniová and Hirt 2006, 2008; Ben-Or et al. 1988, 1993; Chaum et al. 1988a, 1988b; Franklin and Yung 1992; Galil et al. 1987; Goldreich et al. 1987; Hirt et al. 2000; Hirt and Nielsen 2006; Rabin and Ben-Or 1989; Yao 1982, 1986]. Secure multi-party computation also aims to compute a function whose inputs are held by multiple distributed players. Different from our work, secure multi-party computation mainly focuses on the privacy requirement. Namely, when computing the function, a player should not learn any information about the inputs held by other players, except what can already be inferred from the output of the function. Research on secure multi-party computation usually investigates whether it is possible to compute a certain class of functions, and if yes, what is the communication complexity. The failure model considered by secure multi-party computation, given the security nature of the subject, is more diverse than our simple crash failure model. For example, researchers have considered players that (i) are *curious* but follow the protocol [Ben-Or et al. 1988; Franklin and Yung 1992; Galil et al. 1987; Goldreich et al. 1987; Yao 1982, 1986], (ii) may crash [Ben-Or et al. 1993; Franklin and Yung 1992], or (iii) may experience byzantine failures [Beaver 1991; Beerliová-Trubíniová and Hirt 2006, 2008; Ben-Or et al. 1988, 1993; Chaum et al. 1988a, 1988b; Franklin and Yung 1992; Galil et al. 1987; Goldreich et al. 1987; Hirt et al. 2000; Hirt and Nielsen 2006; Rabin and Ben-Or 1989]. In terms of the topology among the players, to the best of our knowledge, research on secure multi-party computation almost always assumes that the players are fully connected and form a clique.

The central difference between our SUM problem and secure multi-party computation is that the latter's key challenge is to preserve privacy. If privacy is not a concern, then secure multi-party computation problems usually become trivial (i.e., with trivial and matching upper/lower bounds). In comparison, our SUM problem is not concerned with privacy—the key challenge instead is to compute SUM over general topologies (rather than just cliques). If we only consider cliques, then SUM becomes trivial (i.e., with trivial and matching upper/lower bounds).

Such central difference between the two problems implies that they are incomparable—neither of them is easier than the other. Furthermore, upper bounds, lower bounds, and proof techniques for one problem usually cannot carry over to the other. For example, the lower bounds in secure multi-party computation are usually derived from the privacy requirement, while we prove lower bounds on SUM by constructing proper lower bound topologies (i.e., worst-case topologies).

Communication Complexity under Unreliable Channels. Other than in the topic of secure multi-party computation, tolerating node failures has not been considered in various developments on different models for communication complexity (e.g., Braverman and Rao [2011], Chandra et al. [1983], Impagliazzo and Williams [2010], Rajagopalan and Schulman [1994], and Schulman [1996]). Among these developments, the closest setting to tolerating node failures is perhaps unreliable channels [Braverman and Rao 2011; Gargano and Rescigno 1993; Rajagopalan and Schulman 1994; Schulman 1996]. For example, the channels may flip the bits adversarially, flip each bit independently with the same probability (i.e., a *binary symmetric channel*), or drop a certain number of messages. Under the binary symmetric channel model, there have also been some information-theoretic lower bounds on the rates of distributed computations [Ayaso et al. 2010; Giridhar and Kumar 2006]. The specific techniques and insights for unreliable channels have limited applicability to tolerating node failures.

Bit Complexity of Other Distributed Computing Tasks in Failure-Prone Settings. Related to the computation of functions, distributed computing researchers have also studied the communication complexity (usually called *bit complexity* here) of other distributed computing tasks in failure-prone settings. For example, there has been a large body of work [Chlebus et al. 2009; Gilbert and Kowalski 2010; King et al. 2006a, 2010; King and Saia 2009, 2010; Pietro and Michiardi 2008] on the bit complexity of distributed consensus and leader election. Compared to our work, all these efforts assume that the players are fully connected and form a clique. As explained earlier, for our SUM problem, the key challenge is exactly to do the computation over general topologies instead of just cliques. On the other hand, these problems have their own unique challenges such as tolerating byzantine failures (instead of just tolerating crash failures as in SUM). Because of this, again, distributed consensus/leader election and SUM are incomparable—neither of them is easier than the other.

Some researchers feel that cliques may not be "realistic" topologies in some cases. Hence, they explicitly construct low-degree network topologies, and then propose novel distributed consensus and leader election protocols specifically for those topologies [Berman and Garay 1993; King et al. 2006b]. In some sense, the performance of these protocols are defined over the *best-case* topology that is low-degree. This corresponds to a setting where the topology is within the control of the protocol designer, and then a protocol is designed specifically for that topology. In comparison, as motivated in Section 1, our SUM problem considers general topologies where the performance (i.e., time complexity and communication complexity) of a SUM protocol is defined over the *worst-case* topology.

3. MODEL AND DEFINITIONS

This section describes the system model and formal definitions used throughout this article, except in Section 9. For clarity, we defer to Section 9 various relaxed/extended versions of the system model and definitions, under which our exponential gap results continue to hold. Table I summarizes the key notations. All "log" in this article means $\log_2$.

System Model. We consider a wireless network with N *nodes* and an arbitrary undirected and connected graph G as the network topology. Each node has a unique id. We assume that the topology G (including the ids of each of the N nodes in G) is known to all nodes. The system is synchronous and a protocol proceeds in synchronous rounds. All nodes simultaneously start executing the protocol in round 1, with no activation needed. In each round, a node (which has not failed) first performs some local computation, and then does either a *send* or *receive* operation (but not both). We also say that the node is in a *sending state* or a *receiving state* in that round, respectively. By doing a send, a node (locally) broadcasts one message to all its neighbors in G. For

Table I. Key Notations in This Article

G	the undirected and connected graph representing the network topology
N	number of nodes (i.e., vertices) in G
$\Lambda(G)$	number of rounds in an aggregation round in G
λ	$\max_{G' \in \mathcal{G}} \Lambda(G')$ where $\mathcal{G}$ is the set of topologies appearing during some given execution
n (except in Section 8)	the size of a two-party problem
n (in Section 8)	a basic parameter in the lower bound construction
ϵ and δ	as in (ϵ, δ)-approximate result
b	time complexity of a SUM protocol, in terms of the number of aggregation rounds
t	time complexity of a two-party problem, in terms of (synchronous) rounds
c	positive constant whose range depends on the context
NFT_0	communication complexity for zero-error results
$\text{NFT}_{\epsilon,\delta}$	communication complexity for (ϵ, δ)-*approximate result*
FT_0	same as NFT_0 except that fault-tolerance is required
$\text{FT}_{\epsilon,\delta}$	same as $\text{NFT}_{\epsilon,\delta}$ except that fault-tolerance is required

our upper bounds, we require that the size of the message is $O(\log N)$. For our lower bounds, we do not restrict the size of the message. Such treatment serves to make our results stronger. Our results are insensitive to whether wireless network communication collisions are possible, but to make everything concrete, we still adopt and stick to the following commonly used collision model. By doing a receive, the node receives the message sent by one of its neighbors j iff node j is the only sending node among all node i's neighbors. If multiple neighbors of i send in the same round, a collision occurs and node i does not receive anything. All our results hold regardless of whether node i can distinguish silence from collision.

Failure Model. Any nodes in G may experience crash failures (but not byzantine failure), and the total number of failures can be up to $N - 1$. (See Section 9 for more discussion on the number of failures.) To model worst-case behavior, we have an adversary determine which nodes fail at what time. The adversary can be adaptive to the behavior of the protocol (including the coin flips) so far, but it cannot predict future coin flip results.

Recall that we eventually aim to prove lower bounds on the fault-tolerant communication complexity of SUM, and also to show some (trivial) upper bounds on the non-fault-tolerant communication complexity of SUM. For the upper bounds, the failure model is irrelevant since we will be considering a failure-free setting. For proving the lower bounds, we will allow the existence of a well-known *root* node in the topology that never fails. Having such a special root node only makes it easier to compute SUM, and hence makes our lower bound *stronger*. In fact, proving our lower bound while allowing such a root will shed additional light onto the problem: Even having such a root does not remove the exponential gap. As a side benefit, having such a root node also helps to simplify our discussions later. Hence, we will use such a concept of a root node in the rest of the article, though one should keep in mind that none of our results rely on the fact that the root does not fail.

The SUM *Problem.* Here, each node i in G has a binary value w_i, which is initially unknown to any other node. Let $s_2 = \sum_{i=1}^{N} w_i$, and let s_1 be the sum of w_j's where by the end of the protocol's execution, node j has not failed or been disconnected from the root due to other nodes' failures. Following the same definitions from Bawa et al. [2007], a *zero-error result* of SUM is any s where $s_1 \leq s \leq s_2$, and an (ϵ, δ)-*approximate result* of SUM is any $\hat{s}$ such that for some zero-error result s, $\Pr[|\hat{s} - s| \geq \epsilon s] \leq \delta$.

Time Complexity of SUM *Protocols.* We will consider only public-coin randomized protocols. By default, a "randomized protocol" in this article is a public-coin randomized protocol. For a randomized SUM protocol and with respect to a topology G, we define the protocol's *time complexity under* G to be the number of rounds needed for the protocol to *terminate*, under the worst-case values of the nodes in G, the worst-case failures (for fault-tolerant cases), and the worst-case random coin flips in the protocol. The protocol *terminates* when the root outputs the SUM result. The topology G has a large impact on time complexity, and we use the notion of *aggregation rounds* to isolate such impact. We will describe the time complexity in terms of aggregation rounds. This is analogous to describing it as a multiple of, for example, the diameter of G.

In failure-free settings, an *aggregation round* in G consists of $\Lambda(G)$ rounds, where $\Lambda(G)$ is a function of the connected graph G. We will define $\Lambda(G)$ precisely later in Section 4, which describes a simple deterministic tree-aggregation protocol and then defines $\Lambda(G)$ as the number of rounds needed for that protocol to finish on G. When failures are possible, the network topology may change during execution. Let $\mathcal{G}$ be the set of all topologies that have ever appeared during the given execution. Note that a $G' \in \mathcal{G}$ may or may not be connected. For any such G' that is not connected, we define $\Lambda(G')$ to be $\Lambda(G'')$ where G'' is the connected component of G' that contains the root. To allow a fair comparison between NFT and FT communication complexity, we define an *aggregation round* in an execution with failures to be $\max_{G' \in \mathcal{G}} \Lambda(G')$ rounds. Since $G \in \mathcal{G}$ and thus $\max_{G' \in \mathcal{G}} \Lambda(G') \geq \Lambda(G)$, an aggregation round for an FT protocol is either the same or longer than that for an NFT protocol, making our gap results stronger.

NFT and FT Communication Complexity of SUM *Protocols.* Classic multi-party communication complexity problems [Kushilevitz and Nisan 1996] usually consider the total number of bits sent by all players, since they usually use the whiteboard model where the whiteboard is the bottleneck. In our distributed computing setting with a topology G, as in other problems in such a setting, it is more natural to consider the number of bits sent by the bottleneck player. Given a randomized SUM protocol, a topology G, a value assignment to the nodes in G, and a failure adversary (if failures are considered), define a_i to be the *expected* (with the expectation taken over coin flips in the protocol) number of bits that node i sends. The protocol's *average-case communication complexity under* G is defined as the largest a_i, across all value assignments of the nodes in G, all failure adversaries (if failures are considered), and all i's ($1 \leq i \leq N$). The protocol's *worst-case communication complexity under* G is similarly defined by considering worst-case coin flips instead of taking the expectation over the coin flips.

We define $\text{NFT}_0(\text{SUM}_G, b)$ to be the smallest average-case communication complexity under G across all randomized SUM protocols that can generate, in a failure-free setting, a zero-error result on G within a time complexity of at most b aggregation rounds.[4] Similarly, we define $\text{NFT}_{\epsilon,\delta}(\text{SUM}_G, b)$ to be the smallest worst-case communication complexity under G across all randomized SUM protocols that can generate, in a failure-free setting, an (ϵ, δ)-approximate result on G within a time complexity of at most b aggregation rounds. Here, note that (i) the length of an aggregation round depends on G, and (ii) using the worst-case communication complexity for defining $\text{NFT}_{\epsilon,\delta}$ is standard practice [Bar-Yossef et al. 2004; Kushilevitz and Nisan 1996]. With respect to any topology G, we similarly define $\text{FT}_0(\text{SUM}_G, b)$ and $\text{FT}_{\epsilon,\delta}(\text{SUM}_G, b)$ across all fault-tolerant randomized SUM protocols.

[4]To make the discussion more accessible, here we use the notation NFT and FT instead of $\mathcal{R}$ (for randomized communication complexity) loaded with various superscripts, as in classic communication complexity literature.

For any given integer N, we define SUM's *NFT communication complexity* $\text{NFT}_0(\text{SUM}_N, b)$ and $\text{NFT}_{\epsilon,\delta}(\text{SUM}_N, b)$ to be the maximum $\text{NFT}_0(\text{SUM}_G, b)$ and $\text{NFT}_{\epsilon,\delta}(\text{SUM}_G, b)$, respectively, across all topology G's where G is connected and has exactly N nodes. Similarly define SUM's *FT communication complexity* $\text{FT}_0(\text{SUM}_N, b)$ and $\text{FT}_{\epsilon,\delta}(\text{SUM}_N, b)$.

Communication Complexity of Two-Party Problems. Our proofs will also need to reason about the NFT communication complexity of some two-party problems. In such a problem Π, Alice and Bob each have an input X and Y respectively, and the goal is to compute the function $\Pi(X, Y)$. For all two-party problems in this article, we only require Alice to learn the final result. We will often use n to denote the size of Π, as compared to N which describes the number of nodes in G. The *communication complexity* of a randomized protocol for computing Π is defined to be either the average-case or worst-case (over random coin flips) number of bits sent by Alice and Bob combined. In the classic setting without synchronous rounds and time complexity constraints [Kushilevitz and Nisan 1996], similar as earlier, we define $\text{NFT}_0(\Pi)$ to be the smallest average-case communication complexity across all randomized protocols that can generate a zero-error result for Π, and define $\text{NFT}_{\epsilon,\delta}(\Pi)$ to be the smallest worst-case communication complexity across all randomized protocols that can generate an (ϵ, δ)-approximate result for Π. We will also need to consider a second setting with synchronous rounds,[5] adapted from Impagliazzo and Williams [2010]. Here Alice and Bob proceed in synchronous rounds, where in each round Alice and Bob may simultaneously send a message to the other party. Alice, or Bob, or both may also choose not to send a message in a round. The *time complexity* of a randomized protocol for computing Π is defined to be the number of rounds needed for the protocol to terminate, over the worst-case input and the worst-case coin flips. We define $\text{NFT}_0(\Pi, t)$ to be the smallest average-case communication complexity across all randomized protocols for Π that can generate a zero-error result within a time complexity of at most t rounds. Similarly, we also define $\text{NFT}_{\epsilon,\delta}(\Pi, t)$ to be the smallest worst-case communication complexity across all randomized protocols for Π that can generate an (ϵ, δ)-approximate result within a time complexity of at most t rounds.

4. UPPER BOUNDS ON NFT COMMUNICATION COMPLEXITY OF SUM

This section describes the NFT upper bounds on SUM, which are from well-known tree-aggregation protocols coupled with some standard tricks. These are not our main contribution—instead, they serve to show the exponential gap from our FT lower bounds.

Tree-Aggregation Protocol and Defining $\Lambda(G)$. Since the topology G is known, every node can locally and deterministically construct a breadth-first spanning tree (with the root of G being the tree root) as the aggregation tree. With this tree in place, a node becomes *ready* when it receives one *aggregation message* from each of its children. Each *aggregation message* encodes the partial sum of all the values in the corresponding subtree. Leaf nodes are *ready* from the beginning. A ready node will combine all these aggregation messages, together with its own value, and then send a single aggregation message to its parent. With the known topology, the protocol easily avoids collision via the following simple deterministic scheduling: Out of all ready nodes, the protocol greedily and deterministically chooses a maximal set of nodes to send messages without incurring collision. A message does not need to include the sender's id—since everything

[5]These synchronous rounds are different from *interaction rounds* which correspond to message exchanges. A protocol using x synchronous rounds incurs x or fewer interaction rounds since a synchronous round may or may not have any message.

is deterministic, the receiver can locally determine the sender. The function $\Lambda(G)$ is formally defined to be the number of rounds needed for this deterministic protocol to finish on G. Thus, by definition, this protocol has a time complexity of one aggregation round.

NFT Upper Bounds. If each aggregation message uses $O(\log N)$ bits to encode the exact partial sum, then the above protocol is a deterministic protocol for SUM with $O(\log N)$ communication complexity and one aggregation round time complexity. For (ϵ, δ)-approximate results, it is possible to reduce the size of the aggregation message to $O(\log \frac{1}{\epsilon} + \log\log N)$ bits, using a simple private-coin protocol with similar tricks as in AMS synopsis [Alon et al. 1996] (see Appendix B). One can further reduce the communication complexity if the time complexity is b aggregation rounds with $b > 1$, since we can now spend b rounds in sending all the bits previously sent in one round. It is known [Impagliazzo and Williams 2010] that an a-bit message sent in one round can be encoded using $a/\log \frac{b}{a}$ bits sent over b rounds, for $b \geq 2a$. To do so, one bit is sent every $\frac{b}{a} \cdot \log \frac{b}{a}$ rounds. Leveraging the round number during which the bit is sent, each such bit can encode $\log(\frac{b}{a} \cdot \log \frac{b}{a}) \geq \log \frac{b}{a}$ bits of information. Combining all of these leads to the following.

THEOREM 4.1. *For any $b \geq 1$, we have:*

$$\mathrm{NFT}_0(\mathrm{SUM}_N, b) = O\left(a/\log\left(\frac{b}{a}+2\right)\right), \text{ where } a = \log N$$

$$\mathrm{NFT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}_N, b) = O\left(a/\log\left(\frac{b}{a}+2\right)\right), \text{ where } a = \log\frac{1}{\epsilon} + \log\log N, \text{ for } \epsilon = \Omega\left(\frac{1}{N}\right).$$

5. LOWER BOUNDS ON FT COMMUNICATION COMPLEXITY OF SUM FOR $b \leq 2 - c$

This section aims to eventually prove the following theorem.

THEOREM 5.1. *For any $b \in [1, 2-c]$ where c is any positive constant, we have:*

$$\mathrm{FT}_0(\mathrm{SUM}_N, b) = \Omega\left(\frac{N}{\log^2 N}\right)$$

$$\mathrm{FT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}_N, b) = \Omega\left(\frac{1}{\epsilon^2 \log N}\right), \quad \text{for } \epsilon = \Omega\left(\frac{\sqrt{\log N}}{\sqrt{N}}\right).$$

This theorem establishes the FT lower bounds for the leftmost region of b in Figure 1 (i.e., for $1 \leq b \leq 2 - c$). In the following, Section 5.1 first gives an overview of our proof for this theorem, which is based on a reduction from UNIONSIZE to SUM. In other words, given any oracle protocol for solving SUM, we will construct a protocol for solving UNIONSIZE. Section 5.2 elaborates the concrete intuitions behind our reduction. The formal reasoning and proofs then follow: Section 5.3 develops a formal framework for our reasoning and Section 5.4 proves Theorem 5.1 using that framework.

5.1. Overview of Our Proof

The UNIONSIZE *Problem.* As discussed in Section 1, one possible approach to achieve fault tolerance when computing SUM is for the nodes to simultaneously propagate their values along multiple directions. But doing so will lead to duplicates which must be addressed. Thus, it is natural to consider a potential reduction from the two-party communication complexity problem UNIONSIZE, which was used for obtaining the optimal $\Omega(\frac{1}{\epsilon^2})$ lower bound on the space complexity of one-pass distinct element counting [Woodruff 2004]. In the two-party problem UnionSize$_n$, Alice and Bob have length-n binary strings X and Y, respectively. Let X_i (Y_i) denote the ith bit of X (Y). Alice aims to determine $|\{i \mid X_i \neq 0 \text{ or } Y_i \neq 0\}|$. If X and Y are the characteristic vectors of two sets, then this is the size of the union of the two sets. Trivially combining a few recent results [Chakrabarti and Regev 2011; Impagliazzo and Williams 2010; Woodruff 2004] tells us that $\text{NFT}_0(\text{UnionSize}_n, O(\text{poly}(n))) = \Omega(\frac{n}{\log n})$ and $\text{NFT}_{\epsilon,\frac{1}{3}}(\text{UnionSize}_n, O(\text{poly}(n))) = \Omega(\frac{1}{\epsilon^2 \log n})$ for $\epsilon = \Omega(\frac{1}{\sqrt{n}})$ (see Corollary A.4 in Appendix A).

Overview of Our Reduction and Its Novelty. While the well-known reduction [Woodruff 2004] from UnionSize to the (centralized) one-pass distinct element counting problem is straightforward, we seek a reduction from UnionSize to Sum, which is less so. In particular, it is not immediately clear what a role failures can play. Our simple yet interesting reduction here will answer this question, which prepares for our trickier reduction in Section 6. Our reduction is based on a certain topology G. Given inputs X and Y to UnionSize, each node in G has some value so that their sum is exactly UnionSize(X, Y). The values of some of the nodes are uniquely determined by X, and thus are known by Alice from her local knowledge of X. If the value of a node τ cannot be uniquely determined by X, then τ is *spoiled* (rigorously defined in Section 5.3) for Alice, in the sense that Alice cannot simulate τ. As the simulation proceeds, a spoiled node τ may causally affect its neighbor node τ', rendering Alice unable to simulate τ' and thus making τ' spoiled as well. Since the Sum protocol may have internal state, if Alice cannot simulate a node for some round, then Alice cannot simulate the node for later rounds either. In this sense, a spoiled node can never get "unspoiled" later. For each round, Alice will simply simulate the (shrinking) group of all those nodes that have not been spoiled for Alice. Bob similarly simulates all unspoiled nodes for Bob. Alice's group and Bob's may intersect.

We want the root of G to remain unspoiled for Alice when the Sum protocol ends, so that it provides the Sum result to Alice for her to determine UnionSize(X, Y). To achieve this, in the reduction, Alice and Bob will need to strategically simulate the failures of certain nodes, to block the spreading of spoiled nodes. This showcases the fundamental role of failures in our reduction. At the same time, we need to avoid failing/disconnecting nodes with a value of 1—failing/disconnecting them would enable the Sum protocol to ignore their values and potentially return a result that cannot be used to determine UnionSize(X, Y). (Recall from Section 3 that a zero-error result for Sum can be any value between s_1 and s_2.) In fact, if we were not concerned with this, then simply failing all nodes except the root would keep the root unspoiled forever. Finally, it is also necessary to enlist help from Bob, who can simulate certain nodes that are spoiled for Alice. By forwarding to Alice messages sent by those nodes, Bob can further hinder the shrinking of Alice's group. The communication (between Alice and Bob) spent in doing so will be the communication complexity incurred for solving UnionSize. Simulating a shrinking group of nodes and properly using failures to hinder such shrinking is the main novelty in our simple reduction.

5.2. Intuitions for Our Reduction from UnionSize to Sum

We now intend to develop some concrete intuitions for our reduction from UnionSize to Sum. For better understanding, we first describe here a topology (Figure 2) that allows a reductions from UnionSize to Sum for $b \leq \frac{10}{9}$. Later, Section 5.4 will use an improved topology so that b can be any value below $2 - c$, with c being any positive constant. Given UnionSize$_n$ with n being a power of 2, the topology G here has n parallel chains of nodes. Each chain has $6 \log n + 1$ nodes. We use γ_i^α, τ_i, and γ_i^β to denote the first,

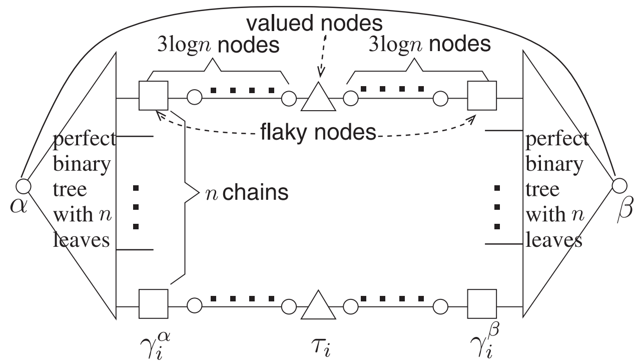

Fig. 2. FT lower bound topology for $b \leq \frac{10}{9}$.

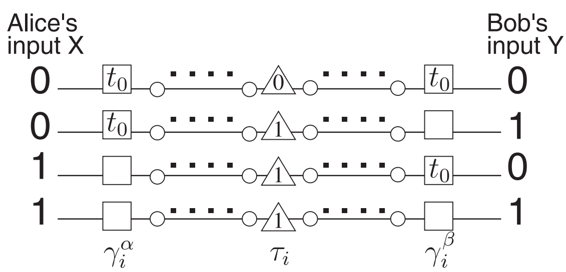

Fig. 3. Values of valued nodes and failure times of flaky nodes, for $X = 0011$ and $Y = 0101$.

middle, and last node on the ith chain, respectively. Next, we construct a perfect binary tree with all the γ_i^α's being the leaves, and let node α denote the tree root. Similarly construct a second perfect binary tree whose leaves are all the γ_i^β's, and let β be the tree root. Finally, we connect α and β with a single edge, and let α be the root of the topology. This topology has total $N = \Theta(n \log n)$ nodes.

The inputs X and Y to UnionSize$_n$ will determine the values of the τ_i's, which are called *valued nodes*. Specifically, τ_i has a binary value of 1 iff $X_i \neq 0$ or $Y_i \neq 0$ (Figure 3). All other nodes (i.e., nonvalued nodes) have values of 0. X and Y also determine the failure times of the γ_i^α's and γ_i^β's, which are called *flaky nodes*. If $X_i = 0$, then γ_i^α fails at the beginning of round $t_0 = 3\log n + 1$. Otherwise, it never fails. Intuitively, t_0 is the very first round where τ_i may causally affect γ_i^α. Similarly, γ_i^β fails at the beginning of round t_0 iff $Y_i = 0$ (Figure 3). Non-flaky nodes never fail. It is worth noting that this failure adversary (i) is actually oblivious to the Sum protocol, and (ii) fails only a vanishingly small fraction (i.e., $o(N)$) of all the nodes in G.

As a key property in the above construction (and later constructions), a τ_i whose value is 1 is never disconnected from the root. This is because if τ_i's value is 1, then it must be unspoiled (by our construction) for either Alice or Bob, and thus can remain connected to α or β (and thus to the root). This in turn ensures that a zero-error result of Sum is always exactly UnionSize(X, Y).

In each round, Alice simulates the group of all the unspoiled nodes for Alice, including node α. Bob similarly simulates the unspoiled nodes for Bob, including node β. These two groups are made precise later in Section 5.4. Whenever α in the Sum protocol sends a message (whose intended recipient may or may not be β) Alice always forwards that message to Bob. Bob does the same whenever β sends a message. Alice and Bob do not exchange any additional messages. Thus, the number of bits sent by Alice and Bob for solving UnionSize is exactly the same as the number of the bits sent by α and β in the Sum protocol.

The need for α (β) to remain unspoiled for Alice (Bob) determines that this simulation cannot last forever. For example, let us consider some i where $X_i = 0$ and $Y_i = 1$. This makes τ_i spoiled for Alice, since Alice cannot determine τ_i's value based on X_i. To prevent τ_i from causally affecting α and thus spoiling α, Alice simulates the failure of γ_i^α before this can happen. Interestingly, since based on Y_i Bob cannot determine whether γ_i^α fails, γ_i^α becomes spoiled for Bob when it fails. Once the failure of γ_i^α can causally affect β (at round $10 \log n + 1$), Bob can no longer simulate β. The simulation must end before this happens, which is guaranteed under $b \leq \frac{10}{9}$ since an aggregation round here has no more than $8 \log n + 1$ rounds.

5.3. A Formal Framework for Reasoning about Reductions to Sum

Having provided the overview and intuitions, we are now ready for the formal reasoning. To facilitate our proof in Section 5.4, we develop a formal framework in this section.

Because FT communication complexity has not been formally studied before, many of the concepts in this framework need to be defined from scratch. This formal framework will also be used for the proofs in later sections; hence, the framework developed here will be more general than what is needed for proving Theorem 5.1. A significantly simplified version of our framework, which does not involve failures, is implicitly used by Sarma et al. [2011] for a reduction from the Disjointness two-party problem to distributed spanning tree verification. Applying the simplified version of our framework to their context would also streamline their proof.

Rounds and Failures. The execution of the Sum oracle protocol starts at round 1. We sometimes for convenience also discuss round 0, during which the Sum protocol does nothing. Note that one can assume, without loss of generality, that all failures happen at the beginning of various rounds:[6] If a node v fails sometime within round r, since v can (locally) broadcast at most one message in a round, the failure can be viewed as happening at the beginning of round $r + 1$ if the failure occurs after v sends the message. Otherwise, the failure can be viewed as happening at the beginning of round r.[7] Thus, from now on, we will assume that all failures happen at the beginning of various rounds. If a node fails at the beginning of round r, we say that the *failure time* of that node is round r.

Simulating a Node. To properly *simulate* a node (i.e., simulate the execution of the Sum oracle protocol on that node) in a certain round, Alice (Bob) needs to feed all necessary parameters to the oracle protocol running on that node. The execution of a randomized oracle protocol on a node in a given round is uniquely determined by the (public) coin flips, the topology (since the topology is known), the id of the node, the (initial) value of the node, the failure time of the node (i.e., a failed node should not send out messages and the oracle protocol should not be invoked on such a node), and all the incoming messages to this node since round 1. Alice can easily generate the coin flips, and she already knows the topology and node id. For some nodes, Alice can uniquely determine their values and failure time based on Alice's input X. Finally, the incoming messages to a node v will have to be obtained via Alice's simulation of v's

[6]This implies that our failure adversary in Section 5.2, which injects failures only at the beginning of various rounds, has already fully exploited the flexibility on failure time.

[7]For wired network settings with point-to-point communication, this assumption will no longer be without of loss of generality. While all our final results will still hold without any modification, the formal framework needs to be slightly different.

neighbors or directly from Bob if β is the sender of that message. Recall that in a round, a node first performs some local computation, and then does either a send or a receive. In particular, the potential message received by a node can only affect its behavior starting from the next round, since the node does not do any further local computation in the current round after the receive operation. Thus, regardless of whether v does a send or receive in round r, to simulate v in round r, we (only) need all the incoming messages to v from round 1 to round $r-1$ (inclusive).

Epicenters and Their Occurrence Time. The following concepts are always defined with respect to a given input X of Alice's. A node v in the topology G is a *value epicenter* if v's value is not uniquely determined by X. Namely given X, there exists Bob's inputs Y and Y' such that v's value is different under the simulated execution of the Sum oracle protocol (used in the reduction) for (X, Y) and (X, Y'). A node v is a *failure epicenter* if v is not already a value epicenter and if v's failure time is not uniquely determined by X. Value epicenters and failure epicenter are all called *epicenters*.

The *occurrence time* of a value epicenter v is defined to be round 1. The *occurrence time* of a failure epicenter v is defined to be v's *earliest* failure time, across all valid Y's given the current X. To get some intuition behind the occurrence time, consider a failure epicenter v. Suppose that given X, the only possible inputs to Bob are Y and Y'. Imagine that v fails at the beginning of round 3 if Bob's input is Y, and fails at the beginning of round 8 if Bob's input is Y'. Since Alice does not know Bob's input, starting from round 3, Alice no longer knows whether v is still alive and thus can no longer simulate v. This also explains why a value epicenter v has an occurrence time of round 1—Alice cannot simulate v even for round 1.

Spoil Paths and Spoiled Nodes. All the following concepts are still with respect to a given input X of Alice's. If a node's failure time r is uniquely determined by X, we say that the node fails *stably* at the beginning of round r. We also call such a failure a *stable failure*. A *spoil path* from an epicenter u_0 (occurring at round r_0) to a node v is a sequence of nodes $u_0, u_1, u_2, \ldots, u_k, v$ where

—for $0 \le i \le k$, $u_i \ne \alpha$ and $u_i \ne \beta$,
—v is u_k's neighbor and u_i is u_{i-1}'s neighbor for $1 \le i \le k$,
—v has not failed stably before the beginning of round $r_0 + k + 2$, and u_i has not failed stably before the beginning of round $r_0 + i + 1$ for $0 \le i \le k$. Intuitively, this enables u_i to potentially send a message to u_{i+1} (and also u_{i+1} to receive this message) in round $r_0 + i + 1$. In turn, starting from round $r_0 + i + 2$, u_{i+1}'s behavior may potentially be affected by this message.

We define the *length* of a spoil path $u_0, u_1, u_2, \ldots, u_k, v$ to be $k + 1$. Intuitively, a spoil path is a potential path for u_0 to causally affect v, without going through α or β. Since Alice (Bob) will send to the other party all messages sent by α (β), paths going through α (β) are already taken care of. We intentionally define spoil paths in such a way that they can only be "blocked" by stable failures. This makes this definition consistent with the following intuition: If a node on a spoil path fails and if that failure is not a stable failure, then that node must be an epicenter itself and will already cause the spreading of spoiled nodes. Thus, intuitively, such a nonstable failure can never block the spreading of spoiled nodes. The *spoil distance* of a node v from an epicenter u_0 occurring at round r_0 is simply the length of the shortest spoil path from u_0 to v, or infinite if there is no such spoil path. For any epicenter u_0, we also define the spoil distance of u_0 from itself to be 0. For any given round r, a node v is *spoiled* in round r with respect to Alice's input X if v is within spoil distance of $r - r_0$ hops from some epicenter occurring at round r_0 where $r_0 \le r$. By such definition, an epicenter with occurrence time of r becomes first spoiled in round r, which is consistent with the intuition. We use $S_{A,X}(r)$ to denote the set of all spoiled nodes at round r with respect to Alice's input X. We will prove in the next section that in each round r, Alice with input X can simulate all unspoiled nodes (i.e., all nodes in $\overline{S}_{A,X}(r)$).

We similarly define the notion of epicenters, spoil paths, spoiled nodes, and $S_{B,Y}(r)$ for Bob and his input Y.

The Simulatability Lemma. Given the formal framework, we can now prove the following simple but useful lemma which we will repeatedly invoke later.

LEMMA 5.2. *Let X be Alice's input and Y be Bob's. Let R be any positive integer where $\alpha \in \overline{S}_{A,X}(R)$ and $\beta \in \overline{S}_{B,Y}(R)$. Assume that Alice (Bob) always forwards to the other party the message sent by α (β) in a round whenever Alice (Bob) is able to simulate the execution of the* SUM *oracle protocol on α (β) for that round. Then, for all $0 \le r \le R$, Alice can properly simulate the execution of the* SUM *oracle protocol on all nodes in $\overline{S}_{A,X}(r)$ for round r and Bob can properly simulate all nodes in $\overline{S}_{B,Y}(r)$ for round r.*

PROOF. We do an induction on r. First, $\alpha \in \overline{S}_{A,X}(R)$ and $\beta \in \overline{S}_{B,Y}(R)$ imply $\alpha \in \overline{S}_{A,X}(r)$ and $\beta \in \overline{S}_{B,Y}(r)$ for all $0 \le r \le R$. $\overline{S}_{A,X}(0)$ simply contains all nodes, since there are no epicenters occurring in round 0. Clearly, Alice can simulate all nodes for round 0 since the SUM protocol does nothing in round 0 and no failures happen in round 0. Similarly, for round 0, Bob can simulate all nodes in $\overline{S}_{B,Y}(0)$.

Assume that the claim holds for round r, and consider any node $v \in \overline{S}_{A,X}(r+1)$. We distinguish two cases.

—v is not an epicenter for Alice's input X. Then, Alice can uniquely determine both the (initial) value and the failure time of v. If the failure time is round $r+1$ or earlier, then Alice trivially simulates v in round $r+1$ by doing nothing and we are done. Otherwise, Alice knows that v is alive in round $r+1$.

—v is an epicenter for Alice's input X. We claim that it is impossible for the occurrence time of this epicenter to be round $r+1$ or earlier, since otherwise v would have been spoiled in round $r+1$ and thus would not be in $\overline{S}_{A,X}(r+1)$. Given that the occurrence time is round $r+2$ or later, it means that the occurrence time is not round 1. Thus, v is not a value epicenter and Alice must know v's initial value. Furthermore, while Alice cannot determine v's exact failure time, Alice knows for sure that the failure time of v is round $r+2$ or later, and that v is alive in round $r+1$.

Now we only need to prove that Alice can simulate v in round $r+1$, given that Alice knows v's initial value and that v is alive in round $r+1$.

We trivially have $v \in \overline{S}_{A,X}(r)$ and by inductive hypothesis, Alice can simulate v from round 1 to r (inclusive). It thus suffices to prove that Alice can generate the potential message that v receives in round r from some neighbor u, so that Alice can simulate v in round $r+1$. We distinguish three cases for u. If u is β and since $\beta \in \overline{S}_{B,Y}(r)$, by inductive hypothesis, Bob can properly simulate β for round r. By condition of the lemma, Bob must have forwarded the message from β to Alice. Similarly, if u is α and since $\alpha \in \overline{S}_{A,X}(r)$, Alice can properly simulate α for round r and generate the message herself. Finally, if $u \neq \alpha$ and $u \neq \beta$, we first show that u must be in $\overline{S}_{A,X}(r)$, via a contradiction. Since u sends a message in round r, u must have not failed in round r or earlier. In turn, u must have not failed stably in round r or earlier. Thus, if $u \notin \overline{S}_{A,X}(r)$ (i.e., u is spoiled in round r), then v must be spoiled in round $r+1$, which contradicts with $v \in \overline{S}_{A,X}(r+1)$. Now, given that $u \in \overline{S}_{A,X}(r)$, by inductive hypothesis

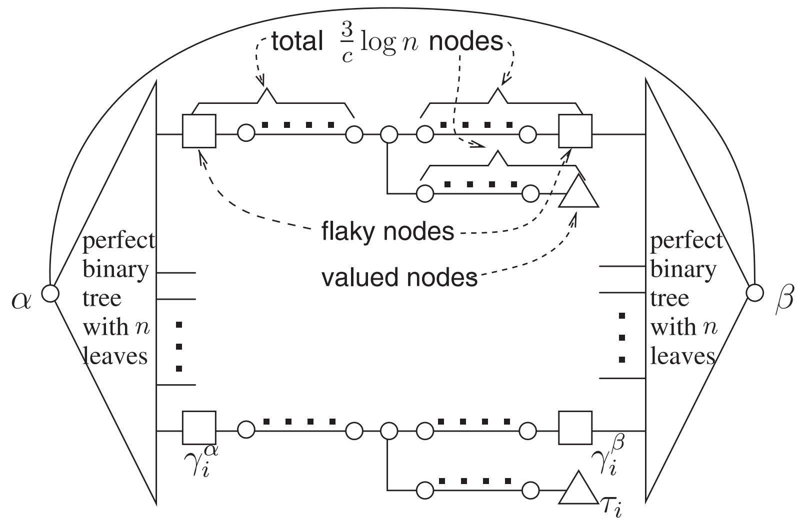

Fig. 4. FT lower bound topology for $b \le 2 - c$.

Alice can simulate u for round r and generate u's message locally. Thus, Alice has all the information needed to simulate v for round $r + 1$.

Similar arguments apply to Bob. □

5.4. Proof for Theorem 5.1

To prove Theorem 5.1 using the previous formal framework, we use an improved topology G as in Figure 4. The n chains in the topology in Figure 2 are now replaced with n *T-structures*, where n is still a power of 2. The ith T-structure has three sequences of $\frac{3}{c}\log n$ nodes (total $\frac{9}{c}\log n$ nodes) attached to a degree-3 node in the middle. For the first sequence, the last node at the other end is a valued node τ_i. Let γ_i^α and γ_i^β denote the last node at the other end of each of the remaining two sequences, respectively. Both of them are flaky nodes. Same as in Figure 2, we next construct a perfect binary tree with all the γ_i^α's being the leaves, and let node α denote the tree root. Similarly construct a second perfect binary tree whose leaves are all the γ_i^β's, and let node β denote the tree root. Finally, we connect α with β using a single edge, and let α be the root of the topology. It is easy to verify that there are $N = \frac{9}{c} n \log n + 3n - 2$ nodes in the topology. The valued node τ_i has a binary value of 1 iff $X_i \neq 0$ or $Y_i \neq 0$. If $X_i = 0$, then the flaky node γ_i^α fails at the beginning of round $t_0 = \frac{6}{c}\log n + 1$. Otherwise, it never fails. Similarly, γ_i^β fails at the beginning of round t_0 iff $Y_i = 0$.

We now prove a series of lemmas, which will eventually lead to a proof for Theorem 5.1. The first simple lemma (Lemma 5.3) shows that, in this construction, α (β) will always remain unspoiled for Alice (Bob) in round R, where R is large enough for the Sum oracle protocol to have terminated.

LEMMA 5.3. *Consider the topology, valued nodes (with their values), and flaky nodes (with their failure times), as described in Section 5.4. Under this construction and under* $R = \frac{12}{c}\log n + \log n$*, for all possible input X of Alice's, we have* $\alpha \in \overline{S}_{A,X}(R)$*. Similarly, for all possible input Y of Bob's, we have* $\beta \in \overline{S}_{B,Y}(R)$.

PROOF. Without loss of generality, we prove $\alpha \in \overline{S}_{A,X}(R)$. With respect to X, there are only two kinds of epicenters. The flaky node γ_i^{β} ($1 \le i \le n$) is always a failure epicenter, occurring at round $t_0 = \frac{6}{c}\log n + 1$. One can easily verify that the spoil distance from γ_i^{β} to α is at least $\frac{6}{c}\log n + \log n$. If $X_i = 0$, then τ_i is a value epicenter with respect to X. The spoil distance from τ_i to α is at least $\frac{12}{c}\log n + \log n$, since the shortest possible spoil path (of length $\frac{6}{c}\log n + \log n$) is blocked by the stable failure of γ_i^{α}. □

The next lemma is the main lemma, which proves the reduction from UNIONSIZE$_n$ to SUM on an N-node topology. Given any n that is a power of 2, the topology as constructed in Figure 4 has total $N = \frac{9}{c}n\log n + 3n - 2$ nodes. The lemma will first prove the reduction for such values of N. To reason about the asymptotic lower bound with respect to N later, however, we need to cover all values of N. Thus, the lemma moves on to prove a reduction for those values of N where the equation $\frac{9}{c}n\log n + 3n - 2 = N$ does not have a power-of-2 solution for n.

LEMMA 5.4. *Consider any positive constant c, any $b \in [1, 2-c]$, and any sufficiently large integer N. Let n be the largest integer such that n is a power of 2 and $\frac{9}{c}n\log n + 3n - 2 \le N$. There exists a connected topology G with N nodes, such that:*

$$\mathrm{FT}_0(\mathrm{SUM}_G, b) \ \ge\ \frac{1}{2}\mathrm{NFT}_0(\mathrm{UNIONSIZE}_n, bN),$$
$$\mathrm{FT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}_G, b) \ \ge\ \frac{1}{2}\mathrm{NFT}_{\epsilon,\frac{1}{3}}(\mathrm{UNIONSIZE}_n, bN).$$

PROOF. We first prove the lemma for $N = \frac{9}{c}n\log n + 3n - 2$. We construct G as in Figure 4, and let $R = \frac{12}{c}\log n + \log n$. Under the failure adversary constructed in Section 5.4, let $\lambda = \max_{G' \in \mathcal{G}} \Lambda(G')$ where $\mathcal{G}$ is the set of topologies that have ever appeared during the execution. We first prove that $R \ge (2-c)\lambda$ for all $n \ge 2$. Since in our construction all failures are injected at the same time, $\mathcal{G}$ actually only contains two topologies: one before failures and one after failures. It is easy to verify that for either $G' \in \mathcal{G}$, $\Lambda(G')$ is at most $\frac{6}{c}\log n + 2\log n + 1$. Here, the $2\log n$ term takes care of the need to avoid collision on the internal trees nodes—without collision, it would be only $\log n$ rounds. We now have

$$\frac{R}{\lambda} \ge \frac{\frac{12}{c}\log n + \log n}{\frac{6}{c}\log n + 2\log n + 1} > 2 - c.$$

Next we reduce the UNIONSIZE$_n$ problem (in the synchronous rounds setting) to SUM. Consider any (black-box) oracle protocol for SUM. Given input X to Alice in UNIONSIZE$_n$, Alice will simulate the execution of the oracle protocol on all nodes in $\overline{S}_{A,X}(r)$ at round r for $0 \le r \le R$. Similarly, given input Y to Bob, Bob simulates all nodes in $\overline{S}_{B,Y}(r)$. Furthermore, whenever α sends a message, Alice will forward that message to Bob. The same applies to Bob and β. Note that it is possible for Alice and Bob to send each other a message simultaneously in one round. By Lemma 5.2 and Lemma 5.3, such simulation is possible. When the oracle protocol terminates, which must be no later than round $(2-c)\lambda \le R$ since the time complexity of the oracle protocol is no larger than $2-c$, α and thus Alice will know the final result of the sum. By our construction, the zero-error result of the sum on G exactly equals UNIONSIZE(X, Y), and thus any (ϵ, δ)-approximate result of the sum is also an (ϵ, δ)-approximate result of UNIONSIZE(X, Y). The total amount of communication between Alice and Bob is exactly the total number of bits sent by α and β combined in the above simulation. Thus either

α or β must have sent at least half of the total number of bits sent by Alice and Bob. Together with the trivial property that $\lambda \leq N$, we now have

$$\mathrm{FT}_0(\textsc{Sum}_G, b) \ \geq \ \frac{1}{2}\mathrm{NFT}_0(\textsc{UnionSize}_n, b\lambda) \ \geq \ \frac{1}{2}\mathrm{NFT}_0(\textsc{UnionSize}_n, bN),$$
$$\mathrm{FT}_{\epsilon,\frac{1}{3}}(\textsc{Sum}_G, b) \ \geq \ \frac{1}{2}\mathrm{NFT}_{\epsilon,\frac{1}{3}}(\textsc{UnionSize}_n, b\lambda) \ \geq \ \frac{1}{2}\mathrm{NFT}_{\epsilon,\frac{1}{3}}(\textsc{UnionSize}_n, bN).$$

We still need to prove the lemma for $N > \frac{9}{c}n\log n + 3n - 2$. Let $N_1 = \frac{9}{c}n\log n + 3n - 2$. We first construct a connected topology G_1 with N_1 nodes as in Figure 4. Next, we add $N_2 = N - N_1 = O(n\log n)$ extra nodes to G_1 to obtain the topology G with N nodes. Those N_2 nodes will always have a value of 0 and will never fail. Since we want G to be connected, we need to attach those N_2 nodes to some existing nodes in G_1. We want to do so carefully so that (i) Lemma 5.3 continues to hold after adding those nodes, and (ii) the length of an aggregation round is not affected by adding those nodes. These two properties will allow our earlier proof (for $N = \frac{9}{c}n\log n + 3n - 2$) to carry over without modification. Specifically, we partition the N_2 nodes into n equal-sized groups, with each group having $O(\log n)$ nodes. We then have each group form a binary tree of height $O(\log\log n)$. Finally, we attach the root of the ith binary tree to the middle node (i.e., the degree-3 node) of the ith T-structure in G_1.

It is trivial to verify that Lemma 5.3 continues to hold after adding those N_2 nodes in this way. Next to understand why the length of an aggregation round is never affected by those extra N_2 nodes, consider a given T-structure and the binary tree attached to the middle node of that T-structure. Recall that the length of an aggregation round is the number of rounds needed for the deterministic tree-aggregation protocol in Section 4 to terminate. When running that protocol, the root of the binary tree here will send an aggregation message to the middle node of the T-structure in round $O(\log\log n)$. On the other hand, under any G' in $\mathcal{G}$ where $\mathcal{G}$ is the set of topologies that have ever appeared during the execution, that middle node will never receive any other aggregation message before round $\frac{3}{c}\log n - 1$. Thus, this extra binary tree (i) is itself not a bottleneck for the tree-aggregation protocol to terminate, and (ii) never potentially collides with other aggregation messages. In turn, this means that the length of an aggregation round in the execution under G is the same as the length of an aggregation round in the execution under G_1.

With these two properties in G, we now know that our earlier proof for $N = \frac{9}{c}n\log n + 3n - 2$ carries over to $N > \frac{9}{c}n\log n + 3n - 2$ without modification. □

The following proof for Theorem 5.1 follows naturally from the Lemma 5.4.

Proof for Theorem 5.1. For any sufficiently large N, consider the N-node connected topology G as constructed by Lemma 5.4. We trivially have

$$\mathrm{FT}_0(\textsc{Sum}_N, b) \geq \mathrm{FT}_0(\textsc{Sum}_G, b) \geq \frac{1}{2}\mathrm{NFT}_0(\textsc{UnionSize}_n, bN),$$
$$\mathrm{FT}_{\epsilon,\frac{1}{3}}(\textsc{Sum}_N, b) \geq \mathrm{FT}_{\epsilon,\frac{1}{3}}(\textsc{Sum}_G, b) \geq \frac{1}{2}\mathrm{NFT}_{\epsilon,\frac{1}{3}}(\textsc{UnionSize}_n, bN).$$

By Lemma 5.4, the n in these inequalities is the largest integer that is a power of 2 and satisfies $\frac{9}{c}n\log n + 3n - 2 \leq N$. Thus, we have $N = \Theta(n\log n)$. Applying Corollary A.4

gives

$$\begin{aligned}
\mathrm{FT}_0(\textsc{Sum}_N, b) &\geq \frac{1}{2}\mathrm{NFT}_0(\textsc{UnionSize}_n, bN) \geq \frac{1}{2}\mathrm{NFT}_0(\textsc{UnionSize}_n, (2-c)N)\\
&= \Omega\left(\frac{n}{\log n}\right) = \Omega\left(\frac{N}{\log^2 N}\right)\\
\mathrm{FT}_{\epsilon,\frac{1}{3}}(\textsc{Sum}_N, b) &\geq \frac{1}{2}\mathrm{NFT}_{\epsilon,\frac{1}{3}}(\textsc{UnionSize}_n, bN) \geq \frac{1}{2}\mathrm{NFT}_{\epsilon,\frac{1}{3}}(\textsc{UnionSize}_n, (2-c)N)\\
&= \Omega\left(\frac{1}{\epsilon^2 \log n}\right) = \Omega\left(\frac{1}{\epsilon^2 \log N}\right), \quad \text{for } \epsilon = \Omega\left(\frac{\sqrt{\log N}}{\sqrt{N}}\right). \quad \square
\end{aligned}$$

6. LOWER BOUNDS ON FT COMMUNICATION COMPLEXITY OF SUM FOR $b \leq N^{0.25-c}$ OR $b \leq 1/\epsilon^{0.5-c}$

This section aims to eventually prove the following theorem.

THEOREM 6.1. *For any $b \geq 1$, we have:*

$$\begin{aligned}
\mathrm{FT}_0(\textsc{Sum}_N, b) &= \Omega\left(\frac{\sqrt{N}}{b^2 \log N}\right)\\
\mathrm{FT}_{\epsilon,\frac{1}{3}}(\textsc{Sum}_N, b) &= \Omega\left(\frac{1}{\epsilon b^2 \log N}\right), \quad \textit{for } \epsilon = \Omega\left(\frac{1}{\sqrt[4]{N}}\right).
\end{aligned}$$

This theorem establishes the FT lower bounds for the middle region of b in Figure 1 (i.e., for $2-c < b \leq N^{0.25-c}$ or $2-c < b \leq \frac{1}{\epsilon^{0.5-c}}$). Note that while the theorem applies to all $b \geq 1$, the result in the previous section is stronger when $1 \leq b \leq 2-c$. Similarly, for $b > N^{0.25-c}$ or $b > \frac{1}{\epsilon^{0.5-c}}$, the theorem only gives trivial lower bounds. In the following, Section 6.1 first gives an overview of our proof for this theorem, which is based on a reduction from UNIONSIZECP to SUM. Section 6.2 elaborates the concrete intuitions behind our reduction. The formal reasoning and proofs then follow: Section 6.3 proves our lower bound on the communication complexity of UNIONSIZECP, and Section 6.4 proves Theorem 6.1.

6.1. Overview of Our Proof

Why the Previous Construction Cannot be Extended. The FT lower bounds in Section 5 no longer hold for larger b since the failure of γ_i^α (as simulated by Alice) makes γ_i^α spoiled for Bob, which will in turn spoil β under larger b. A natural attempt to fix this is to inject new failures to prevent such propagation of spoiled nodes, as in Figure 5. Here, when $Y_i = 1$, Bob simulates a new failure to the left[8] of τ_i, to prevent the propagation of spoiled nodes due to γ_i^α. Similarly, Alice needs to simulate a new failure on the right side of τ_i, when $X_i = 1$. This eventually implies that when $X_i = Y_i = 1$, both of these two new failures will be introduced, disconnecting τ_i. One could avoid this problem by adding a promise and disallowing X_i and Y_i to simultaneously be 1. Unfortunately, such a naive promise decreases the communication complexity of UNIONSIZE$_n$ to $O(\log n)$, failing to yield the exponential gap that we hope for.

The UNIONSIZECP *Problem.* To overcome this problem, we will introduce and reduce from a new two-party communication complexity problem called

[8]This new failure cannot be to the right of τ_i because otherwise when $X_i = 0$ (implying the failure of γ_i^α) and $Y_i = 1$, τ_i has a value of 1 and is disconnected from the root. As explained in Section 5, this prevents us from using the SUM result to determine UNIONSIZE.

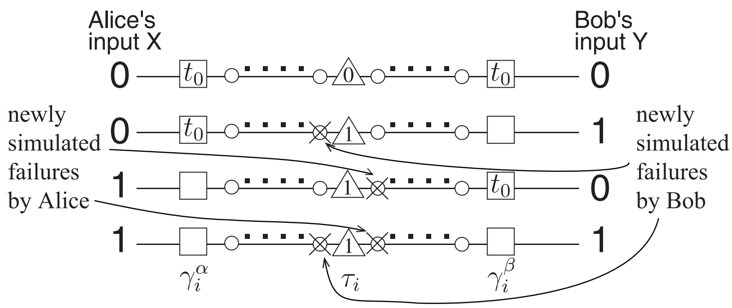

Fig. 5. Why the construction from Section 5 cannot be extended to larger b.

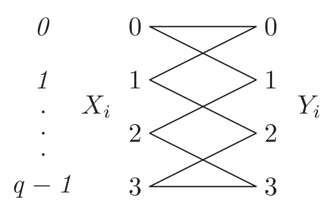

Fig. 6. The cycle promise for $q = 4$.

UnionSizeCP. UnionSizeCP is intuitively UnionSize extended with a novel promise which we call the *cycle promise*. This promise is not constructed ad hoc—rather, we will later see that it can be *derived*. In UnionSizeCP$_{n,q}$ where $q \geq 2$, Alice and Bob respectively have length-n strings X and Y. The characters in the strings are integers in $[0, q-1]$. Let X_i and Y_i denote the ith character of X and Y, respectively. X and Y satisfy the following *cycle promise* where for all i: If $X_i = 0$, then Y_i must be 0 or 1; if $X_i = q-1$, then Y_i must be $q-2$ or $q-1$; if $0 < X_i < q-1$, then Y_i must be $X_i - 1$ or $X_i + 1$. This promise is illustrated in Figure 6 as a bipartite *promise graph*, where values for X_i and Y_i are vertices and two values are connected by an edge if they satisfy the promise. Note that this promise graph is actually a cycle. Same as in UnionSize, the goal in UnionSizeCP is for Alice to determine $|\{i \mid X_i \neq 0 \text{ or } Y_i \neq 0\}|$. When $q = 2$, UnionSizeCP degrades to UnionSize. Later, we will show that different from the earlier naive promise, the cycle promise does not reduce the communication complexity of UnionSizeCP to $O(\log n)$. In our reduction to Sum, the cycle promise will enable us to continuously introduce new failures to block the spreading of spoiled nodes caused by old failures, without disconnecting any node in G with a value of 1. Those newly failed nodes then become spoiled themselves, requiring further failures to be injected, until the end of the simulation.

6.2. Intuitions for Our Reduction from UnionSizeCP to Sum

We now intend to develop some concrete intuitions for our reduction from UnionSizeCP to Sum. Figure 7 illustrates the topology used in our reduction from UnionSizeCP$_{n,q}$, which has n parallel *chains* of nodes, with each chain having $2n+3$ nodes. We connect the first node of each chain directly to a node α, and the last node of each chain directly to a node β.[9] Finally, we connect α and β with a single edge, and let α be the root of the topology. This topology has total $N = \Theta(n^2)$ nodes. As before, Alice (Bob) will simulate a continuously shrinking group of nodes including α (β). As illustrated in Figure 8, the

[9]Using binary trees will not work here. Consequently, here an aggregation round will contain more rounds than in Section 5, and in turn each chain needs to have more nodes.

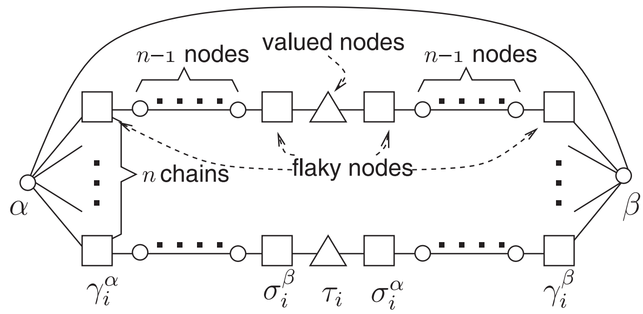

Fig. 7. FT lower bound topology for $b \leq N^{0.25-c}$ or $b \leq \frac{1}{\epsilon^{0.5-c}}$.

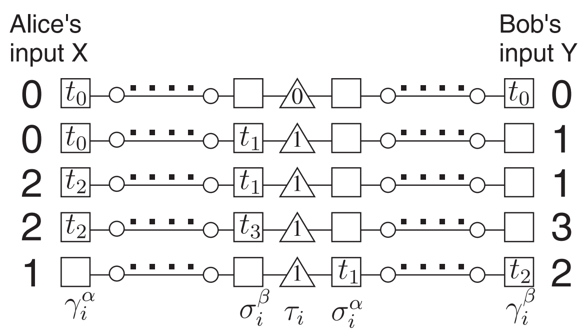

Fig. 8. Values of valued nodes and failure times of flaky nodes, for $q = 4$, $X = 00221$, and $Y = 01132$.

middle node τ_i of the ith chain is a valued node whose value is 1 iff $X_i \neq 0$ or $Y_i \neq 0$. There are four flaky nodes on the chain from left to right: the first node of the chain, the two neighbors of τ_i, and the last node of the chain. We use γ_i^α, σ_i^β, σ_i^α, and γ_i^β to denote these four nodes, respectively. Let $t_j = (j+1)n+1$ for all $0 \leq j \leq q-1$. The flaky node γ_i^α fails at the beginning of round t_{X_i} iff X_i is even, while σ_i^α fails at the beginning of round t_{X_i} iff X_i is odd (Figure 8). Similarly, γ_i^β (σ_i^β) fails at the beginning of round t_{Y_i} iff Y_i is even (odd). Again the failure adversary here is actually oblivious to the Sum protocol, and fails only a vanishingly small fraction (i.e., $o(N)$) of all the nodes in G.

To see intuitively why this reduction works, consider the example in Figure 9. Recall from Section 5.3 the notion of epicenters: A node in the topology is an epicenter for Alice's input X if it is a valued node (or a flaky node) whose value (or failure time) is not uniquely determined by X. Essentially, an epicenter is the source of the spreading of spoiled nodes. When $X_i = 0$, τ_i is an epicenter for Alice and thus Alice simulates the failure of γ_i^α at t_0 to block the influence of such τ_i (i.e., the top/middle scenarios in Figure 9). Next, since the failure of γ_i^α depends on X_i and is not uniquely determined by Y, the node γ_i^α itself now becomes an epicenter for Bob. With the cycle promise and since $X_i = 0$, Y_i must be 0 or 1. If $Y_i = 0$, then Bob does not need to be concerned, since Bob has already simulated the failure of γ_i^β at t_0 and thus blocked the potential influence of γ_i^α (i.e., the top scenario). If $Y_i = 1$ however, Bob needs to simulate the failure of σ_i^β at t_1 (i.e., the middle scenario) to block the influence of γ_i^α. Now σ_i^β again, becomes an epicenter for Alice (i.e., the middle/bottom scenarios). Given the cycle promise and since $Y_i = 1$, we must have $X_i = 0$ or $X_i = 2$. If $X_i = 0$, then Alice has already simulated the failure of γ_i^α at t_0 and has already blocked the potential influence of σ_i^β (i.e., the middle scenario). If $X_i = 2$ however, Alice needs to simulate a new failure of γ_i^α at t_2

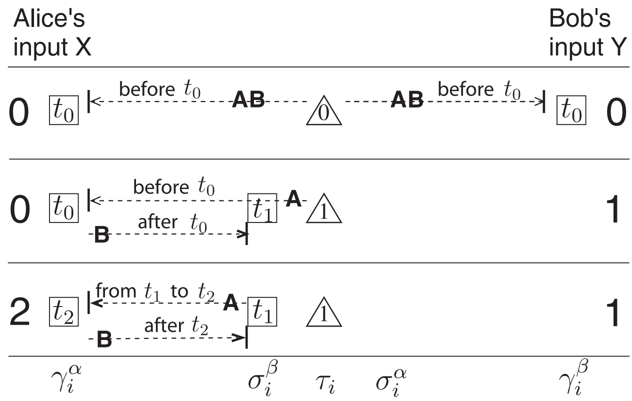

Fig. 9. Failures prevent the spreading of spoiled nodes. Dashed arrows labeled **A** (**B**) indicate the spreading of spoiled nodes for Alice (Bob).

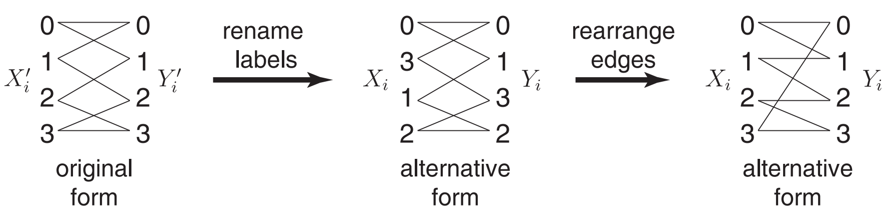

Fig. 10. The alternative form of the cycle promise for $q = 4$, used *only* in Section 6.3 and Appendix C.

(i.e., the bottom scenario). Extending such reasoning can show that, by continuously injecting new failures, we can always manage to block the spreading of spoiled nodes.

Finally, note that the simulation still cannot continue forever. Under the cycle promise, it is possible for $X_i = Y_i = q - 1$. Thus, we need the Sum protocol to stop by round $t_{q-1} - 1$, since otherwise at the beginning of round t_{q-1}, Alice and Bob would simulate failures such that τ_i (with a value of 1) would be disconnected. This means that q needs to be chosen based on the Sum protocol's time complexity b: A larger q is needed when b is larger. Since the communication complexity of UnionSizeCP depends on q (as shown later), as expected, our lower bounds here will be a function of b.

6.3. Communication Complexity of UnionSizeCP

Having provided the overview and intuitions, we are now ready for the formal reasoning. This section proves a lower bound on the communication complexity of UnionSizeCP. Since UnionSizeCP has never been studied, there are no existing results on its communication complexity. Proving these results is thus also a contribution of our work, which may be of independent interest. This lower bound, together with our reduction, enables us to later prove Theorem 6.1 in Section 6.4.

An Alternative Form of the Cycle Promise. For discussion in this section, it will be convenient to consider an alternative form of the cycle promise (Figure 10). Consider any two length-n strings X and Y where the characters in the strings are integers in $[0, q-1]$. X and Y satisfy the alternative form of the cycle promise iff for all i's where $1 \le i \le n$, either $Y_i = X_i$ or $Y_i = (X_i + 1) \bmod q$.

Given X' and Y' satisfying the original cycle promise, Alice and Bob can always locally generate X and Y, such that X and Y satisfy the alternative form of the cycle promise and UnionSizeCP(X, Y) = UnionSizeCP(X', Y'). Specially to do so, Alice sets $X_i = X_i'/2$

for even X'_i and $X_i = q - (X'_i + 1)/2$ for odd X'_i. Bob sets $Y_i = (q - Y'_i/2) \bmod q$ for even Y'_i and $Y_i = (Y'_i + 1)/2$ for odd Y'_i. Clearly, we have $X_i = 0$ iff $X'_i = 0$, and $Y_i = 0$ iff $Y'_i = 0$, which implies that UnionSizeCP(X, Y) = UnionSizeCP(X', Y'). It is easy to verify that X and Y satisfy the alternative form of the cycle promise. Finally, since this mapping from X' (Y') to X (Y) is a bijection, one can also construct a reverse mapping from X (Y) to X' (Y'). Given such mappings in both directions, we trivially know that the communication complexity of UnionSizeCP with the original cycle promise is exactly the same as the communication complexity of UnionSizeCP with the alternative form of the cycle promise.

This section will always consider UnionSizeCP under this alternative form of the cycle promise.

An $O(\frac{n}{q})$ upper bound protocol for UnionSizeCP. On the surface, it may appear that the communication complexity of UnionSizeCP should not be very different from that of UnionSize. This first thought turns out to be incorrect. For $q \le n$, the following presents a $O(\frac{n}{q})$ upper bound protocol for $\text{NFT}_0(\text{UnionSizeCP}_{n,q}, \text{poly}(n))$, implying that its communication complexity drops at least linearly with $\frac{1}{q}$.

In this protocol, given input X to Alice, let j $(0 \le j \le q-1)$ be the integer with the smallest occurrence count in X. (If there are multiple such j's, simply pick an arbitrary one.) Alice first sends Bob the value of j and the set $Z = \{i \mid X_i = j\}$. This takes at most $O(\log q + \frac{n}{q}\log n)$ bits in one round, or $O(\frac{\log q}{\log n} + \frac{n}{q})$ bits in poly(n) rounds [Impagliazzo and Williams 2010]. For $q \le n$, this becomes $O(\frac{n}{q})$. Now we only need to worry about indices not in the set. For those indices, the promise graph (Figure 10) degrades to a chain, since two edges are removed from the cycle. The problem becomes easy to solve under the degraded chain promise.

Specifically, Bob will now know both X_i and Y_i for all $i \in Z$. If $j = 0$ or $j = q-1$, then Bob can already determine $\{i \mid X_i = Y_i = 0\}$, and can locally compute the final result. If $j \ne 0$ and $j \ne q-1$, then for any index $i' \notin Z$, Bob knows that $X_{i'} \ne j$. Thus, if $Y_{i'} = j+1$, then $X_{i'}$ must be $j+1$ as well. This observation enables the following trick. Alice locally calculates $h_A = |\{i' \mid i' \notin Z \text{ and } j+1 \le X_{i'} \le q-1\}|$ and sends h_A to Bob, using $\log n$ bits in one round, or $O(1)$ bits in poly(n) rounds [Impagliazzo and Williams 2010]. Bob calculates $h_B = |\{i' \mid i' \notin Z \text{ and } (j+1 \le Y_{i'} \le q-1 \text{ or } Y_{i'} = 0)\}|$. Given that $X_{i'} \ne j$ for $i' \notin Z$, one can easily verify from the cycle promise that $h_B - h_A$ is exactly the total number of indices i where $X_i = Y_i = 0$. The result for UnionSizeCP(X, Y) is thus $n - (h_B - h_A)$.

Lower bounds of $\Omega(n/(q^2 \log n))$ and $\Omega(1/(\epsilon q^2 \log n))$ for UnionSizeCP. To lower bound UnionSizeCP's communication complexity, we find that the cycle promise makes it challenging to apply classic arguments based on rectangles [Kushilevitz and Nisan 1996].[10] But we also find that UnionSizeCP is rather amenable to information cost arguments [Bar-Yossef et al. 2004], which lead to the following theorem. We leave its proof, which largely uses standard techniques, to Appendix C.

Theorem 6.2. *We have*

$$\text{NFT}_0(\text{UnionSizeCP}_{n,q}, O(\text{poly}(n))) = \Omega\left(\frac{n}{q^2 \log n}\right),$$

$$\text{NFT}_{\epsilon,\frac{1}{5}}(\text{UnionSizeCP}_{n,q}, O(\text{poly}(n))) = \Omega\left(\frac{1}{\epsilon q^2 \log n}\right), \quad \textit{for } \epsilon = \Omega\left(\frac{1}{\sqrt{n}}\right).$$

[10] Leveraging some strong results on the sperner capacity of the cyclic q-gon [Blokhuis 1993], we managed to obtain some results on NFT_0(UnionSizeCP), but not on $\text{NFT}_{\epsilon,\delta}$(UnionSizeCP).

6.4. Proof for Theorem 6.1

Having obtained a lower bound on UNIONSIZECP's communication complexity, we now move on to prove Theorem 6.1 using the framework established in Section 5.3. The proof follows a rather similar structure to the proof in Section 5.4. Specifically, Lemma 6.3 first proves that in the construction in Section 6.2, α (β) will always remain unspoiled for Alice (Bob) in round R, where R is large enough for the SUM oracle protocol to have terminated. Lemma 6.4 then proves the reduction from UNIONSIZECP to SUM, while taking care of those values of N that does not correspond to the topology in Figure 7 for any integer n. Combining this lemma with the earlier lower bound on UNIONSIZECP directly gives us Theorem 6.1.

LEMMA 6.3. *Consider the topology, valued nodes (with their values), and flaky nodes (with their failure times), as constructed in Section 6.2. Under this construction and under $R = qn$, for all possible input X of Alice's, we have $\alpha \in \overline{S}_{A,X}(R)$. Similarly, for all possible input Y of Bob's, we have $\beta \in \overline{S}_{B,Y}(R)$.*

PROOF. Without loss of generality, we prove $\alpha \in \overline{S}_{A,X}(R)$. We exhaustively consider all the epicenters with respect to Alice's input X. First, if $X_i = 0$ (implying Y_i must be 0 or 1), then τ_i, σ_i^β, and γ_i^β are the only epicenters on the ith chain. The spoil distance from all these epicenters to α is infinite, since γ_i^α fails stably at the beginning of round t_0 and thus blocks the only possible spoil path.

Next if $X_i = q-1$, then Y_i must be $q-1$ or $q-2$. If $q-1$ is even, then σ_i^β (potentially occurring at round t_{q-2}) and γ_i^β (potentially occurring at round t_{q-1}) are the only epicenters on the ith chain. Again, γ_i^α fails stably at the beginning of round t_{q-1} and thus blocks the only possible spoil path from those two epicenters to α. If $q-1$ is odd, then σ_i^β (potentially occurring at round t_{q-1}) and γ_i^β (potentially occurring at round t_{q-2}) are the only epicenters on the ith chain. Since σ_i^α fails stably at the beginning of round t_{q-1}, the only possible spoil path from γ_i^β to α is blocked. The epicenter of σ_i^β has an occurrence time of $t_{q-1} = qn+1 > R$. Thus, it can never cause α to be spoiled in round R.

Finally, if X_i is even and $0 < X_i < q-1$, then Y_i must be odd and thus σ_i^β is the only epicenter on the ith chain, with an occurrence time of round t_{X_i-1}. (Recall that the occurrence time is the earliest possible failure time.) But since X_i is even, γ_i^α fails stably at the beginning of round t_{X_i}. This failure blocks the only possible spoil path from σ_i^β to α. The case where X_i is odd and $0 < X_i < q-1$ is similar. □

LEMMA 6.4. *Consider any $b \geq 1$ and any sufficiently large integer N. Let n be the largest integer such that $2n^2 + 3n + 2 \leq N$ and let $q = 5b$. There exists a connected topology G with N nodes, such that:*

$$\mathrm{FT}_0(\textsc{Sum}_G, b) \geq \frac{1}{2}\mathrm{NFT}_0(\textsc{UnionSizeCP}_{n,q}, bN),$$

$$\mathrm{FT}_{\epsilon,\frac{1}{5}}(\textsc{Sum}_G, b) \geq \frac{1}{2}\mathrm{NFT}_{\epsilon,\frac{1}{5}}(\textsc{UnionSizeCP}_{n,q}, bN).$$

PROOF. We first prove the lemma for $N = 2n^2 + 3n + 2$. We construct G as in Section 6. Let $R = t_{q-1} - 1 = qn = 5bn$. Under our constructed failure adversary, let $\lambda = \max_{G' \in \mathcal{G}} \Lambda(G')$ where $\mathcal{G}$ is the set of topologies that have ever appeared during the execution. We first prove that $R \geq b\lambda$. It is easy to verify that even if we pessimistically assume that all messages to α and β have to be sent sequentially one by one, we still have $\lambda \leq (2n+3)+(n+1)+n = 4n+4$. Thus, we have $R = 5bn \geq b\lambda$ for sufficiently large n.

Next we reduce the UnionSizeCP$_{n,q}$ problem (in the synchronous rounds setting) to Sum, which will prove the lemma for $N = 2n^2 + 3n + 2$. This part of the proof is exactly the same as in the proof of Lemma 5.4, after substituting Lemma 5.3 with Lemma 6.3. Thus, we do not repeat it here.

Finally, we still need to prove the lemma for $N > 2n^2 + 3n + 2$. Let $N_1 = 2n^2 + 3n + 2$. We first construct a connected topology G_1 with N_1 nodes as in Section 6. Next, we add $N_2 = N - N_1 = O(n)$ extra nodes to G_1 to obtain the topology G with N nodes. Those N_2 nodes will always have a value of 0 and will never fail. Same as in the proof of Lemma 5.4, We want to add those N_2 nodes carefully so that (i) Lemma 6.3 continues to hold, and (ii) the length of an aggregation round is not affected. To do so, we partition the N_2 nodes into n groups of size $O(1)$. All nodes in the ith group have a degree of 1 and directly attach to the $\frac{n}{2}$th node (from left to right) on the ith chain in G_1.

It is trivial to verify that Lemma 6.3 continues to hold after adding those N_2 nodes in this way. For the length of an aggregation round, note that each group has only $O(1)$ nodes and thus all nodes in the group will finish sending their aggregation messages by round $O(1)$. On the other hand, the $\frac{n}{2}$th node on a chain will not receive any other aggregation messages before round $\frac{n}{2} - 1$, under any G' in $\mathcal{G}$ where $\mathcal{G}$ is the set of topologies that have ever appeared during the execution. Same as in the proof of Lemma 5.4, these properties ensure that our earlier proof for $N = 2n^2 + 3n + 2$ carries over to $N > 2n^2 + 3n + 2$ without modification. □

The following proof for Theorem 6.1 follows naturally from the Lemma 6.4:

Proof for Theorem 6.1. Since the theorem trivially holds for $b > N$, we only need to prove the theorem for $b \leq N$. For any sufficiently large N, consider the N-node connected topology G as constructed by Lemma 6.4. We trivially have

$$\mathrm{FT}_0(\textsc{Sum}_N, b) \geq \mathrm{FT}_0(\textsc{Sum}_G, b) \geq \frac{1}{2}\mathrm{NFT}_0(\textsc{UnionSizeCP}_{n,q}, bN),$$
$$\mathrm{FT}_{\epsilon,\frac{1}{5}}(\textsc{Sum}_N, b) \geq \mathrm{FT}_{\epsilon,\frac{1}{5}}(\textsc{Sum}_G, b) \geq \frac{1}{2}\mathrm{NFT}_{\epsilon,\frac{1}{5}}(\textsc{UnionSizeCP}_{n,q}, bN).$$

By Lemma 6.4, in these inequalities, $q = 5b$ and n is the largest integer satisfying $2n^2 + 3n + 2 \leq N$. Thus, we have $n = \Theta(\sqrt{N})$. Applying Theorem 6.2 gives

$$\begin{aligned}
\mathrm{FT}_0(\textsc{Sum}_N, b) &\geq \frac{1}{2}\mathrm{NFT}_0(\textsc{UnionSizeCP}_{n,q}, bN) \\
&= \Omega\left(\frac{n}{q^2 \log n}\right) = \Omega\left(\frac{\sqrt{N}}{b^2 \log N}\right) \\
\mathrm{FT}_{\epsilon,\frac{1}{5}}(\textsc{Sum}_N, b) &\geq \frac{1}{2}\mathrm{NFT}_{\epsilon,\frac{1}{5}}(\textsc{UnionSizeCP}_{n,q}, bN) \\
&= \Omega\left(\frac{1}{\epsilon q^2 \log n}\right) = \Omega\left(\frac{1}{\epsilon b^2 \log N}\right), \quad \text{for } \epsilon = \Omega\left(\frac{1}{\sqrt[4]{N}}\right).
\end{aligned}$$ □

7. THE FUNDAMENTAL ROLES OF CYCLE PROMISE AND UNIONSIZECP

Our reduction from UnionSizeCP so far has led to the exponential gap result for Sum, when $b \leq N^{0.25-c}$ or $b \leq \frac{1}{\epsilon^{0.5-c}}$ for any positive constant $c < 0.25$. This restriction on b comes from the $\frac{1}{q^2}$ term in the lower bound of the communication complexity of UnionSizeCP. Our upper bound on UnionSizeCP indicates that such a polynomial dependency on $\frac{1}{q}$ is unavoidable because of the cycle promise. It is thus natural to ask: Can we reduce from problems without promises? Or can we reduce from problems with a different promise, to weaken the polynomial dependency on $\frac{1}{q}$ to $\log \frac{1}{q}$? For any possible *oblivious reduction* (defined next) from any two-party communication complexity problem Π to SUM, this section answers these questions in the negative. Specifically, we will prove the *completeness* of UNIONSIZECP in the sense that such a Π can always be reduced to UNIONSIZECP and must have a communication complexity no larger than that of $\text{UNIONSIZECP}_{N,\lfloor\sqrt{b/3}\rfloor}$. Thus, any FT lower bound on SUM, obtained in such a way via Π, must contain some polynomial term of $\frac{1}{b}$. Overcoming this polynomial term in the lower bound might still be possible, but one would have to resort to methods other than oblivious reductions from two-party problems. Our proof also (implicitly) shows that the cycle promise can be derived and that the promise likely plays a fundamental role in reasoning about many functions beyond SUM.

In the following, Section 7.1 defines the notion of oblivious reductions. Section 7.2 presents the completeness theorem and its proof overview. Finally, Section 7.3 proves the theorem.

7.1. Oblivious Reductions

Consider any two-party communication complexity problem Π, where (only) Alice aims to learn $\Pi(X, Y)$. In a (general) reduction from Π to SUM, Alice and Bob are given some black-box *oracle* fault-tolerant protocol for SUM, and they are supposed to use this oracle to solve Π with any given input pair (X, Y). Since the (global) oracle protocol is distributed, it will be convenient to imagine that each node in the topology has its own oracle protocol, and invoking these protocols in a "consistent" fashion will enable the root to produce a meaningful result.

In an *oblivious reduction* to SUM, there is some fixed topology G and for each (X, Y) pair, there exists some *reference setting* specifying the value and failure time of each node in G. The reference settings are oblivious to the oracle. As explained in Section 5, a reference setting here should not fail or disconnect nodes with a value of 1. The zero-error SUM result in the reference setting should be the same as $\Pi(X, Y)$, so we can directly use it for solving Π. The reduction protocol is required to be oblivious as well. Specifically, Alice and Bob first pick a (public) random string. Next before invoking the oracle and purely based on X (Y), Alice (Bob) decides for each node in G, exactly up to which round she (he) will invoke the oracle. Note that to invoke the oracle for a certain round, Alice/Bob needs to invoke the oracle for all previous rounds as well. Alice (Bob) also decides the (initial) value of each node for which she (he) will invoke the oracle for at least one round. Requiring Alice and Bob to make these decisions beforehand is the most important aspect of oblivious reductions. We define the *reference execution* for (X, Y) to be the (global) oracle's execution under the reference setting for (X, Y) and under the chosen random string. To enable the root to generate a meaningful result, we require that the initial value, incoming messages, and coin flips fed by Alice/Bob into the oracle protocol on a node be the same as those fed into that node's oracle in the reference execution for (X, Y). Furthermore, after a node has failed in the reference execution, Alice/Bob must not invoke that node's oracle any more (since that node can no longer help out). Finally, there are two special nodes α and β in G, such that Alice and Bob will always invoke the oracle on α and β (respectively) until the root generates a result. Here, α must be the root of G,[11] while β can be any other node. During the reduction, Alice (Bob) may only send to the other party all those outgoing messages generated by the oracle invocation on node α (β). This allows the establishment of a simple factor-2 relation between the communication complexity of Π and SUM.

[11]This is largely for clarity, and can be relaxed if desired.

Our previous reductions from UnionSize and UnionSizeCP to Sum are both oblivious reductions. Besides those two specific instances, the broad class of oblivious reductions further captures reductions from any two-party problem Π with any promise, using any topology G with any proper reference settings.

7.2. The Completeness of UnionSizeCP

We now present a strong result on the completeness of UnionSizeCP, in oblivious reductions.

Theorem 7.1. *Consider any two-party communication complexity problem* Π *that can be obliviously reduced to* Sum *for some topology* G *with* N *nodes, with the* Sum *oracle protocol having a time complexity of up to* b *aggregation rounds where* $b \geq 12$*. For all* $t \geq 1$*, we have*

$$\mathcal{R}_0^{\text{syn}}(\Pi, t) \leq \mathcal{R}_0^{\text{syn}}\big(\text{UnionSizeCP}_{N,\lfloor\sqrt{b/3}\rfloor}, t\big),$$
$$\mathcal{R}_{\epsilon,\delta}^{\text{syn}}(\Pi, t) \leq \mathcal{R}_{\epsilon,\delta}^{\text{syn}}\big(\text{UnionSizeCP}_{N,\lfloor\sqrt{b/3}\rfloor}, t\big).$$

The following gives an overview of the proof of this theorem, and also defines some useful concepts. Let $\mathcal{X}$ be Alice's input domain in Π, and $\mathcal{Y}$ be Bob's. Let $\mathcal{L} \subseteq \mathcal{X} \times \mathcal{Y}$ be the set of all valid input pairs, given the promise in Π. If Π has no promise, then $\mathcal{L} = \mathcal{X} \times \mathcal{Y}$. Given $(X, Y) \in \mathcal{L}$, an oblivious reduction has a reference setting specifying the value of each node in G. For any node τ where $\tau \neq \alpha$ and $\tau \neq \beta$, we define τ's *(value) assignment graph* to be the bipartite graph where $\mathcal{X} \cup \mathcal{Y}$ are vertices and an edge (X, Y) exists iff $(X, Y) \in \mathcal{L}$. In addition, each edge (X, Y) has a binary label which is the value of τ in the reference setting for (X, Y).

We will prove that it is always possible to partition the vertices in τ's assignment graph into $2b'$ (where $b' = \lfloor\sqrt{b/3}\rfloor \geq 2$) disjoint subsets with strong properties as illustrated in Figure 11. Intuitively, this is because otherwise the reference setting for some input pair would need to have so many failures in G such that τ (with a value of 1) would be disconnected from the root. Those failures are needed to ensure that Alice (Bob) can invoke the oracle on α (β) throughout the execution.

At this point, we already have something close to the cycle promise—if we view each subset as a super vertex, then all the $2b'$ super vertices form a subgraph of a length-$2b'$ cycle. It is now possible to reduce Π to $\text{UnionSizeCP}_{N,b'}$, by mapping an input X for Π to an input X' for UnionSizeCP as following: Each τ in G corresponds to a unique i ($1 \leq i \leq N-2$), and X'_i is set to be the index of the subset in τ's assignment graph to which X belongs. Finally, X'_{N-1} is set to be the (initial) value of α in the given oblivious reduction, which can be obtained purely based on X. X'_N is set to be 0. The conversion from Y to Y' is similar, with $Y'_{N-1} = 0$ and Y'_N being the value of β.

7.3. Proof for Theorem 7.1

This section will prove Theorem 7.1, after formalizing various concepts and proving a number of lemmas. We first formalize the notion of a node τ being disconnected from the root in a simulation. Consider any input pair $(X, Y) \in \mathcal{L}$ and the corresponding reference setting in the oblivious reduction from Π to Sum. Define $\Phi(X, Y)$ to be the execution of the Sum oracle protocol under that reference setting and under the (public) random string chosen by Alice and Bob in the oblivious reduction. For a given node τ in G where $\tau \neq \alpha$ and $\tau \neq \beta$, we say that τ is *disconnected* from the root in an execution $\Phi(X, Y)$, if either τ fails during $\Phi(X, Y)$, or the root and τ are no longer in the same connected component at the end of $\Phi(X, Y)$. We have the following trivial lemma.

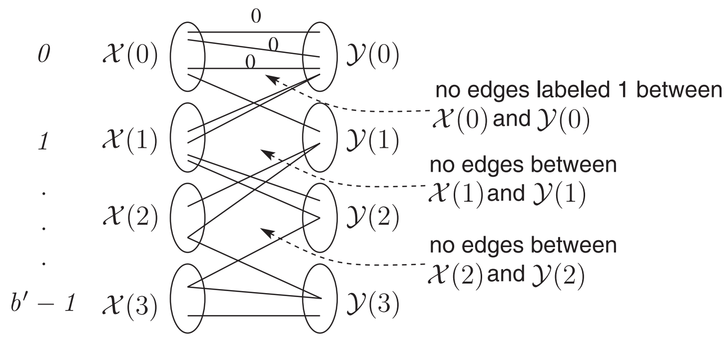

Fig. 11. Example assignment graph for a given node τ and for $b' = 4$.$\mathcal{X}(0), \mathcal{Y}(0), \ldots, \mathcal{X}(3)$, and $\mathcal{Y}(3)$ are the 8 subsets, which may have different sizes and different numbers of incidental edges. All edges without labels indicated have a label of 1. We will later prove that in all assignment graphs, there must be no edges labeled 1 between $\mathcal{X}(0)$ and $\mathcal{Y}(0)$, and no edges between $\mathcal{X}(i)$ and $\mathcal{Y}(i)$ for all $1 \le i \le b' - 2$.

LEMMA 7.2. *For any $(X, Y) \in \mathcal{L}$, if τ has a value of 1 under the reference setting for (X, Y), then in $\Phi(X, Y)$, τ is never disconnected from the root.*

PROOF. Trivially follows from the requirement on the reference settings in oblivious reductions. □

Next consider the assignment graph of any node τ where $\tau \neq \alpha$ and $\tau \neq \beta$. Let $b' = \lfloor\sqrt{b/3}\rfloor$, which implies $b' \ge 2$ since $b \ge 12$. We partition the vertices of τ's assignment graph into $2b'$ disjoint subsets of $\mathcal{X}(0), \mathcal{Y}(0), \ldots, \mathcal{X}(b'-1)$, and $\mathcal{Y}(b'-1)$ in the following way. For all integer $i \in [0, b'-3]$, we recursively define:

$$\begin{aligned}
\mathcal{X}(0) &= \{X \mid \tau(X, Y) = 0 \text{ for some } (X, Y) \in \mathcal{L}\},\\
\mathcal{Y}(0) &= \{Y \mid \tau(X, Y) = 0 \text{ for some } (X, Y) \in \mathcal{L}\},\\
\mathcal{X}(i+1) &= \left(\mathcal{X} \setminus \cup_{j=0}^{i}\mathcal{X}(j)\right) \cap \{X \mid \tau(X, Y) = 1 \text{ for some } (X, Y) \in \mathcal{L} \text{ where } Y \in \mathcal{Y}(i)\},\\
\mathcal{Y}(i+1) &= \left(\mathcal{Y} \setminus \cup_{j=0}^{i}\mathcal{Y}(j)\right) \cap \{Y \mid \tau(X, Y) = 1 \text{ for some } (X, Y) \in \mathcal{L} \text{ where } X \in \mathcal{X}(i)\},\\
\mathcal{X}(b'-1) &= \mathcal{X} \setminus \cup_{j=0}^{b'-2}\mathcal{X}(j),\\
\mathcal{Y}(b'-1) &= \mathcal{Y} \setminus \cup_{j=0}^{b'-2}\mathcal{Y}(j).
\end{aligned}$$

Figure 11 illustrated these sets for a given τ for $b' = 4$. Intuitively, any vertex with some 0-labeled incidental edge belongs to $\mathcal{X}(0)$ or $\mathcal{Y}(0)$. Thus, an edge (X, Y) always has a label of 1 if $X \notin \mathcal{X}(0)$ or $Y \notin \mathcal{Y}(0)$. We say that there are edges between two sets $\mathcal{X}(i)$ and $\mathcal{Y}(j)$ iff there exists some edge (X, Y) in the assignment graph for some $X \in \mathcal{X}(i)$ and some $Y \in \mathcal{Y}(j)$.

We now prove the following lemma regarding τ's assignment graph, showing that there must be no edges labeled 1 between $\mathcal{X}(0)$ and $\mathcal{Y}(0)$, and no edges between $\mathcal{X}(i)$ and $\mathcal{Y}(i)$ for all $1 \le i \le b'-2$. (It is still possible for edges to exist between $\mathcal{X}(b'-1)$ and $\mathcal{Y}(b'-1)$.) Intuitively, this lemma holds because if there existed such an edge connecting Alice's input X and Bob's input Y, then the reference setting for (X, Y) would need to have so many failures in G such that τ would be disconnected from the root. Those failures are needed to ensure that Alice (Bob) can invoke the oracle on α (β) throughout the execution (i.e., to ensure that α and β remain unspoiled). On the other hand, τ must have a value of 1 in the reference setting for such (X, Y). This contradicts with Lemma 7.2.

LEMMA 7.3. *In any oblivious reduction from* Π *to* SUM*, consider any node* τ *in* G *where* $\tau \neq \alpha$ *and* $\tau \neq \beta$*. In* τ*'s assignment graph, there must be no edges labeled 1 between* $\mathcal{X}(0)$ *and* $\mathcal{Y}(0)$*, and no edges between* $\mathcal{X}(i)$ *and* $\mathcal{Y}(i)$ *for all* $1 \leq i \leq b' - 2$*.*

PROOF. The lemma trivially holds for $b' = 2$, and thus we only consider $b' \geq 3$, or equivalently $b \geq 27$. We prove via a contradiction and assume that the lemma does not hold. Intuitively, we will construct a path in τ's assignment graph (not a path in G) where vertices on the path are individual inputs. Since it is a path in the assignment graph, by the definition of τ's assignment graph, any two adjacent inputs on that path must form a valid input pair in $\mathcal{L}$. The path will start from some vertex in $\mathcal{X}(0)$ and end with some vertex in $\mathcal{Y}(0)$. Furthermore, all edges on the path have a label of 1, and the path has no more than $2(b' - 1) - 1$ hops. These properties can be later used to find a contradiction.

We now present the formal arguments. If the lemma does not hold, then in the assignment graph of τ, either there is an edge labeled 1 between $\mathcal{X}(0)$ and $\mathcal{Y}(0)$, or there is an edge (which must have a label of 1) between $\mathcal{X}(j)$ and $\mathcal{Y}(j)$ for some $j \in [1, b' - 2]$. We claim that in either case, we can find in the assignment graph a path $X^{(0)}, Y^{(1)}, X^{(1)}, Y^{(2)}, \ldots, X^{(k)}, Y^{(k+1)}$ for some $k \in [0, b' - 2]$, such that:

—$X^{(i)} \in \mathcal{X}$ for $0 \leq i \leq k$, and $Y^{(i)} \in \mathcal{Y}$ for $1 \leq i \leq k + 1$,
—$X^{(0)} \in \mathcal{X}(0)$ and $Y^{(k+1)} \in \mathcal{Y}(0)$, and
—all edges in the path have a label of 1.

If there is an edge labeled 1 between $\mathcal{X}(0)$ and $\mathcal{Y}(0)$, we simply set $k = 0$ and let $X^{(0)}$ and $Y^{(1)}$ be the two endpoints of that edge, respectively. Our claim then trivially holds. If there is an edge (X, Y) where $X \in \mathcal{X}(j)$ and $Y \in \mathcal{Y}(j)$ for some $j \in [1, b' - 2]$, then we set $k = j$. First consider the case where k is even. By the construction of the assignment graph, X must be connected with some vertex in $\mathcal{Y}(j - 1)$, and that vertex must be connected to some vertex in $\mathcal{X}(j - 2)$, and so on. Since k is even, there must be some path (with exactly k hops) in the assignment graph from X to some $X^{(0)} \in \mathcal{X}(0)$. Similarly, there must be some path (with exactly k hops) in the assignment graph from Y to some $Y^{(k+1)} \in \mathcal{Y}(0)$. These two paths, together with the edge between X and Y, exactly form the path needed by our claim. The case for odd k is similar.

Given this path $X^{(0)}, Y^{(1)}, X^{(1)}, Y^{(2)}, \ldots, X^{(k)}, Y^{(k+1)}$ (satisfying all these properties), we define $\mathcal{I}$ (where $\mathcal{I} \subseteq \mathcal{L}$) to be the set of input pairs (X, Y) such that (X, Y) is an edge in that path. We also call $\mathcal{I}$ as the *problematic input set*. By construction of $\mathcal{I}$, we know that τ has a value of 1 in the reference setting for any $(X, Y) \in \mathcal{I}$. Lemma 7.4 next proves that for all $(X, Y) \in \mathcal{I}$, τ is disconnected from the root by the end of the execution of $\Phi(X, Y)$. This leads to a contradiction with Lemma 7.2. □

LEMMA 7.4. *Suppose* $b \geq 27$*. Consider the problematic input set* $\mathcal{I}$ *as constructed in the proof of Lemma 7.3. For all* $(X, Y) \in \mathcal{I}$*,* τ *is disconnected from the root by the end of the execution of* $\Phi(X, Y)$*.*

PROOF. This technical lemma is challenging to prove. We leave its elementary but involved proof to Appendix D. □

We can now use Lemma 7.3 to prove Theorem 7.1.

PROOF FOR THEOREM 7.1. By the condition in the theorem, there exists some oblivious reduction $\mathcal{P}$ from Π to SUM for some topology G. Let the $N - 2$ nodes other than α and β in G be $\tau_1, \tau_2, \ldots, \tau_{N-2}$. Consider any input pair $(X, Y) \in \mathcal{L}$. Let $\tau_i(X, Y)$, $\alpha(X, Y)$, and $\beta(X, Y)$ be the values of τ_i, α, and β in $\mathcal{P}$'s reference setting for (X, Y), respectively. Since the zero-error SUM result under the reference setting must be the same as $\Pi(X, Y)$ and since the reference setting never fails or disconnects nodes with a value of 1, we have:

$$\Pi(X, Y) = \alpha(X, Y) + \beta(X, Y) + \sum_{i=1}^{N-2} \tau_i(X, Y).$$

We intend to reduce Π to UNIONSIZECP$_{N,b'}$. To do so, Alice converts her input X for Π to a corresponding input X' of length N for UNIONSIZECP in the following way, using only local knowledge. Let X'_i be the ith character in the string X'. Alice sets the last character X'_N to be 0. Alice next leverages $\mathcal{P}$ to obtain the value of $\alpha(X, Y)$, without communicating with Bob. To do so, Alice invokes $\mathcal{P}$ using X and then stops once $\mathcal{P}$ needs to invoke the SUM oracle protocol (which Alice does not have). Doing so clearly does not incur any communication. By definition of oblivious reductions, $\mathcal{P}$ at this point must have decided, based on X, the (initial) value of node α. Furthermore, this value must be the same as $\alpha(X, Y)$. Alice now obtains $\alpha(X, Y)$ purely based on local knowledge. Alice sets X'_{N-1} to this value. Next, Alice needs to determine X'_i for $1 \le i \le N-2$. By definition of oblivious reductions, $\mathcal{P}$ needs to specify the reference setting corresponding to each input pair $(X, Y) \in \mathcal{L}$. Using such information in $\mathcal{P}$, Alice can determine the assignment graph of τ_i for $1 \le i \le N-2$. In τ_i's assignment graph, Alice's current input X must belong to exactly one of the subsets of vertices. Let $\mathcal{X}(j)$ be the subset to which X belongs. Alice then sets $X'_i = j$.

Bob constructs input Y' of length N for UNIONSIZECP similarly, using only his local knowledge of Y. Specifically, Bob sets $Y'_{N-1} = 0$ and $Y'_N = \beta(X, Y)$. Same as earlier, Bob can obtain $\beta(X, Y)$ via $\mathcal{P}$, without communicating with Alice. Next for each i where $1 \le i \le N-2$, Bob sets Y'_i to be j where $\mathcal{Y}(j)$ is the subset to which Y belongs to, in τ_i's assignment graph.

We next show that X' and Y' are strings satisfying the cycle promise with $q = b'$. First, for $i = N-1$ or N, obviously X'_i and Y'_i satisfy the cycle promise. Next consider any $i \in [1, N-2]$ and τ_i's assignment graph. Clearly, X'_i and Y'_i are integer in $[0, b'-1]$ for all such i. By construction of the assignment graph, we know that there are

—no edges between $\mathcal{X}(0)$ and $\mathcal{Y}(j)$ for all $j \ge 2$,
—no edges between $\mathcal{X}(b'-1)$ and $\mathcal{Y}(j)$ for all $j \le b'-3$, and
—no edges between $\mathcal{X}(j_1)$ and $\mathcal{Y}(j_2)$ for all $1 \le j_1 \le b'-2$ and $|j_1 - j_2| \ge 2$.

Furthermore, Lemma 7.3 shows that there are no edges between $\mathcal{X}(j)$ and $\mathcal{Y}(j)$ for all $1 \le j \le b'-2$. Since $(X, Y) \in \mathcal{L}$, there must exist an edge between X and Y in τ_i's assignment graph. Thus, X'_i and Y'_i must satisfy the cycle promise.

Finally, consider any given protocol for UNIONSIZECP$_{N,b'}$. Alice and Bob invoke that protocol using X' and Y' as inputs, respectively. We claim that UNIONSIZECP(X', Y') can be used directly as the result of $\Pi(X, Y)$, since

$$\begin{aligned}
&\text{UNIONSIZECP}(X', Y') \\
&\quad = |\{i \mid (1 \le i \le N) \text{ and } (X'_i \ne 0 \text{ or } Y'_i \ne 0\})| \\
&\quad = \alpha(X, Y) + \beta(X, Y) + |\{i \mid (1 \le i \le N-2) \text{ and } (X \notin \mathcal{X}(0) \text{ for } \tau_i \text{ or } Y \notin \mathcal{Y}(0) \text{ for } \tau_i)\}| \\
&\quad = \alpha(X, Y) + \beta(X, Y) + |\{i \mid 1 \le i \le N-2 \text{ and } \tau_i(X, Y) = 1\}| \\
&\qquad \text{(by construction of the assignment graph and Lemma 7.3)} \\
&\quad = \alpha(X, Y) + \beta(X, Y) + \sum_{i=1}^{N-2} \tau_i(X, Y) = \Pi(X, Y).
\end{aligned}$$

In this derivation, we have leveraged Lemma 7.3, which shows that there are no edges labeled 1 between $\mathcal{X}(0)$ and $\mathcal{Y}(0)$. Together with the construction of the assignment graph, this means that $\tau_i(X, Y) = 1$ if and only if in τ_i's assignment graph, $X \notin \mathcal{X}(0)$ or $Y \notin \mathcal{Y}(0)$. □

8. LOWER BOUNDS ON FT COMMUNICATION COMPLEXITY OF SUM FOR ALL *b*

Our previous FT lower bounds from Section 6 become trivial when $b > N^{0.25}$ or $b > \frac{1}{\epsilon^{0.5}}$. Section 7 has suggested that such limitation might be inherent in the approach used in Section 6. This section uses a different approach to obtain logarithmic FT lower bounds for such b, which is more than exponentially far away from the corresponding $O(1)$ NFT upper bounds for such b. Specifically, this section aims to eventually prove the following theorem.

THEOREM 8.1. *For any $b \geq 1$, we have*

$$\mathrm{FT}_0(\mathrm{SUM}_N, b) = \Omega(\log N),$$
$$\mathrm{FT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}_N, b) = \Omega\left(\log \frac{1}{\epsilon}\right), \quad \text{for } \epsilon = \Omega\left(\frac{1}{N}\right).$$

This theorem establishes the FT lower bounds for the right-most region of b in Figure 1 (i.e., for $b > N^{0.25-c}$ or $b > \frac{1}{\epsilon^{0.5-c}}$). While the theorem applies to all $b \geq 1$, the results in the previous sections are stronger when $b \leq N^{0.25-c}$ or $b \leq \frac{1}{\epsilon^{0.5-c}}$. Also note that under sufficiently large b, a message of any given size can be encoded using a single bit. Hence, $\Omega(\log N)$ and $\Omega(\log \frac{1}{\epsilon})$ actually lower bound the number of messages, and the theorem can only be proved by reasoning about the number of messages.

In the following, Section 8.1 first gives some intuitions and reveals the key challenge in proving the theorem. Next, Section 8.2 describes the topology and the failure adversary used for proving the theorem. Section 8.3 defines a simple single-player *probing game*, and then proves a connection (or a reduction in the general sense) between the probing game and the SUM problem. This is the key step since it enables us to focus on lower bounding the "performance" of the probing game, which is much easier to study than the SUM problem. Section 8.4 then proves a lower bound on the "performance" of the probing game. This lower bound, together with the connection established earlier, enables us to prove Theorem 8.1 finally in Section 8.5.

8.1. Obtaining Some Intuitions under the Gossip Assumption

We first provide some intuitions for Theorem 8.1 under a strong *gossip assumption*. Doing so also helps to reveal the key technical challenges to be addressed by our proof later.

Under the *gossip assumption*, the root computes the sum by explicitly collecting from each node a gossip containing its value. We will intuitively show that to do so, some node will need to send $\Omega(\log N)$ messages, and hence $\Omega(\log N)$ bits even if the gossips can be fully aggregated/compressed. Here the lower bound topology will be an N-node clique with one of nodes being the root (Figure 12). Imagine for now that the adversary can fail edges in this topology, and further there is never more than one node sending messages in a round. These assumptions can be easily removed later once we insert some dummy nodes into each edge. Our adaptive adversary waits until exactly $\frac{N-1}{2}$ nonroot nodes have sent a message (e.g., nodes 1 and 2 in Figure 12). Call these $\frac{N-1}{2}$ nodes as *marked* nodes. The adversary then fails enough edges so that each unmarked nonroot node (e.g., node 3) is paired up with a marked node (e.g., node 1) and the marked node is the only gateway for the unmarked node to reach the root. Now each marked node has already sent a message, and yet it has one new gossip (from the corresponding unmarked node) to forward to the root. Next apply this procedure recursively on these $\frac{N-1}{2}$ marked nodes, and inject a second batch of edge failures

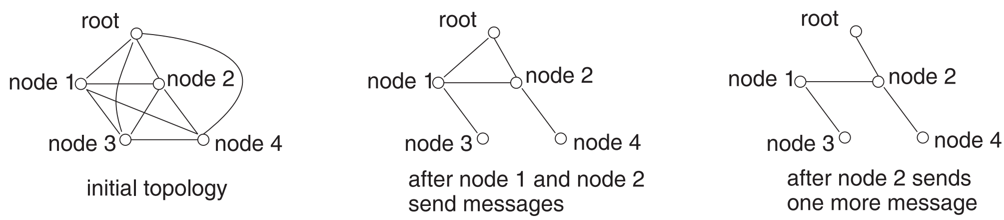

Fig. 12. Example FT lower bound topology for $N = 5$ and unrestricted b.

when exactly $\frac{N-1}{4}$ of them (e.g., node 2) have sent a second message. Continuing this argument can easily show that, for all the gossips to reach the root, some node needs to send at least $\log(N-1)+1$ messages.

Formalizing these arguments would be sufficient to prove Theorem 8.1, if the gossip assumption were valid. Unfortunately, the gossip assumption is actually rather strong and nodes can communicate "via silence". For example, a protocol may be such that if a node's value is 0, then the root does not need to collect a gossip from that node and simply uses 0 as the default value. It is also possible that node i sends a message to node j iff node i's value is 1, and then node j conceptually relays i's value to the root, by sending a message to the root iff this value is 0. Here, the root *never* collects a gossip from node i. Properly capturing all such possibilities will be the key challenge in our proof.

8.2. Topology and Adversary for Proving Theorem 8.1

Having explained the basic intuition and challenges, we now construct the topology and failure adversary for later proving Theorem 8.1. We assume that each node can take an integer value in the domain of $[0, 3n-1]$—this assumption will be removed later in Section 8.5. We construct the lower bound topology starting from a clique with $n+1$ nodes, with n being a power of 2. One of these nodes will be the root, while all other nodes are called *worker* nodes. We next insert a degree-two *dummy node* in the middle of each edge in the previous clique, so that failing the dummy node essentially fails the corresponding edge.[12] The topology thus has total $\frac{(n+1)(n+2)}{2}$ nodes. Each worker node has an integer value in $[0, 3n-1]$. All other nodes have value of 0. We use a vector $W = (w_1, w_2, \ldots, w_n)$ to denote the *input* (to the system), where $w_i \in [0, 3n-1]$ is the input value of worker node i.

Our deterministic and adaptive failure adversary here is the same as the simple failure adversary in the previous section, except that the adversary here fails the dummy nodes instead of failing the edges. Specifically, our adversary here conceptually partitions the worker nodes into groups, where each group has some *group members* and one group member is the *group leader*. Group membership and leadership change whenever the adversary inject failures. Initially, each worker is in its own group, with itself being the sole member (and thus the leader). The adversary keeps track of the set L of current leaders. A leader in L is *marked* once it sends a message. Once the number of marked leaders reaches $\frac{|L|}{2}$, the adversary pairs up each unmarked leader node j with a distinct marked leader node i. In cases where multiple leaders send messages in the same round, the adversary will mark them sequentially (by their ids), until the number of marked leaders becomes exactly $\frac{|L|}{2}$. Next, for each such node j, consider each of its neighboring dummy nodes. The dummy node may connect j with (i) the root,

[12]Under this construction, the total number of failures injected by our adversary later will reach $\Theta(N)$, where N is the graph size. To obtain the same asymptotic FT lower bound while injecting only $o(N)$ failures, one only needs to instead insert $\log n$ dummy nodes on each edge and fail only one of them.

(ii) some leader node other than node i, (iii) leader node i, or (iv) some nonleader node. If the dummy node connects j with the root or with some leader node other than node i, then at the beginning of the next round, the adversary fails that dummy node. After all these failures are injected, node i's group and node j's group are conceptually merged into one new group, with node i being the new group leader. Finally, the adversary updates L to be the set of those $\frac{|L|}{2}$ leaders of the new groups, clears all the marks on those leaders, and repeats this process until $|L|$ reaches 1 or the protocol terminates.

8.3. The Probing Game and Its Connection to Sum

We next define a simple *probing game*. We will draw a connection (or a reduction in the general sense) between this game and *deterministic* Sum protocols, when we run such Sum protocols under the topology and failure adversary as constructed in the previous section.

The *probing game* is played by a single player, against an input $W = (w_1, w_2, \ldots, w_n)$ where w_i is an integer in $[0, 3n-1]$. W is initially not known to the player, but the player knows n. The player proceeds in rounds. In each round, the player may choose to sequentially do zero, one, or multiple *probes*, where each probe is in the form of a tuple (i, j). The outcome of the probe (i, j) is a *hit* if $w_i = j$. Otherwise, it is a *miss*. For each probe, the player may *adaptively* choose what probe to do, based on the probing outcomes in previous rounds and so far in the current round. The goal of the player is to determine $\sum_{i=1}^{n} w_i$ based on the outcomes of the probes, while minimizing the total number of hits. Note that the player is not concerned with the total number of probes. For convenience later, we require that the player never does the same probe multiple times. In addition, if there has been a hit (i, j), then the player does not further probe (i, j') for any j' since the player has already learned w_i precisely.

Consider any *deterministic* Sum protocol running under the topology and adversary as constructed in the previous section. The protocol can obviously play various tricks to minimize communication (e.g., by communicating "via silence"). But Theorem 8.2 reveals that what the protocol fundamentally can do is no different from a virtual player playing the probing game and upon a hit, having some leader node in the topology send a message. In turn, the total number of messages sent by the leaders will be no smaller than the number of hits in the probing game.

Theorem 8.2. *Given any deterministic* Sum *protocol $\mathcal{P}$, there always exists a deterministic (adaptive) probing strategy $\mathcal{S}$ for the player in the probing game that satisfies the following property. For any input W, the player using $\mathcal{S}$ in the probing game against W always generates the same result as the result generated by the* Sum *protocol $\mathcal{P}$ running against W under the topology and adversary in Section 8.2. Furthermore, if the total number of hits in the probing game reaches n when using $\mathcal{S}$ against W, then the maximum number of bits sent by a node (across all nodes), when running $\mathcal{P}$ against W under the topology and adversary in Section 8.2, is at least* $\log n + 1$.

Proof. We will construct $\mathcal{S}$ based on the given (black-box) deterministic protocol $\mathcal{P}$. The constructed strategy $\mathcal{S}$ will determine the sequence of probes that the player should do in each round r, so that the probing outcomes will enable the player to simulate the execution of $\mathcal{P}$ in round r. During the course of the probing game, we say that a worker node i has been *hit* if there has been some probe (i, j) that is a hit. In any given round r of $\mathcal{P}$'s execution, we say that a group leader node i sends an *influential message* in round r if node i sends (i.e., locally broadcasts) a message in round r and the adversary does not fail the dummy node connecting node i with the root at the beginning of round $r + 1$. Note that since node i is a group leader in round r, it is guaranteed (by design of our adversary) that the dummy node connecting i with the root has not failed in round

r or earlier. If the adversary indeed fails that dummy node at the beginning of round $r+1$, then node i will no longer be a group leader in round $r+1$. Furthermore, node i (and node i's group) will be merged with another group, with another node being the new (merged) group's leader.

We will prove that when the player uses our constructed probing strategy $\mathcal{S}$, all the following properties hold for all round r where $0 \le r \le R$. Here R is the total number of rounds in $\mathcal{P}$'s execution over W, which must be finite. Recall that round 1 is the first round where there can possibly be a message sent in $\mathcal{P}$'s execution.

Property 1. In round r of the probing game, if the player does a probe (i, j) and if this probe is a hit, then in round r of $\mathcal{P}$'s execution, nodes i's group leader must send an influential message.

Property 2. In $\mathcal{P}$'s execution, if a group leader sends an influential message in round r, then all nodes in that group have been hit by the player in the probing game by the end of round r.

Property 3. In $\mathcal{P}$'s execution, immediately after the adversary injects potential failures at the beginning of round $r+1$, in each group there is at most one worker node that has not been hit by the player in the probing game.[13]

Property 4. In round r, the player in the probing game can generate all influential messages sent by all group leaders in round r of $\mathcal{P}$'s execution.

Lemma 8.3 proves that these four properties indeed hold under a properly constructed probing strategy $\mathcal{S}$. Now, by Property 4, since the influential messages from group leaders are the only messages that can affect the root via the dummy nodes, the player will be able to simulate those dummy nodes and all incoming messages to the root throughout the execution. Then, the root will be able to produce a final result in round R, and the player simply uses this result as the result for the probing game. Next, if the total number of hits in the probing game reaches n, then all nodes must have been hit since each node can only contribute one hit. By Property 3, we know that in any round, each group can have at most one worker node that has not been hit. Initially there are n groups, and thus there must exist at least $\frac{n}{2}$ groups where each group contributes a hit. By Property 1, the $\frac{n}{2}$ leaders of these $\frac{n}{2}$ groups will each send an influential message. Once all these influential messages are sent, our adversary will introduce failures so that there will be $\frac{n}{2}$ new groups, with these $\frac{n}{2}$ nodes as new leaders. Since we still need to have $\frac{n}{2}$ more hits and since each group can contribute at most one hit, among those $\frac{n}{2}$ groups, there must exist $\frac{n}{4}$ groups whose leaders will each send a second influential message. Continuing such argument will show that in order for the total number of hits in the probing game to reach n, some node in the Sum protocol $\mathcal{P}$ will have to send at least a influential messages, where a is the total number of terms in the summation of $n = \frac{n}{2} + \frac{n}{4} + \cdots + 2 + 1 + 1$. Observing that $a = \log_2 n + 1$ and that a messages translate to at least a bits completes the proof. □

Lemma 8.3. *Under the conditions of Theorem 8.2, there exists a probing strategy $\mathcal{S}$ such that the four properties described in the proof of Theorem 8.2 hold for all round r where $0 \le r \le R$. Here R is the total number of rounds in $\mathcal{P}$'s execution over W.*

Proof. We prove via an induction on r. For $r = 0$, we construct $\mathcal{S}$ such that no probes are done in round 0. The four properties trivially hold for round 0. Now consider any round $r > 0$, while assuming that they hold for all rounds before r.

[13]First, we explicitly mention "after failure injection" since group membership is affected by failures. Second, we consider the beginning of round $r+1$ instead of the beginning of round r to facilitate our later proof by induction.

We first construct the set of probes that the player should do in round r. Consider any group g in round r of $\mathcal{P}$'s execution, after the adversary injects potential failures at the beginning of round r. (Note that failures affect group membership, and thus we explicitly state that g is defined after the failures have been injected in round r.) By inductive hypothesis on Property 3, there is at most one worker node $miss(g)$ in g that has not been hit. Thus, the player knows the value of all other nodes in g. If $miss(g)$ exists, then the player will exhaustively enumerate all j's such that there has not been a probe $(miss(g), j)$. Intuitively, j has not been ruled out as a possible value for $miss(g)$. For each such j, the player *tries* simulating $\mathcal{P}$'s execution on all nodes in g, from round 0 to round r (inclusive). Doing so will enable the player to determine the message (if any) sent in round r by g's group leader. Such simulation is possible since the player has the values of all the nodes in that group, and also because by inductive hypothesis on Property 4, the player can generate all influential messages sent by other group leaders up to round $r-1$. In particular for a node i in g, all incoming messages from nodes outside of g must be sent from those dummy nodes connecting node i with those group leaders in round 1 through round $r-1$. The reason is that in any round from 1 through $r-1$, nonleader group members can only have neighboring dummy nodes connecting them to other nodes in their own groups and not to node i. Furthermore, noninfluential messages from a group leader can never affect either the root or nodes in other groups (via the corresponding dummy nodes), since those dummy nodes will be failed right after they receive those noninfluential messages. Finally, we do not yet know what influential messages other group leaders will send in round r, but those will not affect the potential message sent in round r by g's group leader.

We say that a $(miss(g), j)$ combination is a *candidate probe* if $miss(g)$ exists and by this process, the player has determined that there will be a message sent in round r by g's group leader *if* j is the value of node $miss(g)$. (We do not yet know whether this message will be influential.) Next, the player orders all candidate probes, using g as the primary key and j as the secondary key. In round r, the player sequentially issues the probes in this ordered list, subject to the following two constraints. First, if a certain $(miss(g), j)$ probe is a hit, then the player will skip all following probes in the form of $(miss(g), j')$ for all j'. Second, if the number of hits so far is such that the adversary is ready to inject the next batch of failures, the player skips all the remaining probes in the ordered list.

We next prove that such a probing strategy does satisfy the four properties. Property 1 clearly holds since a hit of $(miss(g), j)$ means that node $miss(g)$ indeed has a value of j. Given the trial simulation and since everything is deterministic, $miss(g)$'s group leader will send a message in round r of $\mathcal{P}$'s execution. Furthermore, since the probes are done sequentially and since the player must have encountered this hit before the adversary is ready to inject the next batch of failures, the adversary will not fail the dummy node connecting $miss(g)$'s group leader to the root at the beginning of round $r+1$.

For Property 2, we need to prove that if a group g's group leader sends an influential message in round r, then all nodes in g have been hit by the end of round r. If $miss(g)$ does not exist, we already hit all nodes in g. If $miss(g)$ exists, let j be the value of the node $miss(g)$. Clearly, there has never been a probe $(miss(g), j)$. By our construction of the probing strategy in round r, $(miss(g), j)$ will be a candidate probe. If the player indeed probes $(miss(g), j)$ in round r, then $miss(g)$ will be hit in round r and we are done. If the player does not probe $(miss(g), j)$ in round r, the only possibility is that the adversary is ready to inject the next batch of failures at the beginning of round $r+1$. In such a case, the adversary will fail the dummy node connecting g's group leader to the root, making the message (if any) sent by g's group leader noninfluential.

For Property 3, if the adversary does not inject failures at the beginning of round $r+1$, then clearly the property inherited from the beginning of round r continues to hold at the beginning of round $r+1$. If the adversary does inject failures at the beginning of round $r+1$, then the group membership in round r and the group membership in round $r+1$ are different. Let $g_1, g_2, \ldots, g_l$ be the l groups in round r, immediately after the adversary potentially inject failures at the beginning of round r. Without loss of generality, assume that after the failures are injected, $g_{i+l/2}$ is merged with g_i (for $1 \le i \le l/2$) to form a new group, with g_i's leader being the leader of the new group. By inductive hypothesis on Property 3, immediately after the adversary potentially injects failures at the beginning of round r, $g_{i+l/2}$ $(1 \le i \le l/2)$ has at most one node that has not been hit. Given how the adversary injects failures, we know that the leader of g_i $(1 \le i \le l/2)$ must have sent an (influential) message in round r or earlier and after group g_i is formed. By Property 2 (both in round r and in earlier rounds), we know that all nodes in group g_i have been hit by the end of round r. This means that after merging g_i and $g_{i+l/2}$, the new group still only has at most one node that has not been hit.

Finally, for Property 4, consider any given group g whose leader sends an influential message in round r. By Property 2, we know that all nodes in g have been hit by the end of round r, and thus the player knows all their values. Same as in the earlier trial simulation, by inductive hypothesis on Property 4, the player can generate all influential messages sent by other group leaders up to and including round $r-1$. This means that the player can generate all incoming messages (up to and including round $r-1$) that may affect nodes in g. By same arguments as earlier, the player will be able to simulate $\mathcal{P}$'s execution on all nodes in g from round 0 to round r (inclusive). Also note that the messages received in round r, which we do not know yet, will not affect the messages sent by g's group leader in round r. Thus, the player can generate that influential message sent by the group leader in round r. $\square$

8.4. Lower Bound on the Number of Hits in the Probing Game

With the connection between Sum and the probing game proved in the previous section, we now intend to obtain a lower bound on the number of hits in the probing game. Such lower bound will later translate to a lower bound on the communication complexity of Sum. A simpler version of this probing game was analyzed in Dhulipala et al. [2010] to reason about silence-based communication, in a failure-free setting. There the player is not allowed to interleave probes on w_i with probes on $w_{i'}$. Thus, for our purpose, we prove the following lower bound result on the probing game where the probes may be arbitrarily interleaved.

Lemma 8.4. *Consider the set U of all the $(3n)^n$ possible inputs to the probing game $(n \ge 2)$ and any given (adaptive) deterministic probing strategy that can give correct (zero-error) results for at least $\frac{2}{3}$ fraction of those inputs. There must exist an input such that using that strategy, the player encounters n hits under that input.*

Proof. We prove by contradiction, and assume that there exists a deterministic probing strategy $\mathcal{S}$ that gives correct results for at least $\frac{2}{3}$ fraction of inputs and has at most $n-1$ hits for all inputs.

Consider any given input $W = (w_1, w_2, \ldots, w_n) \in U$, which is initially unknown to the player. At any point of time during the game, we define w_i's *residual domain* (denoted as D_i) to be the set $\{j\}$ if there has been a probe (i, j) so far which is a hit. Otherwise, w_i's *residual domain* D_i is defined to be the set:

$$\{0, 1, 2, \ldots, 3n-1\} \setminus \{j \mid \text{there has been a probe } (i, j) \text{ that is a miss}\}.$$

Intuitively, w_i's residual domain is the possible domain of w_i given the probe outcomes so far. When the game ends under $\mathcal{S}$, W has a unique *residual domain vector* $D = (D_1, D_2, \ldots, D_n)$, where D_i is the residual domain of w_i. We next prove a simple useful property on W's residual domain vector to facilitate later reasoning. We claim that for any input $Z = (z_1, z_2, \ldots, z_n)$ where $z_i \in D_i$, the probes done by the player and all the probe outcomes must be identical under input W and input Z. This in turn implies that Z will have the same residual domain vector as well as the same final output as that of W. We prove this claim for the kth probe, via a simple induction on k. The induction base for the zeroth probe clearly holds. Now consider the kth probe. We already know that all previous probes and their outcomes are identical under W and under Z. Since the player is deterministic, we know that the kth probe will be the same under W and Z. Let this probe be (i, j). If (i, j) is a hit for W, then we must have $D_i = \{j\}$. Since $z_i \in D_i$, we know that $z_i = j$ and the probe (i, j) will be a hit for Z as well. If (i, j) is a miss for W, then we must have $j \notin D_i$. Since $z_i \in D_i$, we know that $z_i \neq j$ and thus the probe will be a miss for Z as well. Thus, the outcome of the kth probe will be identical under input W and input Z.

We now leverage this property to prove the following claim. Define U' to be that set of inputs such that for each input $W \in U'$, when the game ends, in W's residual domain vector $D = (D_1, D_2, \ldots, D_n)$ there exists some D_i where $|D_i| \geq 2$. We claim that in order for the player to generate results correctly for at least $\frac{2}{3}$ fraction of all the inputs, $|U'|$ must be no larger than $\frac{2}{3}|U|$.

We prove this claim by contradiction and assume that $|U'| > \frac{2}{3}|U|$. We partition U' into disjoint subsets such that all inputs in the same subset have the same residual domain vector. Consider any such subset U'_D where all inputs in the set has the same $D = (D_1, D_2, \ldots, D_n)$ as their residual domain vectors. By the earlier property on residual domain vector, we know that U'_D contains at least all those inputs Z where $z_i \in D_i$ for $1 \leq i \leq n$, and all such inputs Z will result in the same output. Furthermore, if an input Z' is such that $z'_i \notin D_i$ for some i, then it is impossible for Z' to be in U'_D. In other words, U'_D must be exactly the set of those inputs Z where $z_i \in D_i$ for $1 \leq i \leq n$. Next, without loss of generality, assume that $|D_1| \geq 2$. Consider any given $j_2 \in D_2$, $j_3 \in D_3, \ldots, j_n \in D_n$. For each $j \in D_1$, $Z = (j, j_2, j_3, \ldots, j_n)$ must be in U'_D, and the player will produce the same output. Such output can be correct for at most one j. This in turn means that the player can generate a correct result for at most $\frac{1}{2}$ fraction of the inputs in U'_D. Therefore, for all the inputs in the set U', the player can generate correct results for at most $\frac{1}{2}$ fraction as well. Thus, the player can generate a result correctly for at most $\frac{1}{2}|U'| + |U \setminus U'| = \frac{1}{2}|U'| + |U| - |U'| = |U| - \frac{1}{2}|U'| < \frac{2}{3}|U|$ inputs.

We have just proved that $|U \setminus U'| \geq \frac{1}{3}|U|$. We next use this result to prove that under some input in $U \setminus U'$, the number of hits will be n. For every input $W \in U \setminus U'$, we know that W's residual domain vector $D = (D_1, D_2, \ldots, D_n)$ satisfies $|D_i| = 1$ for all i. Essentially, the player has "pinpointed" the value of each w_i and knows W precisely, instead of only its sum. This means that in the probing game, the player actually needs to learn the input precisely for at least $\frac{1}{3}$ fraction of all the inputs. We next prove that for the player to achieve such a goal, there will be n hits on at least one of the inputs in $U \setminus U'$.

Given such a goal, consider any given point of time where the player decides to do a probe (i, j). (This necessarily means that there has not been a hit on w_i, and also means that $j \in D_i$ where D_i is current residual domain of w_i.) Since the goal is to learn the input precisely, doing such a probe is no different from doing any other probe (i, j') as long as $j' \in D_i$. With such an observation, we can now make the following without loss of generality assumption: For any given input W and any given i, consider all probes done by the player in the form of (i, j_0), (i, j_1), (i, j_2), ... Without loss of generality, we can assume that $j_0 = 0$, $j_1 = 1$, $j_2 = 2$, ... We know that under the probing strategy $\mathcal{S}$, there are at most $n-1$ hits under all inputs, including all inputs $W \in U \setminus U'$. Consider any given $W \in U \setminus U'$ and let i be the index of the component that has not been hit. Since $|D_i| = 1$ and by our earlier without loss of generality assumption, we know that $w_i = 3n - 1$. However, the number of such inputs is

$$|\{W \mid W \in U \text{ and } \exists i \text{ such that } w_i = 3n-1\}| = (3n)^n - (3n-1)^n < \frac{1}{3}(3n)^n, \quad \text{for } n \geq 2.$$

This contradicts with $|U \setminus U'| \geq \frac{1}{3}|U|$. Thus, under some $W \in U \setminus U'$, there will be n hits. □

8.5. Proof for Theorem 8.1

We are now ready to prove a series of lemmas, which directly lead to Theorem 8.1. The following lemma is proved by first establishing a connection between randomized complexity and distributional complexity via well-known techniques [Kushilevitz and Nisan 1996], and then using Theorem 8.2 and Lemma 8.4 to obtain a lower bound on the distributional complexity. In the following, the notation of $\mathrm{FT}_{0,\delta}$ simply means $\mathrm{FT}_{\epsilon,\delta}$ with $\epsilon = 0$.

LEMMA 8.5. *Consider any $b \geq 1$ and any integer $N = \frac{(n+1)(n+2)}{2}$ where n is a power of* 2. *If in the* SUM *problem each node may take an integer value in* $\{0, 1, \ldots, 3n-1\}$, *then there exists a connected topology G with N nodes, such that:*

$$\mathrm{FT}_{0,\frac{1}{3}}(\textsc{Sum}_G, b) \geq \log n + 1.$$

PROOF. We let G be the topology constructed in Section 8.2, and consider the deterministic and adaptive adversary described there. It will be convenient to view this adversary as part of the SUM protocol. Namely, given any randomized SUM protocol, we can consider a randomized "augmented protocol" which repeatedly executes one round of the randomized SUM protocol and then invokes the (deterministic) adversary to potentially inject failures. We thus no longer need to discuss the adversary separately.

Now consider the optimal randomized augmented protocol that generates a zero-error result with probability at least $\frac{2}{3}$ for all input W, while incurring a *worst-case* (over the coin flips) communication complexity of $\mathrm{FT}_{0,\frac{1}{3}}(\textsc{Sum}_G, b)$. If we subject this protocol against an input chosen uniformly at random out of all possible inputs, then trivially the protocol still generates a zero-error result with probability at least $\frac{2}{3}$, where the probability is taken over both the input distribution and the random coin flips. Now, let us view this randomized augmented protocol as a distribution over deterministic augmented protocols. Then, there must exist at least one deterministic protocol $\mathcal{P}$ which can generate a zero-error result with probability at least $\frac{2}{3}$ under this uniform input distribution, since otherwise the expectation taken over all deterministic augmented protocols cannot reach $\frac{2}{3}$. Finally, let a denote the maximum number of bits sent by a node, across all nodes when we run $\mathcal{P}$ against the worst-case input (that maximizes a). Since $\mathcal{P}$ is selected by the randomized algorithm with positive probability and since $\mathrm{FT}_{0,\frac{1}{3}}(\textsc{Sum}_G, b)$ is defined over worst-case coin flips, we have $a \leq \mathrm{FT}_{0,\frac{1}{3}}(\textsc{Sum}_G, b)$.

On the other hand, Theorem 8.2 tells us that given such a $\mathcal{P}$, there exists a corresponding probing strategy $\mathcal{S}$ in the probing game so that using $\mathcal{S}$, the player in the probing game can generate the same result as $\mathcal{P}$. Since $\mathcal{P}$ generates a zero-error result for at least $\frac{2}{3}$ fraction of the inputs, we know that the player using $\mathcal{S}$ generates a zero-error result for so many inputs as well. Next, by Lemma 8.4, there exists some input W such that the player encounters n hits. In turn, Theorem 8.2 now tells us that, when running the SUM protocol $\mathcal{P}$ against this input W, some node sends at least $\log n + 1$ bits. Thus, we have $\mathrm{FT}_{0,\frac{1}{3}}(\mathrm{SUM}_G, b) \geq a \geq \log n + 1$. □

The following corollary extends this lemma to our standard setting where nodes only have binary values and also where N can be any integer.

COROLLARY 8.6. *Consider any $b \geq 1$ and any integer $N \geq 5$. Let n be the largest integer that is a power of 2 and satisfies $\frac{(n+1)(n+2)}{2} + n(3n-1) \leq N$. There exists a connected topology G with N nodes, such that:*

$$\mathrm{FT}_{0,\frac{1}{3}}(\mathrm{SUM}_G, b) \;\geq\; \log n + 1.$$

PROOF. Let $N_1 = \frac{(n+1)(n+2)}{2}$ and we first construct a connected topology G_1 with N_1 nodes as described in Section 8.2. Next, we attach $(3n-1)$ degree-1 *follower* nodes to each worker node in G_1, and attach $(N - N_1 - n(3n-1))$ degree-1 nodes to the root of G_1. All those degree-1 nodes attached to G_1's root will always have value 0. Let G be the resulting N-node connected topology. One can trivially obtain a reduction from the SUM problem on G_1 (where each worker node has an integer value in $\{0, 1, \ldots, 3n-1\}$) to the SUM problem on G (where each node has a binary value). In particular in the reduction, the root in G_1 will simulate the root in G and also all the degree-1 neighbors of the root in G. Each worker node in G_1 will simulate the corresponding worker node and its $(3n-1)$ followers in G. If the worker node i in G_1 has a value of w_i, then in G the first w_i of the corresponding worker node's follower nodes will have value 1 and the remaining $(3n-1-w_i)$ follower nodes will have value 0. Combining this reduction with the lower bound on the SUM problem on G_1 from Lemma 8.5, we have $\mathrm{FT}_{0,\frac{1}{3}}(\mathrm{SUM}_G, b) \geq \log n + 1$. □

The next corollary extends the previous corollary to $\mathrm{FT}_{\epsilon,\frac{1}{3}}$ for $\epsilon \geq \frac{1}{N}$.

COROLLARY 8.7. *Consider any $b \geq 1$, any integer $N \geq 15$, and any $\epsilon \in [\frac{1}{N}, \frac{1}{15}]$. Let n be the largest integer that is a power of 2 and satisfies $\frac{(n+1)(n+2)}{2} + n(3n-1) \leq \frac{1}{3\epsilon}$. There exists a connected topology G with N nodes, such that:*

$$\mathrm{FT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}_G, b) \;\geq\; \log n + 1.$$

PROOF. Let $N_1 = \frac{1}{3\epsilon}$ and we first construct a connected topology G_1 with N_1 nodes such that $\mathrm{FT}_{0,\frac{1}{3}}(\mathrm{SUM}, G_1, b_1) \geq \log n + 1$ for any b_1. Corollary 8.6 ensures that such G_1 exists. Next, we attach $N_2 = N - N_1$ nodes to the root of G_1, and let the resulting topology be G. Those N_2 nodes will always have a value of 0 and will never fail. Note that the final sum on G can never be above $\frac{1}{3\epsilon}$. If we have at most ϵ relative error on the final sum on G, then the absolute error is at most $\frac{1}{3\epsilon} \cdot \epsilon = \frac{1}{3}$. Since the exact sum must be an integer, to generate a result with ϵ relative error in G, the protocol intuitively needs to produce a zero-error result. Putting it another way, if the output is not an integer, we can always output the closest integer instead. Doing so will never cause an output that was previously within ϵ-error bound to exceed the ϵ-error bound after the conversion. This observation enables a trivial reduction from $\mathrm{FT}_{0,\frac{1}{3}}(\mathrm{SUM}, G_1, b_1)$ to $\mathrm{FT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}, G, b)$, which gives

$$\mathrm{FT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}_G, b) \;\geq\; \mathrm{FT}_{0,\frac{1}{3}}(\mathrm{SUM}_{G_1}, b_1) \;\geq\; \log n + 1. \quad \square$$

Combining Corollary 8.6 and Corollary 8.7 enables us to easily prove Theorem 8.1.

PROOF FOR THEOREM 8.1. We first prove the lower bound on $\mathrm{FT}_0(\mathrm{SUM}_N, b)$. For any $N \geq 5$, consider the N-node connected topology G as constructed by Corollary 8.6.

Together with Lemma A.1, we trivially have

$$\mathrm{FT}_0(\mathrm{SUM}_N, b) \geq \frac{1}{3}\mathrm{FT}_{0,\frac{1}{3}}(\mathrm{SUM}_N, b) \geq \frac{1}{3}\mathrm{FT}_{0,\frac{1}{3}}(\mathrm{SUM}_G, b) \geq \frac{1}{3}(\log n + 1).$$

By Corollary 8.6, here n is the largest integer that is a power of 2 and satisfies $\frac{(n+1)(n+2)}{2} + n(3n-1) \leq N$. Thus, we have $n = \Theta(\sqrt{N})$, which implies $\mathrm{FT}_0(\mathrm{SUM}_N, b) = \Omega(\log N)$.

We next prove the lower bound on $\mathrm{FT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}_N, b)$ for $\epsilon \geq \frac{1}{N}$. For any $N \geq 15$, consider the N-node connected topology G as constructed by Corollary 8.7. We trivially have

$$\mathrm{FT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}_N, b) \geq \mathrm{FT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}_G, b) \geq \log n + 1.$$

By Corollary 8.7, here n is the largest integer that is a power of 2 and satisfies $\frac{(n+1)(n+2)}{2} + n(3n-1) \leq \frac{1}{3\epsilon}$. Thus, we have $n = \Theta(\frac{1}{\sqrt{\epsilon}})$, which implies $\mathrm{FT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}_N, b) = \Omega(\log \frac{1}{\epsilon})$.

Finally, for $\epsilon = \Omega(\frac{1}{N})$ but $\epsilon < \frac{1}{N}$ (in which case ϵ is necessarily $\Theta(\frac{1}{N})$), we have

$$\mathrm{FT}_{\epsilon,\frac{1}{3}}(\mathrm{SUM}_N, b) \geq \mathrm{FT}_{\frac{1}{N},\frac{1}{3}}(\mathrm{SUM}_N, b) = \Omega(\log N) = \Omega\left(\log \frac{1}{\epsilon}\right). \quad \square$$

9. DISCUSSIONS AND EXTENSIONS

Putting together the NFT upper bounds (Theorem 4.1) and FT lower bounds (Theorem 5.1, 6.1, and 8.1) will directly give us the exponential gaps, as summarized in Figure 1 from Section 1. Specifically, one only needs to apply Theorem 5.1 for $1 \leq b \leq 2 - c$, Theorem 6.1 for $2 - c < b \leq N^{0.25-c}$ or $2 - c < b \leq \frac{1}{\epsilon^{0.5-c}}$, and Theorem 8.1 for $b > N^{0.25-c}$ or $b > \frac{1}{\epsilon^{0.5-c}}$, with c being any positive constant below 0.25. It is worth noting that such exponential gap results apply as well to the following extensions of the model defined in Section 3.

Total Number of Failures. Section 3 allowed the total number of failures to be up to $N - 1$. In all the executions (of the SUM protocol) considered in our FT lower bound proofs, the failure adversary actually injects only $o(N)$ failures in G. Thus, our lower bounds apply, without any modification, as long as the total number of failures is allowed to be up to any constant fraction of N.

Our proofs carry over to even smaller number of failures, without disrupting the exponential gap, if we lower the degree of our polynomial lower bounds. As a concrete example, Theorem 5.1 proved that $\mathrm{FT}_0(\mathrm{SUM}_N, b) = \Omega(\frac{N}{\log^2 N})$. In the theorem's proof, we injected total $f = n = \Theta(\frac{N}{\log N})$ failures. When f is smaller than $\Theta(\frac{N}{\log N})$, it is trivial to generalize the proof and show that $\mathrm{FT}_0(\mathrm{SUM}_N, b) = \Omega(\frac{f}{\log N})$. To do so, one only needs to set $n = f$ in the original proof, and then add enough dummy nodes to the topology so that the total number of nodes in the topology reaches N.

Private-Coin and Deterministic Protocols. Section 3 only considered public-coin protocols. Private-coin protocols and deterministic protocols are also fully but implicitly covered by all our theorems. This is simply because the NFT upper bound protocol with zero-error in Theorem 4.1 is actually deterministic, while the one with (ϵ, δ)-error uses only private coins.

Allowing Integer Values for Each Node. In practice, each node in the network may have some integer value instead of a binary value. Our FT lower bounds obviously carry over to integer values. Our NFT upper bounds continue to apply as long as the integer value has a domain no larger than some polynomial of N.

Other Network Models. Because of the paramount practical importance of communication complexity in wireless networks, Section 3 chose to define a system model capturing wireless networks. All our theorems continue to apply regardless of whether collision is considered (i.e., whether a node can receive messages simultaneously from multiple neighbors in a round) and regardless of whether the communication is point-to-point or (local) broadcast. Note that in settings without collisions, $\Lambda(G)$ is simply the eccentricity of the root in G.

Letting All Nodes Know the Result. We required only the root to learn the final result. To let all nodes know the result, the root in our upper bound protocol in Theorem 4.1 can simply broadcast the result to all nodes along some spanning tree.

Unknown Topology. Assuming a known topology, as in Section 3, strengthens our FT lower bounds. For the upper bounds obtained via tree-aggregation, with unknown topologies, it suffices to simply add a distributed preprocessing phase for building a spanning tree.

10. CONCLUSIONS AND FUTURE WORK

Tolerating failures has been a key focus of distributed computing research from the very beginning. Adding this fault tolerance requirement to multi-party communication complexity leads to the following natural question: "If we want to compute a function in a fault-tolerant way, what will the communication complexity be?" For this question, this paper focuses specifically on (i) tolerating node crash failures, and (ii) computing the function over general topologies. Within such a context, we reveal that the impact of failures on communication complexity can be large, at least for the SUM aggregation function in networks with general topologies. Specifically, we show that there exists (at least) an exponential gap between the NFT and FT communication complexity of SUM.

This result attests that FT communication complexity needs to be studied separately from the simpler NFT communication complexity, instead of being considered as an "amended" version of NFT communication complexity. There are many interesting follow-up open questions on the subject.

—Our lower bound (i.e., worst-case) topologies for SUM are carefully constructed. We are currently investigating to what extent our lower bounds can generalize to other topologies.
—We have mainly focused on the exponential gap for SUM, and have been less concerned about specific degrees of the polynomials in the FT lower bounds. Can we further strengthen these lower bounds? Note that even our lower bound on the communication complexity of UNIONSIZECP is not tight (i.e., roughly $\frac{1}{q}$ factor from the upper bound), and thus improvement might be possible even there.
—Our lower bounds show that the bottleneck node in G will incur a large communication complexity. How many nodes in G will incur asymptotically similar communication complexity as that node? Putting it another way, how many hot spots are there?
—We have defined the FT communication complexity of SUM across all protocols that can tolerate a certain number of failures. Similar to the idea of early stopping distributed consensus protocols, among this class of protocols, it would be interesting to investigate to what extent a protocol can incur a smaller communication complexity when the number of failures *that actually happen* (denoted as f') is small. Repeatedly invoking tree-aggregation incurs a communication complexity of $O(f' \log N)$—can we do better? We are currently investigating both upper bounds and lower bounds on this.

—Our results extend to some other functions such as SELECTION, via trivial reductions to and from SUM. But clearly there are also many interesting functions whose FT communication complexity is still unknown. In particular, can we characterize the set of functions having exponential gaps?

For answering these questions, we believe that some of the insights developed in this article (e.g., on the role of failures in the reduction and on the cycle promise) can be valuable.

APPENDIXES

A. SOME USEFUL KNOWN/TRIVIAL RESULTS

This section describes some known results that this article uses. These results and their proofs are not our contribution. We include the details and sometime the proofs here only for completeness, because some of them were folklore results, or were not formally stated, or were not stated to cover FT communication complexity, or were proved under slightly different models in a restricted form. In the next, the notations $\mathrm{NFT}_{0,\delta}$, and $\mathrm{FT}_{0,\delta}$ simply mean $\mathrm{NFT}_{\epsilon,\delta}$, and $\mathrm{FT}_{\epsilon,\delta}$ with $\epsilon = 0$, respectively.

The Impact of Failures on the Communication Complexity of MAX is Small. Consider the same setting as for the SUM function, except that now each node has an integer value whose domain is no larger than some polynomial of N, with N being the total number of nodes in the network. The MAX functions asks for the maximum of all these values. For simplicity, we will ignore collision in our discussion on MAX and only consider protocols generating zero-error results. Let the time complexity constraint be $\Theta(\log N)$ aggregation rounds. We will show that the gap between the NFT and FT communication complexity of MAX is no larger than the gap between $\Omega(\log N/\log\log N)$ and $O(\log N)$.

We first obtain a lower bound on the NFT communication complexity of MAX. Consider a star topology where the root is directly connected to $N-1$ nodes. An aggregation round in this topology has only one round. Pick an arbitrary node out of those $N-1$ nodes and let it be node τ. All nodes except τ have a value of 0. So computing MAX would be the same as sending the value at τ to the root. It is trivial to show that doing so within $\Theta(\log N)$ rounds requires at least $\Omega(\log N/\log\log N)$ bits of communication.

Next, we sketch a simple folklore fault-tolerant MAX protocol, whose communication complexity is $O(\log N)$. This fault-tolerant MAX protocol is a simplified version of some more complex protocols in the literature (e.g., the protocol from Yu [2011] which tolerates byzantine failures). The protocol does a simple binary search on the entire value domain. At each step of the binary search, the protocol conceptually asks whether any node in the system has a value that is no smaller than some specific value. Clearly, we only need to ask $O(\log N)$ such questions to determine MAX. For each such question, a node floods a single bit of "1" (without including its id) as the reply if its value is no smaller than the specified value. Other nodes in the system will relay/forward only the very first reply they see for that question. One can easily show that this process is fault-tolerant, and each question only requires each node to send $O(1)$ bits. Furthermore, at the end of the flooding, each node in the system will know the answer to this question, and thus can infer what the next question will be. Thus, posing the questions does not incur any extra communication. The total number of bits each node needs to send in this fault-tolerant protocol for MAX is $O(\log N)$.

Known Relation Between NFT_0, FT_0 *and* $\mathrm{NFT}_{0,\delta}$, $\mathrm{FT}_{0,\delta}$. Note that we do not necessarily have $\mathrm{NFT}_0 \geq \mathrm{NFT}_{0,\delta}$, since NFT_0 is the average-case (over random coin flips in the protocol) communication complexity, while $\mathrm{NFT}_{0,\delta}$ is the worst-case (over random-coin flips in the protocol) communication complexity. Nevertheless, the following relation in

Lemma A.1 is well known [Kushilevitz and Nisan 1996]. This relation trivially applies to fault-tolerant communication complexity as well.

LEMMA A.1 (ADAPTED FROM KUSHILEVITZ AND NISAN [1996]). *For any communication complexity problem* Π *and* $\delta > 0$, $\mathrm{NFT}_0(\Pi) \geq \delta \mathrm{NFT}_{0,\delta}(\Pi)$. *Similarly, for any* $b \geq 1$ *and* $\delta > 0$, $\mathrm{FT}_0(\mathrm{SUM}_N, b) \geq \delta \mathrm{FT}_{0,\delta}(\mathrm{SUM}_N, b)$.

PROOF. Consider the optimal zero-error randomized protocol for Π, which generates a zero-error result while incurring an *expected* (over the random coin flips in the protocol) communication complexity of $\mathrm{NFT}_0(\Pi)$ bits. By Markov's inequality, the protocol's communication complexity exceeds $\mathrm{NFT}_0(\Pi)/\delta$ bits with probability at most δ. We can thus construct a new protocol which behaves the same as the original one except that a node stops once it has sent $\mathrm{NFT}_0(\Pi)/\delta$ bits. Obviously, this protocol outputs correct results with probability at least $1 - \delta$, and incurs a *worst-case* communication complexity of $\mathrm{NFT}_0(\Pi)/\delta$ bits, implying $\mathrm{NFT}_{0,\delta}(\Pi) \leq \mathrm{NFT}_0(\Pi)/\delta$. A similar proof can show $\mathrm{FT}_{0,\delta}(\mathrm{SUM}_N, b) \leq \mathrm{FT}_0(\mathrm{SUM}_N, b)/\delta$. □

Known Relation Between $\mathrm{NFT}_0(\Pi)$, $\mathrm{NFT}_{\epsilon,\delta}(\Pi)$ *and* $\mathrm{NFT}_0(\Pi, t)$, $\mathrm{NFT}_{\epsilon,\delta}(\Pi, t)$. The following lemma is a slightly extended version of the corresponding theorem from Impagliazzo and Williams [2010], which draws a connection between NFT communication complexity with synchronized rounds and NFT communication complexity without synchronized rounds. Since our synchronous round model is slightly different from Impagliazzo and Williams [2010], we provide a proof sketch below for the sake of completeness.

LEMMA A.2 (ADAPTED FROM IMPAGLIAZZO AND WILLIAMS [2010]). *For any two-party communication complexity problem* Π *and any* $t \geq 2$, *we have* $\mathrm{NFT}_0(\Pi) = \mathrm{NFT}_0(\Pi, t) \cdot O(\log t)$ *and* $\mathrm{NFT}_{\epsilon,\delta}(\Pi) = \mathrm{NFT}_{\epsilon,\delta}(\Pi, t) \cdot O(\log t)$.

PROOF. Consider any given protocol $\mathcal{P}$ (with $\mathcal{P}_A$ being Alice's part of the protocol and $\mathcal{P}_B$ being Bob's part), that can solve Π under the synchronous round setting with a bits (either on expectation or worst-case) of communication, while always terminating within t synchronous rounds. We construct a protocol $\mathcal{Q}$ (with $\mathcal{Q}_A$ and $\mathcal{Q}_B$ similarly defined) that can solve the problem with $O(a \log t)$ bits (either on expectation or worst-case, respectively) of communication complexity in the classic setting without synchronous rounds.

In $\mathcal{Q}$, Alice and Bob each maintains a local counter initialized to 1. These two counters correspond to the round number needed by $\mathcal{P}$. Let the current counter value on Alice be r_A. In $\mathcal{Q}_A$, Alice first *tries* executing $\mathcal{P}_A$ for rounds $r_A, r_A + 1, r_A + 2, \ldots$, while *assuming* that $\mathcal{P}_B$ does not send any message in any of those rounds. Alice then determines r'_A ($r'_A \geq r_A$), the first round during which $\mathcal{P}_A$ sends a message in this trial execution. Similarly, Bob determines r'_B. Alice and Bob then exchange r'_A and r'_B, taking $2 \log t$ bits. Let $r' = \min(r'_A, r'_B)$. Alice next executes $\mathcal{P}_A$ (for real) for rounds $r_A, r_A + 1, \ldots, r'$, and then sends a message to Bob if $r'_A = r'$. Similarly in $\mathcal{Q}_B$, Bob executes $\mathcal{P}_B$ for rounds r_B, $r_B + 1, \ldots, r'$, and then sends a message to Alice if $r'_B = r'$. Note that for round r', $\mathcal{P}$ must incur at least one bit of communication. Thus, for each bit $\mathcal{P}$ incurs, $\mathcal{Q}$ incurs at most $2 \log t + 1 = O(\log t)$ bits. After the message exchange for round r', Alice and Bob set $r_A = r' + 1$ and $r_B = r' + 1$, and repeat this process until $\mathcal{P}$ terminates. □

Known Results on UNIONSIZE*'s Communication Complexity.* Trivially combining several recent results [Chakrabarti and Regev 2011; Impagliazzo and Williams 2010; Woodruff 2004] leads to the following known results.

THEOREM A.3 (ADAPTED FROM CHAKRABARTI AND REGEV [2011]). $\mathrm{NFT}_0(\mathrm{UNIONSIZE}_n) = \Omega(n)$ *and* $\mathrm{NFT}_{\epsilon,\frac{1}{3}}(\mathrm{UNIONSIZE}_n) = \Omega(\frac{1}{\epsilon^2})$ *for* $\epsilon = \Omega(\frac{1}{\sqrt{n}})$.

PROOF. We first prove the theorem for $\epsilon \geq \frac{1}{\sqrt{n}}$. This proof for $\text{NFT}_{\epsilon,\frac{1}{3}}(\text{UNIONSIZE}_n) = \Omega(\frac{1}{\epsilon^2})$ trivially plugs in a recent strong result [Chakrabarti and Regev 2011] on the GHD (Gap-Hamming-Distance) problem into an existing reduction [Woodruff 2004] from GHD to UNIONSIZE. Consider any given $\epsilon > 0$. In $\text{GHD}_{\frac{2}{5\epsilon^2}}$, Alice and Bob have binary strings X and Y as inputs, respectively, where each string has $\frac{2}{5\epsilon^2}$ bits. Let $\Delta(X, Y)$ denote the hamming distance between X and Y. Alice and Bob are further given the promise that either $\Delta(X, Y) > \frac{1}{5\epsilon^2} + \frac{1}{\epsilon}$ or $\Delta(X, Y) \leq \frac{1}{5\epsilon^2} - \frac{1}{\epsilon}$. They should output 1 iff $\Delta(X, Y)$ satisfies the first inequality. It is proved recently by Chakrabarti and Regev [2011] that $\text{NFT}_{0,\frac{1}{3}}(\text{GHD}_{\frac{2}{5\epsilon^2}}) = \Omega(\frac{1}{\epsilon^2})$.

To reduce GHD to UNIONSIZE, given input string X for $\text{GHD}_{\frac{2}{5\epsilon^2}}$, Alice locally generates an input X' for UNIONSIZE_n by appending 0 to the length-$\frac{2}{5\epsilon^2}$ binary string X until the length reaches n. Bob similarly generates Y'. Let $|X'|$ and $|Y'|$ denote the hamming weight of X' and Y', respectively. Thus, we have $\text{UNIONSIZE}(X', Y') = (|X'| + |Y'| + \Delta(X', Y'))/2 = (|X| + |Y| + \Delta(X, Y))/2$. Leveraging the promise on X and Y, we have the following.

—If $\Delta(X, Y) > \frac{1}{5\epsilon^2} + \frac{1}{\epsilon}$, then $\text{UNIONSIZE}(X', Y') > (|X| + |Y| + \frac{1}{5\epsilon^2} + \frac{1}{\epsilon})/2$.
—If $\Delta(X, Y) \leq \frac{1}{5\epsilon^2} - \frac{1}{\epsilon}$, then $\text{UNIONSIZE}(X', Y') \leq (|X| + |Y| + \frac{1}{5\epsilon^2} - \frac{1}{\epsilon})/2$.

One can easily verify that for all ϵ, we have $(1 + \epsilon)(|X| + |Y| + \frac{1}{5\epsilon^2} - \frac{1}{\epsilon})/2 < (1 - \epsilon)(|X| + |Y| + \frac{1}{5\epsilon^2} + \frac{1}{\epsilon})/2$. Next Bob tells Alice the size of Y, using $\log |Y| = O(\log \frac{1}{\epsilon})$ bits. Alice can now pick any value between these two values as the threshold. Alice outputs 1 iff the $\text{UNIONSIZE}(X', Y')$ execution returns a value above the threshold. Finally, Alice informs Bob of the result, using a single bit. (This is needed since GHD requires both Alice and Bob to know the result.) Since $\text{NFT}_{0,\frac{1}{3}}(\text{GHD}_{\frac{2}{5\epsilon^2}}) = \Omega(\frac{1}{\epsilon^2})$, we have $\text{NFT}_{\epsilon,\frac{1}{3}}(\text{UNIONSIZE}_n) = \Omega(\frac{1}{\epsilon^2} - \log\frac{1}{\epsilon} - 1) = \Omega(\frac{1}{\epsilon^2})$. Now, by Lemma A.1, we have

$$\text{NFT}_0(\text{UNIONSIZE}_n) \geq \frac{1}{3}\text{NFT}_{0,\frac{1}{3}}(\text{UNIONSIZE}_n) \geq \frac{1}{3}\text{NFT}_{\frac{1}{\sqrt{n}},\frac{1}{3}}(\text{UNIONSIZE}_n) = \Omega(n).$$

Finally, we still need to cover the case for $\epsilon = \Omega(\frac{1}{\sqrt{n}})$ but $\epsilon < \frac{1}{\sqrt{n}}$. For such ϵ (which is necessarily $\Theta(\frac{1}{\sqrt{n}})$), we have

$$\text{NFT}_{\epsilon,\frac{1}{3}}(\text{UNIONSIZE}_n) \geq \text{NFT}_{\frac{1}{\sqrt{n}},\frac{1}{3}}(\text{UNIONSIZE}_n) = \Omega(n) = \Omega\left(\frac{1}{\epsilon^2}\right). \quad \square$$

Combining Theorem A.3 and Lemma A.2 gives the following.

COROLLARY A.4. *Let* poly(n) *be any polynomial of* n *with constant degree, we have* $\text{NFT}_0(\text{UNIONSIZE}_n, O(\text{poly}(n))) = \Omega(\frac{n}{\log n})$ *and* $\text{NFT}_{\epsilon,\frac{1}{3}}(\text{UNIONSIZE}_n, O(\text{poly}(n))) = \Omega(\frac{1}{\epsilon^2 \log n})$ *for* $\epsilon = \Omega(\frac{1}{\sqrt{n}})$.

B. TREE-AGGREGATION PROTOCOL WITH $O(\log\frac{1}{\epsilon} + \log\log N)$ AGGREGATION MESSAGE SIZE

This section provides the details of the tree-aggregation protocol with only $O(\log\frac{1}{\epsilon} + \log\log N)$ aggregation message size, as mentioned in Section 4.

Protocol Intuition. First, we should note that directly encoding each partial sum with $O(\log\frac{1}{\epsilon} + \log\log N)$ bits using a floating-point-style representation will not actually work, due to underflow issues when sequentially adding many small numbers to a large number. Thus, instead, we will apply a similar trick as AMS synopsis [Alon et al. 1996]. Intuitively in this protocol, each "1" value in the system is *flagged* with a certain probability. The system then uses the simple tree-aggregation protocol from Section 4 to determine the exact total number (sum) of such flagged "1" values. By properly adjusting the flagging probability, we can always ensure that this sum is no larger than $120/\epsilon^2$, and thus the size of the aggregation message will be no larger than $\log(120/\epsilon^2)$. Furthermore, it is possible to dynamically adjust such flagging probability in one pass of the aggregation protocol, without any global coordination. Finally, the root estimates the final result for SUM based on the sum of flagged "1" values and the associated flagging probability.

ALGORITHM 1: promote(*msg*)

$msg.level++$;
Initialize *tmp* to 0;
for $j = 1$ to $msg.sum$ **do**
    Increase *tmp* by 1 with probability 1/2;
end for
$msg.sum = tmp$;

ALGORITHM 2: merge(msg_1, msg_2) // assuming $msg_1.level \le msg_2.level$

while $msg_1.level < msg_2.level$ **do**
    promote(msg_1);
end while
$msg_3.level = msg_2.level$;
$msg_3.sum = msg_1.sum + msg_2.sum$;
while $msg_3.sum > 120/\epsilon^2$ **do**
    promote(msg_3);
end while
return msg_3;

Protocol Description and Pseudocode. Specifically in this protocol, each aggregation message contains an integer $sum \in [0, 120/\epsilon^2]$ and an integer $level \in [0, \log N]$. Intuitively, these two integers mean that if each "1" value in the subtree is flagged with probability 2^{-level}, then the partial sum of the flagged values is *sum*. A node with a value of 1 generates an aggregation message with $sum = 1$ and $level = 0$, for its own value. Intermediate tree nodes will need to combine multiple aggregation messages into one. Without loss of generality, we only need to explain how to combine two aggregation messages msg_1 and msg_2 into one, where $msg_1.level \le msg_2.level$. We *promote* (Algorithm 1) an aggregation message msg_1, by (i) increasing $msg_1.level$ by one, and (ii) tossing $msg_1.sum$ fair coins and then updating $msg_1.sum$ to be the total number of heads we observe. To merge msg_1 and msg_2 into msg_3 (Algorithm 2), we first repeatedly promote msg_1, until $msg_1.level = msg_2.level$. We then set $msg_3.level = msg_2.level$, and $msg_3.sum = msg_1.sum + msg_2.sum$. If $msg_3.sum > 120/\epsilon^2$, we will again repeatedly promote msg_3 until the first time that $msg_3.sum \le 120/\epsilon^2$. Finally, imagine that the root has a virtual parent and let *msg* be the aggregation message sent by the root to its virtual parent. The root will estimate the final sum to be $msg.sum \times 2^{msg.level}$.

Formal Properties. It is obvious that the number of bits sent by each node in this protocol is $O(\log \frac{1}{\epsilon} + \log\log N)$. We next prove that the protocol does give us an $(\epsilon, 1/3)$-approximate result.

THEOREM B.1. *Consider any graph G with N nodes and any constant $\epsilon \in (0, 1]$. Let s denote the exact sum of the values of all the N nodes and $\hat{s}$ denote output of this protocol. We have*

$$\Pr[(1-\epsilon)s \leq \hat{s} \leq (1+\epsilon)s] \geq \frac{2}{3}.$$

PROOF. Consider the sequence of random variables $S_0, S_1, \ldots,$ where $S_0 = s$ and S_{i+1} (for $i \geq 0$) is the number of heads observed when flipping a fair coin exactly S_i times. Furthermore, for generating S_{i+1}, the random process uses the same coin flip results as the protocol uses in promoting all messages with $level = i$ (i.e., at Line 4 of Algorithm 1). Let random variable L be the smallest integer such that $S_L \leq z$ where $z = \frac{120}{\epsilon^2}$. Let *msg* be the aggregation message sent by the root to its virtual parent. We claim that $msg.level = L$ and $msg.sum = S_L$. First, it is impossible for $msg.level < L$, since otherwise *msg.sum* will be above z and thus the *msg* will be promoted by the root. Next if $msg.level > L$, it means that some node must have observed a message msg' whose level is L, and has further promoted msg'. But this is impossible since if $msg'.level = L$, then $msg'.sum \leq S_L \leq z$ by our definition of L. Now given that $msg.level = L$, we have $msg.sum = S_L$.

Let $l = \lfloor \log_2 \frac{3s}{4z} \rfloor$, and we have $2^l \in [\frac{3s}{8z}, \frac{3s}{4z}]$ and $2^{l+2} \in [\frac{3s}{2z}, \frac{3s}{z}]$. Since for all $i \geq 0$, S_i is a binomial random variable with parameter $(s, 2^{-i})$, we have

$$\mathrm{E}[S_l] = 2^{-l}s \geq \frac{4}{3}z \quad \text{and} \quad \mathrm{VAR}[S_l] \leq \frac{8}{3}z$$
$$\mathrm{E}[S_{l+2}] = 2^{-l-2}s \leq \frac{2}{3}z \quad \text{and} \quad \mathrm{VAR}[S_{l+2}] \leq \frac{2}{3}z.$$

We claim that with probability at most $\frac{1}{4}$, $L \notin [l+1, l+2]$, since by Chebyshev's inequality:

$$\Pr[L \leq l] \;=\; \Pr[S_l \leq z] \leq \frac{24}{z} \leq \frac{1}{5}$$
$$\Pr[L > l+2] \;=\; \Pr[S_{l+2} > z] \leq \frac{6}{z} \leq \frac{1}{20}.$$

Denote $\mathcal{E}_i$ as the event $2^i S_i \notin [(1-\epsilon)s, (1+\epsilon)s]$, and we claim that for any $i \leq l+2$, $\Pr[\mathcal{E}_i] \leq \frac{1}{40}$. Since S_i is a binomial random variable with parameter $(s, 2^{-i})$, We have $\mathrm{E}[2^i S_i] = s$ and $\mathrm{VAR}[2^i S_i] \leq 2^{2i}2^{-i}s = 2^i s$. By Chebyshev's inequality, we have $\Pr[\mathcal{E}_i] = 1 - \Pr[2^i S_i \in [(1-\epsilon)s, (1+\epsilon)s]] \leq \frac{2^i}{\epsilon^2 s} \leq \frac{3}{z\epsilon^2} = \frac{1}{40}$. Next, denote $\mathcal{E}$ as the event that $\hat{s} \notin [(1-\epsilon)s, (1+\epsilon)s]$, or equivalently $2^L S_L \notin [(1-\epsilon)s, (1+\epsilon)s]$. We have

$$\begin{aligned}
\Pr[\mathcal{E}] \;&=\; \sum_i \Pr[L = i] \Pr[\mathcal{E}|L = i] \\
&=\; \sum_{i \in [l+1, l+2]} \Pr[L = i] \Pr[\mathcal{E}|L = i] + \sum_{i \notin [l+1, l+2]} \Pr[L = i] \Pr[\mathcal{E}|L = i] \\
&\leq\; \Pr[L = l+1] \Pr[\mathcal{E}_{l+1}|L = l+1] + \Pr[L = l+2] \Pr[\mathcal{E}_{l+2}|L = l+2] \\
&\quad + \sum_{i \notin [l+1, l+2]} \Pr[L = i] \\
&\leq\; \Pr[\mathcal{E}_{l+1} \text{ and } L = l+1] + \Pr[\mathcal{E}_{l+2} \text{ and } L = l+2] + \frac{1}{4} \\
&\leq\; \Pr[\mathcal{E}_{l+1}] + \Pr[\mathcal{E}_{l+2}] + \frac{1}{4} \leq \frac{1}{20} + \frac{1}{4} < \frac{1}{3}. \quad \square
\end{aligned}$$

C. PROOF FOR THEOREM 6.2 FROM SECTION 6.3

Throughout this section, we use the alternative form of the cycle promise as defined in Section 6.3 and as illustrated in Figure 10.

C.1. A Simple Reduction from DISJOINTNESSCP to UNIONSIZECP

To lower bound the communication complexity of UNIONSIZECP and prove Theorem 6.2, we will reduce from a new DISJOINTNESSCP problem, which is the natural extension of the standard DISJOINTNESS problem on binary strings [Kushilevitz and Nisan 1996]. In DISJOINTNESSCP$_{n,q}$ ($q \geq 2$), again Alice and Bob each have a length-n string X and Y as input, where the characters in the strings are integers in $[0, q-1]$. X and Y here satisfy the alternative form of the cycle promise as in UNIONSIZECP. Alice and Bob aim to determine whether there exists any i where $X_i = 0$ and $Y_i = 0$, and they output 0 iff there exists such i. For convenience later, different from UNIONSIZECP, for DISJOINTNESSCP we require both Alice and Bob to know the final result. Recall from Appendix A that the notation $\text{NFT}_{0,\delta}$ simply means $\text{NFT}_{\epsilon,\delta}$ with $\epsilon = 0$. The next section will prove the following theorem on the communication complexity of DISJOINTNESSCP, via an information theoretic approach [Bar-Yossef et al. 2004].

THEOREM C.1.

$$\text{NFT}_0(\text{DISJOINTNESSCP}_{n,q}) = \Omega\left(\frac{n}{q^2}\right) - O(\log n) \textit{ and}$$
$$\text{NFT}_{0,\frac{1}{5}}(\text{DISJOINTNESSCP}_{n,q}) = \Omega\left(\frac{n}{q^2}\right) - O(\log n).$$

Using this theorem, one can obtain a lower bound on the communication complexity of UNIONSIZECP, under the setting without synchronized rounds, via a direct reduction.

THEOREM C.2.

$$\text{NFT}_0(\text{UNIONSIZECP}_{n,q}) = \Omega\left(\frac{n}{q^2}\right) - O(\log n) \textit{ and}$$
$$\text{NFT}_{\epsilon,\frac{1}{5}}(\text{UNIONSIZECP}_{n,q}) = \Omega\left(\frac{1}{\epsilon q^2}\right) - O\left(\log\frac{1}{\epsilon}\right) \;\; \textit{for } \epsilon = \Omega\left(\frac{1}{\sqrt{n}}\right).$$

PROOF. We first prove the theorem for $\epsilon \geq \frac{1}{\sqrt{2n}}$. $\text{NFT}_0(\text{UNIONSIZECP}_{n,q}) = \Omega(\frac{n}{q^2}) - O(\log n)$ follows from a reduction from DISJOINTNESSCP$_{n,q}$. Consider any given protocol for UNIONSIZECP$_{n,q}$. Given inputs X and Y to DISJOINTNESSCP$_{n,q}$, Alice and Bob directly invoke the protocol for UNIONSIZECP$_{n,q}$, with X and Y being the inputs. Alice outputs 1 iff UNIONSIZECP$_{n,q}$ returns n. Alice further sends Bob a single bit to inform Bob of this result. We have

$$\text{NFT}_0(\text{UNIONSIZECP}_{n,q}) \geq \text{NFT}_0(\text{DISJOINTNESSCP}_{n,q}) - 1 = \Omega\left(\frac{n}{q^2}\right) - O(\log n).$$

$\text{NFT}_{\epsilon,\frac{1}{5}}(\text{UNIONSIZECP}_{n,q}) = \Omega(\frac{1}{\epsilon q^2}) - O(\log\frac{1}{\epsilon})$ follows from a reduction from DISJOINTNESSCP$_{\frac{1}{2\epsilon},q}$. Consider any given protocol for UNIONSIZECP$_{n,q}$. Given a length-$\frac{1}{2\epsilon}$ input X for DISJOINTNESSCP$_{\frac{1}{2\epsilon},q}$, Alice locally generates a length-n input X' by first replicating each character in X for $\frac{1}{\epsilon}$ times, and then appending 0 until the length of X' reaches n. This is always possible since $\frac{1}{2\epsilon^2} \leq n$. Bob generates Y' in a similar way. We now have the following.

—If DISJOINTNESSCP$_{\frac{1}{2\epsilon},q}(X, Y) = 1$, then UNIONSIZECP$_{n,q}(X', Y') = \frac{1}{2\epsilon^2}$.
—If DISJOINTNESSCP$_{\frac{1}{2\epsilon},q}(X, Y) = 0$, then UNIONSIZECP$_{n,q}(X', Y') \leq \frac{1}{2\epsilon^2} - \frac{1}{\epsilon}$.

One can easily verify that for all $\epsilon > 0$, we have $(1+\epsilon)(\frac{1}{2\epsilon^2} - \frac{1}{\epsilon}) < (1-\epsilon)\frac{1}{2\epsilon^2}$. Alice can now pick any value between $(1+\epsilon)(\frac{1}{2\epsilon^2} - \frac{1}{\epsilon})$ and $(1-\epsilon)\frac{1}{2\epsilon^2}$ as the threshold. Alice outputs 1 for $\text{DISJOINTNESSCP}_{\frac{1}{2\epsilon},q}(X, Y)$ iff $\text{UNIONSIZECP}_{n,q}(X', Y')$ returns a value above that threshold. Finally, Alice sends Bob a single bit to inform Bob of the result. We thus have

$$\text{NFT}_{\epsilon,\frac{1}{5}}(\text{UNIONSIZECP}_{n,q}) \geq \text{NFT}_{0,\frac{1}{5}}(\text{DISJOINTNESSCP}_{\frac{1}{2\epsilon},q}) - 1 = \Omega\left(\frac{1}{\epsilon q^2}\right) - O\left(\log\frac{1}{\epsilon}\right).$$

We still need to cover the case for $\epsilon = \Omega(\frac{1}{\sqrt{n}})$ but $\epsilon < \frac{1}{\sqrt{2n}}$. For such ϵ (which is necessarily $\Theta(\frac{1}{\sqrt{n}})$), we have

$$\begin{aligned}\text{NFT}_{\epsilon,\frac{1}{5}}(\text{UNIONSIZECP}_{n,q}) &\geq \text{NFT}_{\frac{1}{\sqrt{2n}},\frac{1}{5}}(\text{UNIONSIZECP}_{n,q}) \\ &= \Omega\left(\frac{\sqrt{n}}{q^2}\right) - O(\log n) \\ &= \Omega\left(\frac{1}{\epsilon q^2}\right) - O\left(\log\frac{1}{\epsilon}\right). \quad \square\end{aligned}$$

Theorem 6.2 then directly follows from Theorem C.2 and Lemma A.2.

C.2. Proving Theorem C.1 via Information Cost

This section proves Theorem C.1 for the DISJOINTNESSCP problem. For convenience in the proofs later, we define DISJOINTNESSCP more formally as follows.

Definition C.3 (DISJOINTNESSCP). In the $\text{DISJOINTNESSCP}_{n,q}$ problem, Alice and Bob hold X and Y, respectively, which are two strings of length n satisfying $(X, Y) \in \mathcal{L}_q^n$, where

$$\mathcal{L}_q^n = \{(X, Y) \mid X \in \mathbb{Z}_q^n \text{ and } Y \in \mathbb{Z}_q^n \text{ and } (Y - X) \in \{0, 1\}^n\}.$$

The goal is to compute the function $\text{DISJOINTNESSCP}_{n,q} : \mathcal{L}_q^n \rightarrow \{0, 1\}$ defined as

$$\text{DISJOINTNESSCP}_{n,q}(X, Y) = \begin{cases} 0 & \exists i \in \{1, 2, \ldots, n\} \text{ such that } X_i = Y_i = 0 \\ 1 & \text{otherwise.} \end{cases}$$

Our proof for Theorem C.1 will be almost entirely based on the information theoretic approach from Bar-Yossef et al. [2004]. In this approach, the *information complexity* of a function is used to lower bound the communication complexity of that function. Under certain conditions, it is further shown that the *conditional information complexity* of a function is a lower bound on the function's information complexity. Next, under certain conditions, the approach establishes a direct-sum result between the conditional information complexity of a function (e.g., $\text{DISJOINTNESSCP}_{n,q}$) and the conditional information complexity of its constituent primitive function (e.g., $\text{DISJOINTNESSCP}_{1,q}$). Finally, the approach also provides some tools for reasoning about the conditional information complexity of such constituent primitive functions. The final lower bound on communication complexity obtained via this approach is for private-coin randomized protocols only. Since we will need a lower bound for public-coin protocols, at the end of this section, we will apply the well-known result from Newman [1991] to convert this lower bound to a public-coin setting. Recall from Appendix A that the notation $\text{NFT}_{0,\delta}$ simply means $\text{NFT}_{\epsilon,\delta}$ with $\epsilon = 0$. We define $\text{NFT}^{\text{pri}}_{0,\delta}(\text{DISJOINTNESSCP}_{n,q})$ to be the same as $\text{NFT}_{0,\delta}(\text{DISJOINTNESSCP}_{n,q})$, except that $\text{NFT}^{\text{pri}}_{0,\delta}$ is for private-coin protocols.

In the next, we first summarize the definitions and lemmas that we will use in this information theoretic approach. All these definitions (Definition C.4 to C.8) and lemmas

(Lemma C.9 to C.12) are directly adapted from Bar-Yossef et al. [2004], and are not our contribution. See Bar-Yossef et al. [2004] for a more detailed discussion.

Definition C.4 (*Decomposable Functions*). *(Adapted from Bar-Yossef et al. [2004]).* If there are functions $h : \mathcal{L}_q^1 \rightarrow \{0, 1\}$ and $g : \{0, 1\}^n \rightarrow \{0, 1\}$ such that $\text{DISJOINTNESSCP}_{n,q}(X, Y) = g(h(X_1, Y_1), h(X_2, Y_2), \ldots, h(X_n, Y_n))$, then we say that $\text{DISJOINTNESSCP}_{n,q}$ is *g-decomposable with primitive h*. When the context is clear, we simply say that $\text{DISJOINTNESSCP}_{n,q}$ is *decomposable with primitive h*.

We construct g as $g(x_1, x_2, \ldots, x_n) = \Pi_{i=1}^n x_i$. Then, according to this definition, $\text{DISJOINTNESSCP}_{n,q}$ is a decomposable function with primitive $h = \text{DISJOINTNESSCP}_{1,q}$. For convenience, from now on in this section, h stands for the function $\text{DISJOINTNESSCP}_{1,q}$. Namely $h(x, y) = 0$ if $x = y = 0$, otherwise, $h(x, y) = 1$.

Definition C.5 (*Mixture of Product Distributions*). *(Adapted from Bar-Yossef et al. [2004]).* For random variables X_i, Y_i, and T_i $(1 \le i \le n)$, their joint distribution (X_i, Y_i, T_i) is called a *mixture of product distribution* if conditioned on T_i, X_i and Y_i are independent.

Let $\mathcal{T} = \{0, 1\} \times \{1, 2, \ldots, q-1\}$, and let T_i $(1 \le i \le n)$ be a random variable drawn uniformly randomly from $\mathcal{T}$. Let X_i and Y_i $(1 \le i \le n)$ be two random variables depending on T_i where:

—if $T_i = (0, j)$, then $X_i = j$ and Y_i is drawn uniformly randomly from $\{j, j+1\}$;
—if $T_i = (1, j)$, then $Y_i = j$ and X_i is drawn uniformly randomly from $\{j, j-1\}$.

Here all additions and subtractions are on $\mathbb{Z}_q$. Note that under this construction, it is impossible for $X_i = Y_i = 0$, which is intentional. Define ζ as the joint distribution of (X_i, Y_i, T_i). Clearly, conditioned on T_i, X_i, and Y_i are independent. Hence, ζ is a mixture of product distribution for all i's $(1 \le i \le n)$.

Definition C.6 (*Collapsing Distribution*). *(Adapted from Bar-Yossef et al. [2004]).* A distribution on $\mathcal{L}_q^n$ is called a *collapsing distribution* for $\text{DISJOINTNESSCP}_{n,q}$ with respect to h, if $\text{DISJOINTNESSCP}_{n,q}$ is g-decomposable with primitive h, and if for all (X, Y)'s in the support of that distribution, all j's where $1 \le j \le n$, and all $(x, y) \in \mathcal{L}_q^1$, the following holds:

$$g(h(X_1, Y_1), \ldots, h(X_{j-1}, Y_{j-1}), h(x, y), h(X_{j+1}, Y_{j+1}), \ldots, h(X_n, Y_n)) = h(x, y).$$

Define $\eta = \zeta^n$, and let $(X, Y, T) \sim \eta$. Consider the marginal distribution η_{XY} of (X, Y) in η. Since $(X_i, Y_i, T_i) \sim \zeta$, X_i and Y_i cannot simultaneously be 0, which means $h(X_i, Y_i) = 1$. Hence, for all (X, Y)'s in the support of η_{XY}, we have $h(X_i, Y_i) = 1$ for all i's. This implies that $g(\ldots, h(x, y), \ldots) = h(x, y)$, and thus η_{XY} is a collapsing distribution for $\text{DISJOINTNESSCP}_{n,q}$.

Definition C.7 (*Conditional Information Cost*). *(Adapted from Bar-Yossef et al. [2004]).* Let $\mathcal{P}$ be any two-party private-coin randomized protocol for $\text{DISJOINTNESSCP}_{1,q}$. Let $(X_i, Y_i, T_i) \sim \zeta$, which is a mixture of product distributions on $\mathcal{L}_q^1 \times \mathcal{T}$. Given X_i and Y_i as the input to $\mathcal{P}$, the transmitted messages in $\mathcal{P}$ can be viewed as a random variable $\mathcal{P}(X_i, Y_i)$. The *conditional information cost* of $\mathcal{P}$ with respect to ζ (denoted as $\text{CIC}_\zeta(\mathcal{P})$) is the mutual information between (X_i, Y_i) and $\mathcal{P}(X_i, Y_i)$ conditioned on T_i. Or formally:

$$\text{CIC}_\zeta(\mathcal{P}) = \sum_{t \in \mathcal{T}} I(\{(X_i, Y_i); \mathcal{P}(X_i, Y_i)\} | T_i = t) \Pr[T_i = t].$$

Here I stands for the standard notion of conditional mutual information [Bar-Yossef et al. 2004].

Definition C.8 (*Conditional Information Complexity*). *(Adapted from Bar-Yossef et al. [2004]).* Let $\mathcal{P}$ be any two-party private-coin randomized protocol for DISJOINTNESSCP$_{1,q}$, such that for any input (x, y), $\mathcal{P}$ can generate the correct result with probability at least $1 - \delta$. The *δ-error conditional information complexity of* DISJOINTNESSCP$_{1,q}$ *with respect to* ζ, denoted as $\mathrm{CIC}_{\zeta,\delta}$(DISJOINTNESSCP$_{1,q}$), is defined as the minimum conditional information cost across all possible $\mathcal{P}$'s satisfying the earlier property.

LEMMA C.9 (ADAPTED FROM BAR-YOSSEF ET AL. [2004]). *Consider* DISJOINTNESSCP$_{n,q}$*, and the distribution ζ, η, and η_{XY} as defined earlier. We already know* DISJOINTNESSCP$_{n,q}$ *is a decomposable function with primitive* DISJOINTNESSCP$_{1,q}$*, ζ is a mixture of product distribution on $\mathcal{L}_q^1 \times \mathcal{T}$, and η_{XY} is a collapsing distribution for* DISJOINTNESSCP$_{n,q}$ *with respect to* DISJOINTNESSCP$_{1,q}$*. We must have:*

$$\mathrm{NFT}_{0,\delta}^{\mathrm{pri}}(\text{DISJOINTNESSCP}_{n,q}) \geq n \times \mathrm{CIC}_{\zeta,\delta}(\text{DISJOINTNESSCP}_{1,q}).$$

LEMMA C.10 (ADAPTED FROM BAR-YOSSEF ET AL. [2004]). *Let Z be a random variable uniformly randomly distributed on $\{z_1, z_2\}$, and let $\Phi(z_1)$ and $\Phi(z_2)$ be two additional random variables. If $\Phi(z_1)$ and $\Phi(z_2)$ are both independent of Z, then we have*

$$I(Z; \Phi(Z)) \geq H^2(\Phi_{z_1}, \Phi_{z_2}).$$

Here Φ_{z_i} is the distribution of $\Phi(z_i)$, and H is the Hellinger distance [Bar-Yossef et al. 2004] between two distributions.

LEMMA C.11 (ADAPTED FROM BAR-YOSSEF ET AL. [2004]). *For any two-party private-coin randomized protocol $\mathcal{P}$, let random variable $\mathcal{P}(x, y)$ denote the transmitted message in $\mathcal{P}$ under input x and y. Let $\mathcal{P}_{x,y}$ denote the distribution of $\mathcal{P}(x, y)$. For all x, x', y, and y', we have*

$$2H^2(\mathcal{P}_{x,y}, \mathcal{P}_{x',y'}) \geq H^2(\mathcal{P}_{x,y}, \mathcal{P}_{x',y}) + H^2(\mathcal{P}_{x,y'}, \mathcal{P}_{x',y'}).$$

LEMMA C.12 (ADAPTED FROM BAR-YOSSEF ET AL. [2004]). *Let $\mathcal{P}$ be any private-coin randomized protocol for* DISJOINTNESSCP$_{1,q}$*, such that for any input (x, y), $\mathcal{P}$ can generate the correct result with probability at least $1 - \delta$. For any two input pairs $(x, y) \in \mathcal{L}_q^1$ and $(x', y') \in \mathcal{L}_q^1$ where* DISJOINTNESSCP$_{1,q}(x, y) \neq$ DISJOINTNESSCP$_{1,q}(x', y')$*, we have*

$$H^2(\mathcal{P}_{x,y}, \mathcal{P}_{x',y'}) \geq 1 - 2\sqrt{\delta}.$$

Having introduced the definitions and lemmas needed for the information theoretic arguments, we are now ready to prove Theorem C.1. We start with the following theorem on the communication complexity of DISJOINTNESSCP for private coin protocols.

THEOREM C.13. $\mathrm{NFT}_{0,\delta}^{\mathrm{pri}}(\text{DISJOINTNESSCP}_{n,q}) = \Omega(\frac{n}{q^2})$ *for any positive constant* $\delta \leq 0.22$.

PROOF. Let $\mathcal{P}$ denote the optimal protocol with the minimum conditional information cost, across all possible two-party private-coin randomized protocols for DISJOINTNESSCP$_{1,q}$ where for any input (x, y), the protocol can always generate the correct result with probability at least $1 - \delta$.

By Lemma C.9 and Definition C.7 and C.8, we have

$$\begin{aligned}
&\mathrm{NFT}^{\mathrm{pri}}_{0,\delta}(\textsc{DisjointnessCP}_{n,q})\\
&\geq n \times \mathrm{CIC}_{\zeta,\delta}(\textsc{DisjointnessCP}_{1,q})\\
&= n \times \mathrm{CIC}_{\zeta}(\mathcal{P})\\
&= \frac{n}{2(q-1)} \sum_{t\in\mathcal{T}} I(\{X_1, Y_1; \mathcal{P}(X_1, Y_1)\} \mid T_1 = t)\\
&= \frac{n}{2(q-1)} \sum_{j=1}^{q-1} (I(\{X_1, Y_1; \mathcal{P}(X_1, Y_1)\} \mid T_1 = (0, j)) + I(\{X_1, Y_1; \mathcal{P}(X_1, Y_1)\} \mid T_1 = (1, j))).
\end{aligned}$$

Conditioned on $T_1 = (0, j)$, (X_1, Y_1) is uniformly distributed on $\{(j, j), (j, j+1)\}$. Let $z_1 = (j, j)$, $z_2 = (j, j+1)$, and $Z = (X_1, Y_1)$. Lemma C.10 tells us:

$$I(\{X_1, Y_1; \mathcal{P}(X_1, Y_1)\} \mid T_1 = (0, j)) \geq H^2(\mathcal{P}_{j,j}, \mathcal{P}_{j,j+1}).$$

Similarly, we have

$$I(\{X_1, Y_1; \mathcal{P}(X_1, Y_1)\} \mid T_1 = (1, j)) \geq H^2(\mathcal{P}_{j,j}, \mathcal{P}_{j-1,j}).$$

Apply Cauchy inequality and triangle inequality, and we have

$$\begin{aligned}
&\mathrm{NFT}^{\mathrm{pri}}_{0,\delta}(\textsc{DisjointnessCP}_{n,q})\\
&\geq \frac{n}{2(q-1)} \sum_{j=1}^{q-1} \left(H^2(\mathcal{P}_{j,j}, \mathcal{P}_{j,j+1}) + H^2(\mathcal{P}_{j,j}, \mathcal{P}_{j-1,j})\right)\\
&\geq \frac{n}{4(q-1)^2} \left(\sum_{j=1}^{q-1} (H(\mathcal{P}_{j,j}, \mathcal{P}_{j,j+1}) + H(\mathcal{P}_{j,j}, \mathcal{P}_{j-1,j}))\right)^2\\
&\geq \frac{n}{4(q-1)^2} \left(\sum_{j=1}^{q-1} H(\mathcal{P}_{j,j+1}, \mathcal{P}_{j-1,j})\right)^2\\
&= \frac{n}{4(q-1)^2} (H(\mathcal{P}_{1,2}, \mathcal{P}_{0,1}) + H(\mathcal{P}_{2,3}, \mathcal{P}_{1,2}) + \cdots + H(\mathcal{P}_{q-1,0}, \mathcal{P}_{q-2,q-1}))^2\\
&\geq \frac{n}{4(q-1)^2} H^2(\mathcal{P}_{q-1,0}, \mathcal{P}_{0,1}).
\end{aligned}$$

Next, apply Lemma C.11, and we have

$$\begin{aligned}
\mathrm{NFT}^{\mathrm{pri}}_{0,\delta}(\textsc{DisjointnessCP}_{n,q}) &\geq \frac{n}{8(q-1)^2} H^2(\mathcal{P}_{q-1,0}, \mathcal{P}_{0,0}) + \frac{n}{8(q-1)^2} H^2(\mathcal{P}_{q-1,1}, \mathcal{P}_{0,1})\\
&\geq \frac{n}{8(q-1)^2} H^2(\mathcal{P}_{q-1,0}, \mathcal{P}_{0,0}).
\end{aligned}$$

Finally, apply Lemma C.12, and we have

$$\mathrm{NFT}^{\mathrm{pri}}_{0,\delta}(\textsc{DisjointnessCP}_{n,q}) \geq \frac{n}{8(q-1)^2}(1 - 2\sqrt{\delta}) = \Omega\left(\frac{n}{q^2}\right). \quad \square$$

We can now prove Theorem C.1.

PROOF FOR THEOREM C.1. According to Newman [1991],[14] we have

$$\mathrm{NFT}^{\mathrm{pri}}_{0,0.22}(\textsc{DisjointnessCP}_{n,q}) \leq \mathrm{NFT}_{0,0.2}(\textsc{DisjointnessCP}_{n,q}) + O(\log n + \log\log q).$$

Apply Theorem C.13 and we have

$$\mathrm{NFT}_{0,0.2}(\textsc{DisjointnessCP}_{n,q}) = \Omega\left(\frac{n}{q^2}\right) - O\left(\log n + \log\log q\right).$$

This lower bound is only nontrivial when $q < \sqrt{\frac{n}{\log n}}$. Thus, we can discard the $\log\log q$ term for clarity:

$$\mathrm{NFT}_{0,0.2}(\textsc{DisjointnessCP}_{n,q}) = \Omega\left(\frac{n}{q^2}\right) - O\left(\log n\right).$$

Finally, apply Lemma A.1 and we have $\mathrm{NFT}_0(\textsc{DisjointnessCP}_{n,q}) = \Omega(\frac{n}{q^2}) - O(\log n)$ as well. □

D. PROOF FOR LEMMA 7.4 FROM SECTION 7

This appendix will prove Lemma 7.4. The proof is elementary but involved, and we need much preparation work in Sections D.1 and D.2 before actually proving the lemma in Section D.3. Recall from Section 7 the concepts of oblivious reduction, reference setting, reference execution, assignment graph, the definitions of b', $\mathcal{X}(i)$, $\mathcal{Y}(i)$, and $\Phi(X, Y)$, the concept of a node being disconnected from the root in a given execution, and also the problematic input set $\mathcal{I}$ as constructed in the proof of Lemma 7.3.

D.1. Node α and β Must Remain Unspoiled

This section proves that node α (β) must remain unspoiled for Alice (Bob) in an oblivious reduction. To do so, we inherit the formal framework developed in Section 5.3. Since for each input pair $(X, Y) \in \mathcal{L}$, there is a corresponding reference setting in the oblivious reduction, the notions of values and failure time of nodes in Section 5.3 are still well defined here. Namely, all we need to do is to replace the notion of "simulated execution under (X, Y)" in Section 5.3 by the notion of "reference setting for (X, Y)".[15] All concepts defined in Section 5.3 (e.g., spoiled nodes) now carry over directly without modification. For example, a node v is a value epicenter for Alice's input X if its value in the reference setting is not uniquely determined by X.

Lemma 5.2 in Section 5.3 proved that Alice can simulate all unspoiled nodes. In this appendix, we intend to prove the reverse for oblivious reductions—if a node is spoiled for Alice in a round r, then in an oblivious reduction, Alice can never invoke the oracle protocol on that node for round r. Since in an oblivious reduction Alice (Bob) is required to invoke the oracle on node α (β) throughout the execution, this in turn implies that α (β) must remain unspoiled for Alice (Bob). Our proof will hinge upon the property of oblivious reductions, which requires Alice (Bob) to decide, beforehand, exactly up to which rounds she (he) will invoke the oracle on each node.

LEMMA D.1. *Consider any oblivious reduction from Π to* SUM. *If a node v in G is spoiled for Alice's input X (Bob's input Y) in round $r' \geq 0$, then when Alice (Bob) has the input X (Y), Alice (Bob) will not invoke the oracle on v for round r'.*

[14] Newman's original result was only stated for functions, while here we are dealing with the partial functions of DISJOINTNESSCP. Nevertheless, Newman's original proof actually holds without modification to partial functions.

[15] These two notions are actually exactly the same. In Section 5.3, there was no need to introduce the more formal notion of reference settings, so there we used the notion of simulated execution.

Proof. We only need to prove the part for Alice. Let $r \in [1, r']$ be the very first round during which v is spoiled. It suffices to prove that Alice will not invoke the oracle on v for round r—since the oracle protocol carries internal state from round to round, Alice can never invoke the oracle for round r' without invoking the oracle for earlier rounds.

If round r is the first round during which v is spoiled, there must exist some epicenter u_0 with an occurrence time of r_0 ($r_0 \leq r$) such that there exists a spoil path from u_0 to v with exactly $l = r - r_0$ hops. To show that Alice will not invoke the oracle on v for round r, we use an induction on l.

If $l = 0$, v itself must be an epicenter occurring at round r. We consider two cases. If v is a value epicenter, then the occurrence time is round 1 and $r = 1$. In an oblivious reduction, Alice needs to decide purely based on X, the input value of each node for which she will invoke the oracle for at least one round. This means that Alice must never invoke the oracle on v—otherwise, she risks deviating from the corresponding execution under the reference setting. Next if v is a failure epicenter, then round r (i.e., the occurrence time of the epicenter) must be the earliest possible failure time. This means that there exists Bob's inputs Y and Y', such that v's failure time is exactly round r in the reference setting for (X, Y) and is after round r in the reference setting for (X, Y'). If Alice decides that she will invoke the oracle on v for round r, then again she risks deviating from the execution under the reference setting since the reference setting could be (X, Y).

For the inductive step, assume that the lemma holds for all values up to l and we consider $l + 1$. Again, there exists some epicenter u_0 with an occurrence time of r_0 ($r_0 \leq r$) such that there exists a spoil path from u_0 to v with exactly $l+1$ hops. Consider the node u immediately before v in this spoil path. Then, the length of the spoil path from u_0 to u is exactly l hops, and u is spoiled in round $r - 1$, where $r - 1 \geq 1$. By the inductive hypothesis, Alice (with an input X) does not invoke the oracle on u for round $r - 1$. Next, we prove via a contradiction and assume that Alice still invokes the oracle on v for round r. Note that in an oblivious reduction, the only way for Alice to obtain the potential message sent in round $r - 1$ by the oracle protocol on u ($u \neq \beta$) is for Alice to invoke the oracle on u for round $r - 1$ herself. Thus, for Alice to still invoke the oracle on v for round r, u must have failed in round $r - 1$ or earlier in all the reference settings for all possible input pairs (X, Y) given the current X. We claim that it is impossible for u to fail exactly in round $r - 1$ in all these reference settings, since otherwise this failure is a stable failure for X, and there would be no spoil path from u_0 to v via u. Thus, there must exist some Y such that u fails before round $r - 1$. This in turn, means that the occurrence time of the epicenter u is round $r - 2$ or earlier. Thus, v is spoiled by u in round $r - 1$ or earlier, which contradicts with the fact that r is the first round that v becomes spoiled. □

Corollary D.2. *Consider any oblivious reduction from* Π *to* Sum. *For any input pair* $(X, Y) \in \mathcal{L}$, α (β) *must remain unspoiled for Alice's input* X *(Bob's input* Y*) throughout the execution of* $\Phi(X, Y)$.

Proof. Trivially follows from Lemma D.1 and the fact that in an oblivious reduction, Alice (Bob) is required to invoke the oracle on α (β) throughout the entire execution. □

D.2. Reasoning about Paths – Some Technical Lemmas

This section proves a series of technical lemmas, which we will later use to prove Lemma 7.4. We start with some useful concepts and definitions, as summarized in Table II.

Some Useful Concepts and Definitions. Throughout this appendix, we use $\mathcal{I}$ to denote the problematic input set as constructed in the proof of Lemma 7.3.

Table II. Notations and Definitions used in Sections D.2 and D.3

$\mathcal{I}$	the problematic input set as constructed in the proof of Lemma 7.3
$\Phi(X, Y)$	the execution of the SUM oracle protocol under the reference setting for (X, Y) and under the given public coin outcomes chosen by Alice and Bob
$\lambda(X, Y)$	number of rounds in each aggregation round in the execution $\Phi(X, Y)$
$F_A(X, v)$	v's failure time in the simulation, if v has a stable failure with respect to Alice's input X
$F_B(Y, v)$	v's failure time in the simulation, if v has a stable failure with respect to Bob's input Y
p	a simple path in the topology G
$\|p\|$	length of the path p
α-path	path from τ to α without passing β
β-path	path from τ to β without passing α
dummy path	a path that will be cut by the end of the execution of $\Phi(X, Y)$ for all $(X, Y) \in \mathcal{I}$.
$p_A(X, t)$ or $p(X, t)$	$\exists v \in p$ such that $F_A(X, v) \leq t\|p\|$
$p_B(Y, t)$ or $p(Y, t)$	$\exists v \in p$ such that $F_B(Y, v) \leq t\|p\|$
$\mathbb{P}^\alpha$	the finite set of all non-dummy α-paths
$\mathbb{P}^\beta$	the finite set of all non-dummy β-paths
$\mathbb{P}^\alpha_{<p}$	the set of all paths in $\mathbb{P}^\alpha$ whose lengths are smaller than the length of p
$\mathbb{P}^\beta_{<p}$	the set of all paths in $\mathbb{P}^\beta$ whose lengths are smaller than the length of p
$\mathbb{P}^\alpha(X, t)$	either $\mathbb{P}^\alpha = \emptyset$ or $p'(X, t)$ holds for all $p' \in \mathbb{P}^\alpha$
$\mathbb{P}^\alpha(Y, t)$	either $\mathbb{P}^\alpha = \emptyset$ or $p'(Y, t)$ holds for all $p' \in \mathbb{P}^\alpha$
$\mathbb{P}^\beta(X, t)$	either $\mathbb{P}^\beta = \emptyset$ or $p'(X, t)$ holds for all $p' \in \mathbb{P}^\beta$
$\mathbb{P}^\beta(Y, t)$	either $\mathbb{P}^\beta = \emptyset$ or $p'(Y, t)$ holds for all $p' \in \mathbb{P}^\beta$
$\mathbb{P}^\alpha_{<p}(X, t)$	either $\mathbb{P}^\alpha_{<p} = \emptyset$ or $p'(X, t)$ holds for all $p' \in \mathbb{P}^\alpha_{<p}$
$\mathbb{P}^\alpha_{<p}(Y, t)$	either $\mathbb{P}^\alpha_{<p} = \emptyset$ or $p'(Y, t)$ holds for all $p' \in \mathbb{P}^\alpha_{<p}$
$\mathbb{P}^\beta_{<p}(X, t)$	either $\mathbb{P}^\beta_{<p} = \emptyset$ or $p'(X, t)$ holds for all $p' \in \mathbb{P}^\beta_{<p}$
$\mathbb{P}^\beta_{<p}(Y, t)$	either $\mathbb{P}^\beta_{<p} = \emptyset$ or $p'(Y, t)$ holds for all $p' \in \mathbb{P}^\beta_{<p}$

A *path* p in the topology G is a sequence of nodes $(v_1, v_2, \ldots, v_k)$ such that $k \geq 2$ and for all $i \in [1, k-1]$, v_{i+1} is a neighbor of v_i in G. For any node v, $v \in p$ simply means that v appears in p. A path p is a *simple path* if no node appears more than once in the path. All paths we discuss will be simple paths. The *length* of a path p, denoted as $|p|$, is defined as the number of nodes in p minus 1. Consider any node τ in G, where $\tau \neq \alpha$ and $\tau \neq \beta$. With respect to τ, an α-path is a path from τ to α without passing β. Formally, it is a path $(v_1, v_2, \ldots, v_k)$ satisfying $v_1 = \tau$, $v_k = \alpha$, and $v_i \neq \beta$ for all $i \in [2, k-1]$. We similarly define β-paths with respect to τ, as paths from τ to β without passing α. We will only discuss α-paths and β-paths with respect to τ, and thus we will drop the phrase "with respect to τ". As we will easily prove later, since α is itself the root, any path p from τ to the root must contain an α-path or a β-path as a part.

For any α-path or β-path p, we say that p is *cut* in a certain round if some node (potentially τ) in p fails in or before that round. Given the problematic input set $\mathcal{I}$, we say that an α-path or β-path p is *dummy* if for all $(X, Y) \in \mathcal{I}$, p is cut by the end of the execution $\Phi(X, Y)$. Otherwise, p is *non-dummy*. A non-dummy path p may still be cut in the execution of $\Phi(X, Y)$ for some $(X, Y) \in \mathcal{I}$. Intuitively, a dummy path p can be easily dismissed in our proofs later since we will be focusing on the input pairs in $\mathcal{I}$ and a dummy path is always cut in the corresponding executions. So usually we will only need to focus on non-dummy paths. We use $\mathbb{P}^\alpha$ ($\mathbb{P}^\beta$) to denote the finite set of all non-dummy α-paths (β-paths). Note that the paths in $\mathbb{P}^\alpha$ and $\mathbb{P}^\beta$ are not necessarily edge-disjoint or vertex-disjoint. For any given non-dummy α-path or β-path p, we use

$\mathbb{P}^{\alpha}_{<p}$ to denote the set of all paths in $\mathbb{P}^{\alpha}$ whose lengths are smaller than the length of p. Similarly define $\mathbb{P}^{\beta}_{<p}$.

For any input X of Alice's, if node v has a stable failure in the simulation, we use the function $F_A(X, v)$ to denote v's failure time. Otherwise, $F_A(X, v)$ is undefined. Similarly define the function $F_B(Y, v)$. For any path p and integer t, we use $p_A(X, t)$ to denote the existence of some node $v \in p$ satisfying $F_A(X, v) \leq t|p|$. Intuitively, this means that the path p will be cut in round $t|p|$ or earlier if Alice's input is X *and* if the execution continues up to round $t|p|$. We similarly define $p_B(Y, t)$. We will often drop the subscripts in $p_A(X, t)$ and $p_B(Y, t)$ since they are usually obvious. We say that $\mathbb{P}^{\alpha}_{<p}(X, t)$ holds if either $\mathbb{P}^{\alpha}_{<p}$ is empty or if $p'(X, t)$ holds for all $p' \in \mathbb{P}^{\alpha}_{<p}$. Similarly define $\mathbb{P}^{\alpha}_{<p}(Y, t)$, $\mathbb{P}^{\beta}_{<p}(X, t)$, and $\mathbb{P}^{\beta}_{<p}(Y, t)$. Also similarly define $\mathbb{P}^{\alpha}(X, t)$, $\mathbb{P}^{\alpha}(Y, t)$, $\mathbb{P}^{\beta}(X, t)$, and $\mathbb{P}^{\beta}(Y, t)$. For any given input pair $(X, Y) \in \mathcal{I}$, we say that a non-dummy α-path or β-path p is a *focal* path iff both of the following two properties hold:

—$\mathbb{P}^{\alpha}_{<p}(X, b)$, or $\mathbb{P}^{\alpha}_{<p}(Y, b)$, or both hold;
—$\mathbb{P}^{\beta}_{<p}(X, b)$, or $\mathbb{P}^{\beta}_{<p}(Y, b)$, or both hold.

Recall that b is the time complexity of the SUM protocol, in terms of aggregation rounds. As we will prove later, a focal path p has the nice property that all α-paths and β-paths shorter than p will by cut by the end of the execution $\Phi(X, Y)$. This is often a necessary precondition for us to reason about various properties on p.

Finally, recall the definition of an aggregation round from Section 3. We define $\lambda(X, Y)$ to be the number of rounds in an aggregation round in the execution of $\Phi(X, Y)$. In other words, $\lambda(X, Y) = \max_{G' \in \mathcal{G}} \Lambda(G')$ where $\mathcal{G}$ is the set of topologies that have ever appeared in the execution of $\Phi(X, Y)$. Since an oblivious reduction needs to work for any arbitrary and black-box SUM oracle protocol whose time complexity is up to b aggregation rounds for some given b, the oblivious reduction needs to work even under the worst-case scenario where the execution of $\Phi(X, Y)$ takes as long as $b\lambda(X, Y)$ rounds.

Key Challenge in the Proof. Recall that Lemma 7.4 intends to claim that τ will be disconnected from the root in the execution of $\Phi(X, Y)$ for any $(X, Y) \in \mathcal{I}$. To prove that τ will be disconnected from the root, we need to show that there does not exist any path in G (which can be arbitrary) for τ to reach the root when the execution ends. The key challenge here is that a failure in the reference setting for (X, Y) may or may not actually occur in the execution of $\Phi(X, Y)$—if the execution terminates before the failure time of a node v, then node v does not actually fail in $\Phi(X, Y)$. This is further complicated by the fact that the total number of rounds in $\Phi(X, Y)$ (i.e., $b\lambda(X, Y)$) depends on the value of $\lambda(X, Y)$, which is itself affected by failures.

This challenge implies that conceptually in our proof, we will need to start with an initial pessimistic guess (i.e., a loose lower bound) on $\lambda(X, Y)$ and on the total number of rounds in $\Phi(X, Y)$. Based on this pessimistic guess, we can show that certain failures must occur by the end of $\Phi(X, Y)$. Those failures in turn allow us to obtain a better guess on $\lambda(X, Y)$ (i.e., raising our lower bound on $\lambda(X, Y)$). This better guess then enables us to prove that some more failures will occur. Repeating this process, implicitly via an induction, will address the above challenge.

Some Technical Lemmas. In this section, we will prove a series of technical lemmas (Lemmas D.3 through D.9), which will be needed for the proof of Lemma 7.4 later.

LEMMA D.3. *Any path p from τ ($\tau \neq \alpha$ and $\tau \neq \beta$) to the root must contain an α-path or a β-path as a part.*

PROOF. Trivially follows from the fact that α is the root. In fact, it is possible to prove the following stronger claim: p must either be an α-path itself or contains a β-path as a part. We chose to still state the lemma in its current form since we want the lemma to be symmetric for α and β. □

The following lemma says that if a path p is a focal path for an input pair $(X, Y) \in \mathcal{I}$, then $\lambda(X, Y)$ will be no smaller than the length of p. Intuitively, this holds because $\lambda(X, Y)$ is no smaller than the length of the shortest path from τ to α at the end of the execution $\Phi(X, Y)$. This shortest path must contain an α-path or a β-path as a part. But since p is a focal path, we will show that all α-paths and β-paths that are shorter than p must have been cut by the end of the execution. Hence, this shortest path is no shorter than p.

LEMMA D.4. *Consider any focal path p for any input pair $(X, Y) \in \mathcal{I}$. We have $|p| \le \lambda(X, Y)$.*

PROOF. By the construction of $\mathcal{I}$, we know that for any input pair $(X, Y) \in \mathcal{I}$, τ has a value of 1 in the execution of $\Phi(X, Y)$. Lemma 7.2 tells us that τ will not be disconnected from the root in $\Phi(X, Y)$. Let p_1 denote the shortest path from τ to the root at the end of the execution $\Phi(X, Y)$. By definition of an aggregation round, we know that the number of round in an aggregation round is no smaller than the root's eccentricity in the graph, and thus we have $|p_1| \le \lambda(X, Y)$. Next, consider the set of all α-paths and β-paths. We claim that any α-path or β-path that is shorter than p will no longer exist (i.e., been cut) by the end of the execution $\Phi(X, Y)$. If this claim does hold, then notice that, by Lemma D.3, p_1 must contain an α-path or a β-path. This means that $|p| \le |p_1| \le \lambda(X, Y)$.

We prove the earlier claim via a contradiction, and assume that there exist some α-paths and/or β-paths that are shorter than p and they still exist at the end of the execution $\Phi(X, Y)$. Let p_2 be the shortest one of those paths (if there are multiple such p_2's, simply pick an arbitrary one). Note that p_2 must be a non-dummy path. Again, since Lemma D.3 tells us that p_1 must contain an α-path or a β-path, we must have $|p_2| \le |p_1| \le \lambda(X, Y)$. If $p_2 \in \mathbb{P}^\alpha$, then $p_2 \in \mathbb{P}^\alpha_{<p}$ since $|p_2| < |p|$. Since p is a focal path, we know that either $\mathbb{P}^\alpha_{<p}(X, b)$ or $\mathbb{P}^\alpha_{<p}(Y, b)$ hold, implying that either $p_2(X, b)$ or $p_2(Y, b)$ hold. Since $b|p_2| \le b\lambda(X, Y)$, there will be a failure on p_2 by the end of the execution of $\Phi(X, Y)$. Contradiction. The case for $p_2 \in \mathbb{P}^\beta$ is similar. □

The following lemma says that, for any input pair $(X, Y) \in \mathcal{I}$, $\mathbb{P}^\alpha(X, b)$ and $\mathbb{P}^\beta(Y, b)$ cannot both hold. Intuitively, this is because if they both held, then τ would be disconnected from the root (i.e., α) by the end of the execution $\Phi(X, Y)$.

LEMMA D.5. *For any input pair $(X, Y) \in \mathcal{I}$, it is impossible for $\mathbb{P}^\alpha(X, b)$ and $\mathbb{P}^\beta(Y, b)$ to both hold.*

PROOF. By Lemma 7.2, we know that τ will not be disconnected from the root in the execution of $\Phi(X, Y)$. This means there exists some path p_1 from τ to the root at the end of the execution. Lemma D.3 tells us that p_1 must contain an α-path or a β-path. This means that at the end of the execution, there is at least one α-path or β-path that has not been cut. Let p be the shortest α-path or β-path that has not been cut at the end of the execution (if there are multiple such p's, simply pick an arbitrary one). Clearly, p must be a non-dummy path.

First consider the case where p is a non-dummy α-path. By how we pick p, we know that p is a focal path for (X, Y). Lemma D.4 tells us that $|p| \le \lambda(X, Y)$. Now since p has not been cut at the end of the execution in round $b\lambda(X, Y)$, we know that $p(X, b)$ must not hold. This means that $\mathbb{P}^\alpha(X, b)$ does not hold.[16] If p is a non-dummy β-path, then one can similarly show that $\mathbb{P}^\beta(Y, b)$ does not hold. □

The next lemma considers any given focal path p with respect to an input pair $(X, Y) \in \mathcal{I}$. The lemma intuitively says that if some node in p is already spoiled for Alice's input X in a certain round, then this spoiled node will cause all other nodes in p to become spoiled within the next $|p|$ rounds, unless a stable failure is simulated on the path to "block" such spreading of spoiled nodes.

LEMMA D.6. *Consider any focal path p for any input pair $(X, Y) \in \mathcal{I}$. In the execution of $\Phi(X, Y)$, for all $t \le b - 1$:*

—*If some node in p is spoiled for Alice's input X in round $t|p|$ and if $p(X, t+1)$ does not hold, then all nodes in p are spoiled for Alice's input X in round $(t+1)|p|$.*
—*If some node in p is spoiled for Bob's input Y in round $t|p|$ and if $p(Y, t+1)$ does not hold, then all nodes in p are spoiled for Bob's input Y in round $(t+1)|p|$.*

PROOF. First, Lemma D.4 tells us that $|p| \le \lambda(X, Y)$, and thus $(t+1)|p| \le b\lambda(X, Y)$. The remainder of the proof follows directly from the definition of spoil paths. In particular, by definition of spoil paths, only stable failures can block spoil paths. □

The next lemma still considers any given focal path p with respect to an input pair $(X, Y) \in \mathcal{I}$. The lemma intuitively says that if some node on p has a stable failure with respect to Y and is not simultaneously a stable failure with respect to X, then that node becomes an epicenter for Alice. Unless we simulate a stable failure (with respect to X) on p to "block" the spreading of spoiled nodes caused by this epicenter, all nodes on p will be spoiled for Alice's input X within the next $|p|$ rounds.

LEMMA D.7. *Consider any focal path p for any input pair $(X, Y) \in \mathcal{I}$. In the execution of $\Phi(X, Y)$, for all $t \le b - 1$:*

—*If $p(Y, t)$ holds and if $p(X, t+1)$ does not hold, then all nodes in p are spoiled for Alice's input X in round $(t+1)|p|$.*
—*If $p(X, t)$ holds and if $p(Y, t+1)$ does not hold, then all nodes in p are spoiled for Bob's input Y in round $(t+1)|p|$.*

PROOF. First, Lemma D.4 tells us that $|p| \le \lambda(X, Y)$, and thus $(t+1)|p| \le b\lambda(X, Y)$. Without loss of generality, we only prove the first part of the lemma. By definition of $p(Y, t)$, we know that there exists some node $v \in p$ such that $F_B(Y, v) \le t|p| < (t+1)|p| \le b\lambda(X, Y)$. Since $p(X, t+1)$ does not hold, the failure of v in round $F_B(Y, v)$ must not be a stable failure for Alice's input X. But since v does fail in the reference setting for (X, Y) in round $F_B(Y, v)$, it means that v is an epicenter for Alice's input X. (Note that v may still be either a value epicenter or a failure epicenter.) The occurrence time of this epicenter is round $F_B(Y, v)$ or earlier. This means that v must be spoiled in round $F_B(Y, v) \le t|p|$. Applying Lemma D.6 then finishes the proof. □

The next lemma still considers any given focal path p with respect to an input pair $(X, Y) \in \mathcal{I}$. The lemma intuitively says that, if τ is spoiled for Alice's input X, in order to prevent α from being spoiled within th next $|p|$ rounds, we must simulate a stable failure with respect to X somewhere on p.

LEMMA D.8. *Consider any focal path p for any input pair $(X, Y) \in \mathcal{I}$. For all $t \le b-1$:*

—*If $p \in \mathbb{P}^\alpha$ and if τ is spoiled for Alice's input X in round $t|p|$, then $p(X, t+1)$ must hold.*
—*If $p \in \mathbb{P}^\beta$ and if τ is spoiled for Bob's input Y in round $t|p|$, then $p(Y, t+1)$ must hold.*

[16]One can also simultaneously show that $\mathbb{P}^\alpha(Y, b)$ does not hold, though we do not need that claim.

PROOF. First, Lemma D.4 tells us that $|p| \leq \lambda(X, Y)$, and thus $(t+1)|p| \leq b\lambda(X, Y)$. Without loss of generality, we only prove the first part, via a contradiction. By Lemma D.6, if $p(X, t+1)$ does not hold, then in the execution of $\Phi(X, Y)$, all nodes in p are spoiled for Alice's input X in round $(t+1)|p|$. Since $(t+1)|p| \leq b\lambda(X, Y)$, this means that α (which is in p) is spoiled by the end of the execution, which contradicts with Corollary D.2. □

The following will use these lemmas to prove the final technical lemma in this appendix. Even though the proof only uses elementary induction, it is rather complex because while we are doing an induction on the input pairs in the problematic input set $\mathcal{I}$, we need to simultaneously reason about the multiple paths in G and these two issues are entangled together. Later, we will only need to use the second claim in the following lemma—the first claim in the lemma is proved so that we can carry both claims in the induction, which is critical for the proof to work.

LEMMA D.9. *Suppose $b \geq 27$. Let $X^{(0)}$, $Y^{(1)}$, $X^{(1)}$, $Y^{(2)}, \ldots, X^{(k)}$, $Y^{(k+1)}$ (where $k+1 \leq \lfloor\sqrt{b/3}\rfloor$) be those inputs in the proof of Lemma 7.3 that correspond to the problematic input set $\mathcal{I}$. For any path $p \in \mathbb{P}^{\alpha}$, any integer $i \in [1, k+1]$, and any integer $t_i \in [0, b-2i^2-2i]$, we have the following.*

—*If $\mathbb{P}^{\alpha}_{<p}(X^{(i-1)}, t_i+4i)$ and $\mathbb{P}^{\beta}_{<p}(Y^{(i)}, t_i)$ holds, then $p(X^{(i-1)}, t_i+4i)$ holds.*
—*In particular, if $\mathbb{P}^{\alpha}_{<p} = \mathbb{P}^{\beta}_{<p} = \emptyset$, then $p(X^{(i-1)}, t_i+4i)$ holds.*

PROOF. First, note that $i \leq k+1 \leq \lfloor\sqrt{b/3}\rfloor$ and $b \geq 27$ imply $b - 2i^2 - 2i \geq 0$. This means that the range for t_i is never empty. We only prove the first part of the lemma, since the second part is the special case of the first part. We prove the first part via an induction on i. For $i = 1$, since $t_1 < t_1 + 4 \leq b$, we have $\mathbb{P}^{\alpha}_{<p}(X^{(0)}, b)$ and $\mathbb{P}^{\beta}_{<p}(Y^{(1)}, b)$. This means that p is a focal path for the input pair $(X^{(0)}, Y^{(1)})$. In the execution of $\Phi(X^{(0)}, Y^{(1)})$, τ has a value of 1 and is spoiled for Alice's input $X^{(0)}$ in round $1 \leq |p|$. Apply Lemma D.8 and we have $p(X^{(0)}, 2)$, which implies $p(X^{(0)}, t_1+4)$ for all $t_1 \in [0, b-4]$.

Now consider any $i \geq 2$, while assuming that the lemma holds for $i-1$. We are given the condition $\mathbb{P}^{\alpha}_{<p}(X^{(i-1)}, t_i+4i)$ and $\mathbb{P}^{\beta}_{<p}(Y^{(i)}, t_i)$. We will prove $p(X^{(i-1)}, t_i+4i)$ via a contradiction and assume that it does not hold. The final contradiction will be obtained by sequentially proving the following claims.

—CLAIM 1. $\mathbb{P}^{\beta}_{<p}(X^{(i-1)}, t_i+1)$ *holds, which will be proved via the execution of* $\Phi(X^{(i-1)}, Y^{(i)})$.
—CLAIM 2. $\mathbb{P}^{\beta}_{<p}(Y^{(i-1)}, t_i+2)$ *holds, which will be proved via the execution of* $\Phi(X^{(i-1)}, Y^{(i-1)})$.
—CLAIM 3. $\mathbb{P}^{\alpha}_{<p}(X^{(i-2)}, t_i+4i-2)$ *holds, which will be proved via the inductive hypothesis and implicitly via the execution of* $\Phi(X^{(i-2)}, Y^{(i-1)})$.
—CLAIM 4. $p(X^{(i-2)}, t_i+4i-2)$ *holds, which will be proved via the inductive hypothesis and implicitly via the execution of* $\Phi(X^{(i-2)}, Y^{(i-1)})$.
—CLAIM 5. $p(Y^{(i-1)}, t_i+4i-1)$ *holds, which will be proved via the execution* $\Phi(X^{(i-2)}, Y^{(i-1)})$.
—CLAIM 6. $p(X^{(i-1)}, t_i+4i)$ *holds, which will be proved via the execution of* $\Phi(X^{(i-1)}, Y^{(i-1)})$.

Figure 13 illustrates these six claims in an example topology.

PROVING CLAIM 1. We prove $\mathbb{P}^{\beta}_{<p}(X^{(i-1)}, t_i+1)$ via a contradiction and let p_1 be any path in $\mathbb{P}^{\beta}_{<p}$ where $p_1(X^{(i-1)}, t_i+1)$ does not hold. We first show that p and p_1 are both focal

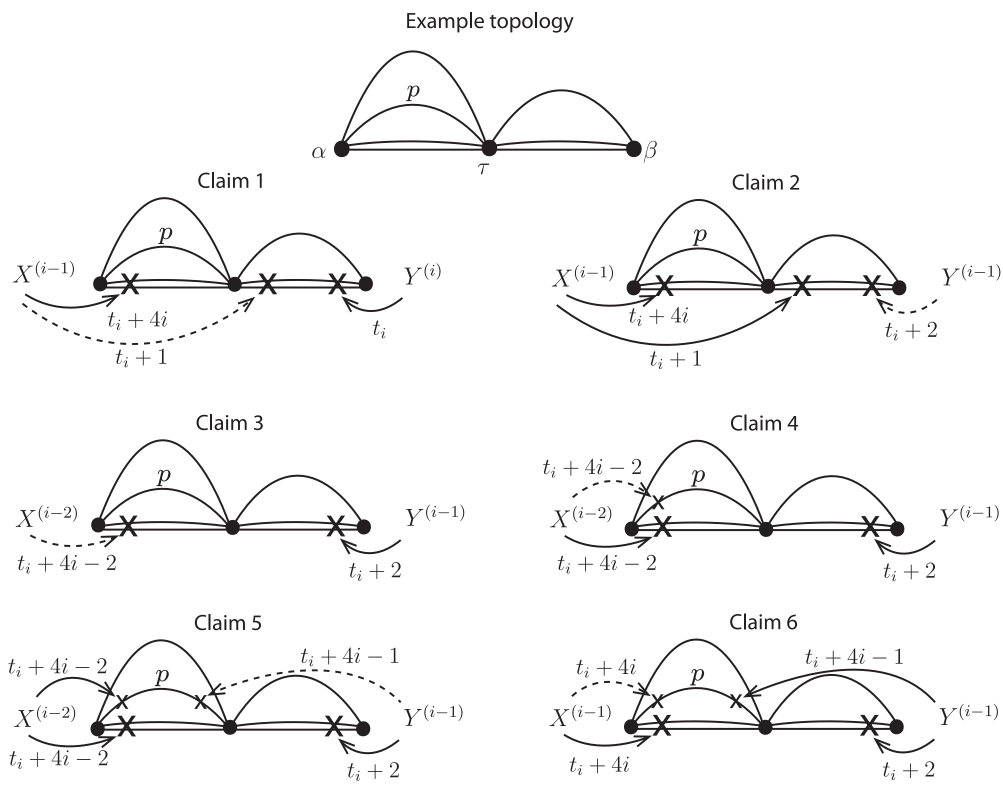

Fig. 13. Illustration of the six claims proved in Lemma D.9 in an example topology. For the given τ, this example topology has four α-paths and three β-paths. Nodes other than α, β, and τ are not shown in the figure. The α-path marked by p is the path p in the lemma. For clarity, we omit the labels for α, β, and τ when illustrating the claims. For each of the claims, the figure indicates on the left the input (e.g., $X^{(i-1)}$) to Alice, and on the right the input to Bob. Solid arrows indicate those (stable) failures that we already know, given the corresponding input to Alice or Bob. Dashed arrows indicate those (stable) failures whose existence is proved in the corresponding claim.

paths for the input pair $(X^{(i-1)}, Y^{(i)})$. For p, we have $\mathbb{P}^{\alpha}_{<p}(X^{(i-1)}, t_i + 4i)$ and $\mathbb{P}^{\beta}_{<p}(Y^{(i)}, t_i)$. Since $t_i < t_i + 4i < b$, we know that p is a focal path for $(X^{(i-1)}, Y^{(i)})$. For p_1, since $\mathbb{P}^{\alpha}_{<p_1} \subset \mathbb{P}^{\alpha}_{<p}$ and $\mathbb{P}^{\beta}_{<p_1} \subset \mathbb{P}^{\beta}_{<p}$, by similar argument, we know that p_1 is a focal path for $(X^{(i-1)}, Y^{(i)})$ as well.

Next, by the original condition, we have $p_1(Y^{(i)}, t_i)$. Invoke Lemma D.7 for p_1 and we know that all nodes on p_1 (including τ) are spoiled for Alice's input $X^{(i-1)}$ in the execution of $\Phi(X^{(i-1)}, Y^{(i)})$ in round $(t_i + 1)|p_1| < (t_i + 1)|p|$. We next invoke Lemma D.8 for p and we know that $p(X^{(i-1)}, t_i + 2)$ must hold, which implies $p(X^{(i-1)}, t_i + 4i)$. Contradiction.

Proving Claim 2. Consider any path $p_1 \in \mathbb{P}^{\beta}_{<p}$. We first show that p_1 is a focal path for the input pair $(X^{(i-1)}, Y^{(i-1)})$. From the original condition of $\mathbb{P}^{\alpha}_{<p}(X^{(i-1)}, t_i + 4i)$ and since $\mathbb{P}^{\alpha}_{<p_1} \subset \mathbb{P}^{\alpha}_{<p}$, we have $\mathbb{P}^{\alpha}_{<p_1}(X^{(i-1)}, t_i + 4i)$ which implies $\mathbb{P}^{\alpha}_{<p_1}(X^{(i-1)}, b)$. Next, Claim 1 tells us that $\mathbb{P}^{\beta}_{<p}(X^{(i-1)}, t_i + 1)$. By a similar argument, we have $\mathbb{P}^{\beta}_{<p_1}(X^{(i-1)}, b)$. Thus, p_1 is a focal path for $(X^{(i-1)}, Y^{(i-1)})$.

From Claim 1, we also have $p_1(X^{(i-1)}, t_i + 1)$. Now, invoke Lemma D.7 for p_1. We will then have $p_1(Y^{(i-1)}, t_i + 2)$, since otherwise all nodes in p_1 (including β) are spoiled for Bob's input $Y^{(i-1)}$ in the execution of $\Phi(X^{(i-1)}, Y^{(i-1)})$ in round $(t_i + 2)|p_1|$.

PROVING CLAIM 3. Prove by contradiction and assume that $\mathbb{P}^{\alpha}_{<p}(X^{(i-2)}, (t_i+2)+4(i-1))$ does not hold. Let p_1 be the shortest path (if there are multiple such p_1's, simply pick an arbitrary one) in $\mathbb{P}^{\alpha}_{<p}$ such that the $p_1(X^{(i-2)}, (t_i+2)+4(i-1))$ does not hold. We next want to invoke the inductive hypothesis for $i-1$ on $p_1 \in \mathbb{P}^{\alpha}$ with $t_{i-1} = t_i + 2$. Such invocation is possible since:

—$t_{i-1} = t_i + 2 \le b - 2i^2 - 2i + 2 < b - 2i^2 - 2i + 4i = b - 2(i-1)^2 - 2(i-1)$;
—By definition of p_1, we have that $\mathbb{P}^{\alpha}_{<p_1}(X^{(i-2)}, (t_i+2)+4(i-1))$ holds;
—We know from Claim 2 that $\mathbb{P}^{\beta}_{<p}(Y^{(i-1)}, t_i+2)$ holds, implying that $\mathbb{P}^{\beta}_{<p_1}(Y^{(i-1)}, t_i+2)$ holds.

This invocation tells us that $p_1(X^{(i-2)}, (t_i+2)+4(i-1))$ holds, leading to a contradiction.

PROVING CLAIM 4. We want to invoke the inductive hypothesis for $i-1$ on p with $t_{i-1} = t_i + 2$. Such invocation is possible since:

—$t_{i-1} = t_i + 2 \le b - 2i^2 - 2i + 2 < b - 2i^2 - 2i + 4i = b - 2(i-1)^2 - 2(i-1)$;
—Claim 3 gives us $\mathbb{P}^{\alpha}_{<p}(X^{(i-2)}, (t_i+2)+4(i-1))$;
—Claim 2 gives us $\mathbb{P}^{\beta}_{<p}(Y^{(i-1)}, t_i+2)$.

This invocation tells us that $p(X^{(i-2)}, (t_i+2)+4(i-1))$ holds.

PROVING CLAIM 5. We first show that p is a focal path for $(X^{(i-2)}, Y^{(i-1)})$. We already have $\mathbb{P}^{\alpha}_{<p}(X^{(i-2)}, t_i+4i-2)$ from Claim 3 and $\mathbb{P}^{\beta}_{<p}(Y^{(i-1)}, t_i+2)$ from Claim 2. Since $t_i + 2 \le t_i + 4i - 2 \le b - 2i^2 - 2i + 4i - 2 < b$, we now know that p is a focal path for $(X^{(i-2)}, Y^{(i-1)})$. We next prove $p(Y^{(i-1)}, t_i+4i-1)$ via a contradiction.

We already have $p(X^{(i-2)}, t_i+4i-2)$ from Claim 4. Since $p(Y^{(i-1)}, t_i+4i-1)$ does not hold, we can invoke Lemma D.7 for p. That lemma tells us that in the execution of $\Phi(X^{(i-2)}, Y^{(i-1)})$, all nodes in p (including τ) become spoiled for Bob's input $Y^{(i-1)}$ in round $(t_i+4i-1)|p|$. This is a critical property which we will use later.

Next, consider the two properties $\mathbb{P}^{\alpha}(X^{(i-2)}, t_i+8i-4)$ and $\mathbb{P}^{\beta}(Y^{(i-1)}, t_i+4i)$. By Lemma D.5, it is impossible for both of them to hold, since otherwise they would imply that both $\mathbb{P}^{\alpha}(X^{(i-2)}, b)$ and $\mathbb{P}^{\beta}(Y^{(i-1)}, b)$ hold. Let p_1 be the shortest path (if there are multiple such p_1's, simply pick an arbitrary one) in $\mathbb{P}^{\alpha} \cup \mathbb{P}^{\beta}$ where $p_1(X^{(i-2)}, t_i+8i-4)$ (if $p_1 \in \mathbb{P}^{\alpha}$) or $p_1(Y^{(i-1)}, t_i+4i)$ (if $p_1 \in \mathbb{P}^{\beta}$) does not hold.

We consider two cases. If $p_1 \in \mathbb{P}^{\beta}$, we will first show that p_1 is a focal path. By definition of p_1, we have $\mathbb{P}^{\alpha}_{<p_1}(X^{(i-2)}, t_i+8i-4)$ and $\mathbb{P}^{\beta}_{<p_1}(Y^{(i-1)}, t_i+4i)$ holds. Since $t_i + 4i \le b - 2i^2 - 2i + 4i < b$, $\mathbb{P}^{\alpha}_{<p_1}(X^{(i-2)}, b)$ and $\mathbb{P}^{\beta}_{<p_1}(Y^{(i-1)}, b)$ holds. This means that p_1 is a focal path for the input pair $(X^{(i-2)}, Y^{(i-1)})$. We next want to show that $|p| \le |p_1|$. By how we chose p_1, we know that $p_1(Y^{(i-1)}, t_i+4i)$ does not hold. On the other hand, Claim 2 tells us that $\mathbb{P}^{\beta}_{<p}(Y^{(i-1)}, t_i+2)$ holds, implying that $\mathbb{P}^{\beta}_{<p}(Y^{(i-1)}, t_i+4i)$ holds (since $i \ge 2$). Thus, we must have $p_1 \notin \mathbb{P}^{\beta}_{<p}$ and $|p_1| \ge |p|$. Finally, as shown earlier, in the execution of $\Phi(X^{(i-2)}, Y^{(i-1)})$, the node τ must be spoiled for Bob's input $Y^{(i-1)}$ in round $(t_i+4i-1)|p| \le (t_i+4i-1)|p_1|$. Now, we can invoke Lemma D.8, which shows that $p_1(Y^{(i-1)}, t_i+4i)$ holds and thus leads to a contradiction.

For the second case where $p_1 \in \mathbb{P}^{\alpha}$, we want to invoke the inductive hypothesis for $i-1$ on p_1 with $t_{i-1} = t_i + 4i$. Such invocation is possible since

—$t_{i-1} = t_i + 4i \le b - 2i^2 - 2i + 4i = b - 2(i-1)^2 - 2(i-1)$;
—by definition of p_1, we have that $\mathbb{P}^{\alpha}_{<p_1}(X^{(i-2)}, (t_i+4i)+4(i-1))$ and $\mathbb{P}^{\beta}_{<p_1}(Y^{(i-1)}, t_i+4i)$ holds.

The invocation gives us $p_1(X^{(i-2)}, (t_i + 4i) + 4(i-1))$, leading to a contradiction.

PROVING CLAIM 6. We first show that p is a focal path for $(X^{(i-1)}, Y^{(i-1)})$. From the original condition, we have $\mathbb{P}^{\alpha}_{<p}(X^{(i-1)}, t_i + 4i)$, which implies $\mathbb{P}^{\alpha}_{<p}(X^{(i-1)}, b)$. By Claim 2, we have $\mathbb{P}^{\beta}_{<p}(Y^{(i-1)}, t_i + 2)$, which implies $\mathbb{P}^{\beta}_{<p}(Y^{(i-1)}, b)$. Thus p is a focal path for $(X^{(i-1)}, Y^{(i-1)})$.

Claim 5 gives us $p(Y^{(i-1)}, t_i + 4i - 1)$. Now, invoke Lemma D.7 for p. That lemma tells us that $p(X^{(i-1)}, t_i + 4i)$ must hold, since otherwise all nodes in p (including α) will be spoiled for Alice's input $X^{(i-1)}$ in round $(t_i + 4i)|p|$. □

D.3. Proof of Lemma 7.4

Using the technical lemmas proved in the previous two appendices, we can now finally prove Lemma 7.4:

PROOF OF LEMMA 7.4. By Lemma D.3, a path from τ to the root must contain either an α-path or a β-path. Thus to prove the lemma, it suffices to prove that all α-paths and β-paths are dummy. Prove by contradiction and assume that some α-paths and/or β-paths are non-dummy. Let p be the shortest path among all such paths (if there are multiple such p's, simply pick an arbitrary one). This means that there exists some $(X, Y) \in \mathcal{I}$ such that p is not cut by the end of the execution $\Phi(X, Y)$. Also by how we chose p, we trivially have $\mathbb{P}^{\alpha}_{<p} = \mathbb{P}^{\beta}_{<p} = \emptyset$.

Next first consider the case where p is a non-dummy α-path. For all i where $1 \le i \le k+1 \le \lfloor\sqrt{b/3}\rfloor$, invoke the second claim in Lemma D.9 with $t_i = 0$ and we have $p(X^{(i-1)}, 4i)$. Since $4i \le 4\lfloor\sqrt{b/3}\rfloor < b$ when $b \ge 27$, this in turn implies $p(X^{(i-1)}, b)$ for $1 \le i \le k+1$. Next since $\mathbb{P}^{\alpha}_{<p} = \mathbb{P}^{\beta}_{<p} = \emptyset$, we trivially know that p is a focal path for (X, Y). Invoke Lemma D.4 and we have $|p| \le \lambda(X, Y)$. Together with $p(X, b)$, we know that p will be cut by the end of the execution of $\Phi(X, Y)$. Contradiction.

For the second case where p is a β-path, the proof is entirely symmetric. In particular, the only difference between α and β is that α is the root while β is not. However, we only used the fact that α is the root in the proof of Lemma D.3. Lemma D.3 itself is already symmetric for α and β. In other words, if we view the proof for Lemma D.3 as a black-box, then α and β are entirely symmetric throughout Sections D.1 and D.2. □

ACKNOWLEDGMENTS

We thank Cheng Yeaw Ku and Y. C. Tay for their valuable help and pointers. We also thank the anonymous PODC and JACM reviewers for their helpful comments.

REFERENCES

N. Alon, Y. Matias, and M. Szegedy. 1996. The space complexity of approximating the frequency moments. In *Proceedings of STOC*.

O. Ayaso, D. Shah, and M. Dahleh. 2010. Information theoretic bounds for distributed computation over networks of point-to-point channels. *IEEE Trans. Inf. Theory* 56, 12, 6020–6039.

T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione. 2009. Broadcast gossip algorithms for consensus. *IEEE Trans. Sig. Proc.* 57, 7, 2748–2761.

Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. 2004. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.* 68, 4, 702–732.

M. Bawa, A. Gionis, H. Garcia-Molina, and R. Motwani. 2007. The price of validity in dynamic networks. *J. Comput. Syst. Sci.* 73, 3, 245–264.

D. Beaver. 1991. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *J. Crypt.* 4, 75–122.

Z. Beerliová-Trubíniová and M. Hirt. 2006. Efficient multi-party computation with dispute control. In *Proceedings of the 3rd Conference on Theory of Cryptography*. 305–328.

Z. Beerliová-Trubíniová and M. Hirt. 2008. Perfectly-secure mpc with linear communication complexity. In *Proceedings of the 5th Conference on Theory of Cryptography*. 213–230.

M. Ben-Or, R. Canetti, and O. Goldreich. 1993. Asynchronous secure computation. In *Proceedings of STOC*. 52–61.

M. Ben-Or, S. Goldwasser, and A. Wigderson. 1988. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of STOC*. 1–10.

P. Berman and J. A. Garay. 1993. Fast consensus in networks of bounded degree. *Distrib. Comput.*

A. Blokhuis. 1993. On the sperner capacity of the cyclic triangle. *J. Algeb. Combinat.* 2, 2, 123–124.

S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. 2006. Randomized gossip algorithms. *IEEE Trans. Inf. Theory* 52, 6, 2508–2530.

M. Braverman and A. Rao. 2011. Towards coding for maximum errors in interactive communication. In *Proceedings of STOC*.

A. Chakrabarti and O. Regev. 2011. An optimal lower bound on the communication complexity of gap-hamming-distance. In *Proceedings of STOC*.

A. Chandra, M. Furst, and R. Lipton. 1983. Multi-party protocols. In *Proceedings of STOC*.

D. Chaum, C. Crépeau, and I. Damgard. 1988a. Multiparty unconditionally secure protocols. In *Proceedings of STOC*. 11–19.

D. Chaum, I. Damgård, and Van Der J. Graaf. 1988b. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *Proceedings of the Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*. 87–119.

B. Chen, H. Yu, Y. Zhao, and P. B. Gibbons. 2012. The cost of fault tolerance in multi-party communication complexity. In *Proceedings of PODC*.

J. Chen and G. Pandurangan. 2010. Optimal gossip-based aggregate computation. In *Proceedings of SPAA*.

J. Chen, G. Pandurangan, and D. Xu. 2005. Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis. In *Proceedings of IPSN*.

B. Chlebus, D. Kowalski, and M. Strojnowski. 2009. Fast scalable deterministic consensus for crash failures. In *Proceedings of PODC*.

J. Considine, F. Li, G. Kollios, and J. Byers. 2004. Approximate aggregation techniques for sensor databases. In *Proceedings of ICDE*.

A. Dhulipala, C. Fragouli, and A. Orlitsky. 2010. Silence-based communication. *IEEE Trans. Inf. Theory* 56, 1, 350–366.

I. Eyal, I. Keidar, and R. Rom. 2011. LiMoSense—Live Monitoring in Dynamic Sensor Networks. In *Proceedings of ALGOSENSORS*.

P. Flajolet and G. N. Martin. 1985. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.* 31, 2, 182–209.

M. Franklin and M. Yung. 1992. Communication complexity of secure computation (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*.

Z. Galil, S. Haber, and M. Yung. 1987. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In *Proceedings of CRYPTO*. 135–155.

L. Gargano and A. A. Rescigno. 1993. Communication complexity of fault-tolerant information diffusion. In *Proceedings of 5th IEEE Symposium on Parallel and Distributed Processing*. 564–572.

S. Gilbert and D. Kowalski. 2010. Distributed agreement with optimal communication complexity. In *Proceedings of SODA*.

A. Giridhar and P. R. Kumar. 2006. Towards a theory of in-network computation in wireless sensor networks. *IEEE Commun. Mag.* 44, 4, 98–107.

O. Goldreich, S. Micali, and A. Wigderson. 1987. How to play any mental game. In *Proceedings of STOC*. 218–229.

J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. 1997. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining Knowl. Disc.* 1, 1.

P. Gupta and P. R. Kumar. 2000. The capacity of wireless networks. *IEEE Trans. Inf. Theory* 46, 2, 388–404.

M. Hirt, U. Maurer, and B. Przydatek. 2000. Efficient secure multi-party computation. In *Adv. Crypt.* 143–161.

M. Hirt and J. B. Nielsen. 2006. Robust multiparty computation with linear communication complexity. In *Proceedings of the 26th Annual International Conference on Advances in Cryptology*. 463–482.

R. Impagliazzo and R. Williams. 2010. Communication complexity with synchronized clocks. In *Proceedings of CCC*.

M. Jelasity, A. Montresor, and O. Babaoglu. 2005. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.* 23, 3, 219–252.

P. Jesus, C. Baquero, and P. Almeida. 2009. Fault-tolerant aggregation by flow updating. In *Proceedings of DAIS*.

S. Kashyap, S. Deb, K. Naidu, R. Rastogi, and A. Srinivasan. 2006. Efficient gossip-based aggregate computation. In *Proceedings of PODS*.

D. Kempe, A. Dobra, and J. Gehrke. 2003. Gossip-based computation of aggregate information. In *Proceedings of FOCS*.

V. King, S. Lonargan, J. Saia, and A. Trehan. 2010. Load balanced scalable Byzantine agreement through quorum building, with full information. In *Proceedings of ICDCN*.

V. King and J. Saia. 2009. From almost everywhere to everywhere: Byzantine agreement with $\tilde{O}(n^{3/2})$ bits. In *Proceedings of DISC*.

V. King and J. Saia. 2010. Breaking the $O(n^2)$ Bit Barrier: Scalable Byzantine agreement with an Adaptive Adversary. In *Proceedings of PODC*.

V. King, J. Saia, V. Sanwalani, and E. Vee. 2006a. Scalable leader election. In *Proceedings of SODA*.

V. King, J. Saia, V. Sanwalani, and E. Vee. 2006b. Towards secure and scalable computation in peer-to-peer networks. In *Proceedings of FOCS*.

E. Kushilevitz and N. Nisan. 1996. *Communication Complexity*. Cambridge University Press.

S. Madden, M. Franklin, J. Hellerstein, and W. Hong. 2002. TAG: A tiny aggregation service for ad-hoc sensor networks. In *Proceedings of OSDI*.

M. McGlynn and S. Borbash. 2001. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *Proceedings of MobiHoc*. 137–145.

D. Mosk-Aoyama and D. Shah. 2006. Computing separable functions via gossip. In *Proceedings of PODC*.

S. Nath, P. Gibbons, S. Seshany, and Z. Anderso. 2008. Synopsis diffusion for robust aggregation in sensor networks. *ACM Trans. Sensor Netw.* 4, 2.

I. Newman. 1991. Private vs. common random bits in communication complexity. *Inf. Proc. Lett.* 39, 2, 67–71.

R. Pietro and P. Michiardi. 2008. Brief announcement: Gossip-based aggregate computation: Computing faster with non address-oblivious schemes. In *Proceedings of PODC*.

T. Rabin and M. Ben-Or. 1989. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of STOC*.

S. Rajagopalan and L. Schulman. 1994. A coding theorem for distributed computation. In *Proceedings of STOC*.

A. Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. 2011. Distributed Verification and Hardness of Distributed Approximation. In *Proceedings of STOC*.

L. Schulman. 1996. Coding for interactive communication. *IEEE Trans. Inf. Theory* 42, 6, 1745–1756.

D. Woodruff. 2004. Optimal space lower bounds for all frequency moments. In *Proceedings of SODA*.

A. C. Yao. 1982. Protocols for secure computations. In *Proceedings of FOCS*. 160–164.

A. C. Yao. 1986. How to generate and exchange secrets. In *Proceedings of FOCS*.

H. Yu. 2011. Secure and highly-available aggregation queries in large-scale sensor networks via set sampling. *Distrib. Comput.* 23, 5, 373–394.

Received November 2012; revised August 2013; accepted February 2014