The Cost of Fault Tolerance in Multi-Party Communication Complexity*

Binbin Chen
Advanced Digital Sciences Center
Republic of Singapore
binbin.chen@adsc.com.sg

Haifeng Yu
National University of Singapore
Republic of Singapore
haifeng@comp.nus.edu.sg

Yuda Zhao
National University of Singapore
Republic of Singapore
zhaoyuda@comp.nus.edu.sg

Phillip B. Gibbons
Intel Labs
Pittsburgh, PA, USA
phillip.b.gibbons@intel.com

ABSTRACT

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 crash 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?*" This natural question, interestingly, has not been formally posed and thoroughly studied prior to this work.

Whether fault-tolerant communication complexity is interesting to study largely depends on how big a difference failures make. This paper proves that the impact of failures is significant, at least for the SUM aggregation function in general networks: 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. Our results also imply the optimality (within polylog factors) of some recent fault-tolerant protocols for computing SUM via duplicate-insensitive techniques, thereby answering an open question as well.

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.

Categories and Subject Descriptors

F.1 [**Computation by Abstract Devices**]: Complexity Measures and Classes; F.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Theory, Algorithms

*The first three authors of this paper are alphabetically ordered.

PODC'12, July 16–18, 2012, Madeira, Portugal.

Keywords

Communication complexity, fault tolerance, aggregation functions, promise problems, wireless networks

1. INTRODUCTION

Fault tolerance in communication complexity and our exponential gap. Multi-party communication complexity [10] 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. (Throughout this paper, failures refer to node crash failures unless otherwise mentioned.) 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 the question to be meaningful, we impose two restrictions before moving forward. First, we allow the computation to ignore/omit the inputs held by those players that have failed (i.e., crashed) or been disconnected. This means that the function needs to be well-defined over any subset of the inputs. Second, we will assume that there is one special *root* player that never fails and only this player needs to learn the final result. This allows us to focus on the communication complexity of the function instead of the difficulty of, for example, achieving fault-tolerant distributed consensus. This also nicely maps to our target scenarios later in wireless sensor networks and wireless ad-hoc networks, where the root corresponds to the base station.

While the above question is natural, interestingly, it has not been formally posed and thoroughly studied — see later for our speculations on the possible reasons. Whether such fault-tolerant communication complexity is interesting to study, given the extensive research on "non-fault-tolerant" communication complexity, 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 function[1] in networks with general topologies: 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 pro-

[1]As an example where the impact of failures is *not* significant, consider the MAX function. Our technical report [11] gives a simple folklore fault-tolerant MAX protocol based on binary search, where each node sends only a logarithmic number of bits.

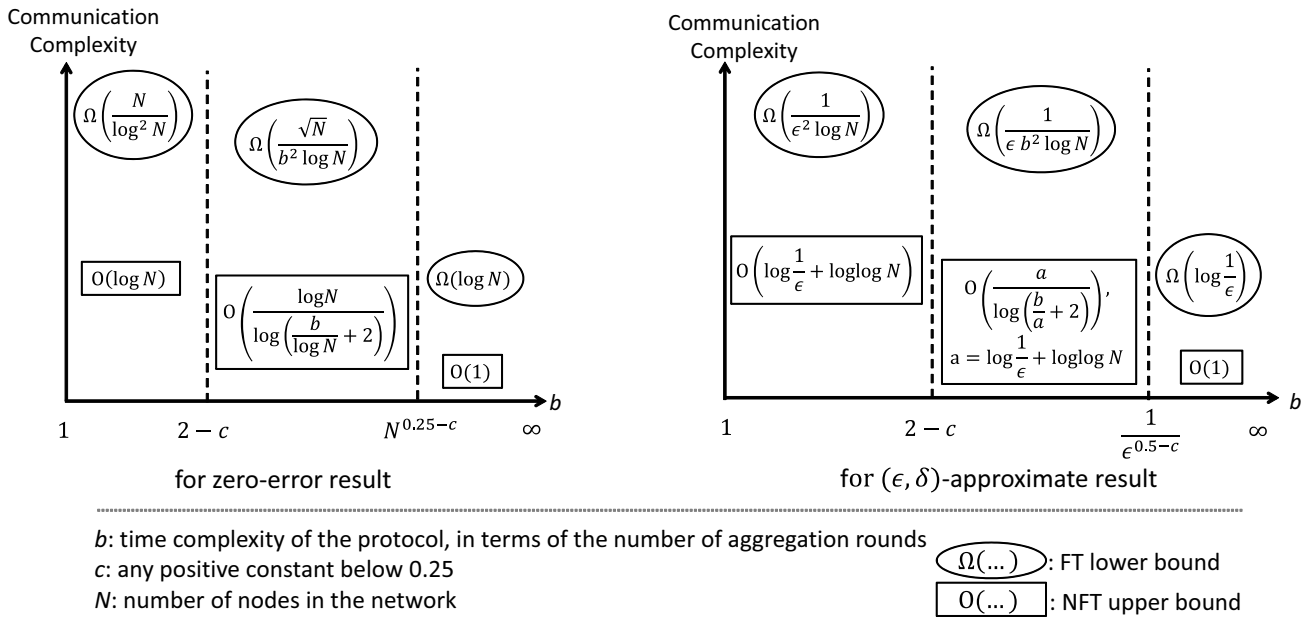

Figure 1: Summary of our exponential gaps. All NFT upper bounds are either well-known or are obtained via standard tricks — they are described in Section 3. All FT lower bounds are novel and are our main contributions. They are obtained in Section 4 (for $1 \leq b \leq 2-c$), Section 5 (for $2-c < b \leq N^{0.25-c}$ or $\frac{1}{\epsilon^{0.5-c}}$), and Section 7 (for $b > N^{0.25-c}$ or $\frac{1}{\epsilon^{0.5-c}}$).

tocols. To our knowledge, ours is the first such result on FT communication complexity. This exponential gap attests that FT communication complexity needs to be studied separately from NFT communication complexity.

The SUM function. Consider a synchronous wireless network (e.g., a wireless sensor network or a wireless ad-hoc network) with N nodes and some arbitrary topology. Each node has a binary value, and the SUM function asks for the sum of all the values. Note that SUM can be easily reduced to and from some other interesting aggregation functions such as SELECTION. Communication complexity has significant practical relevance here 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 capacity of wireless networks does not scale well [18].

Existing results on SUM. In failure-free settings, by leveraging in-network processing, a trivial tree-aggregation protocol can compute SUM with zero-error while requiring each node to send $O(\log N)$ bits. For (ϵ, δ)-approximate results, it is possible to further reduce to $O(\log \frac{1}{\epsilon} + \log\log N)$ bits per node for constant δ. In comparison, to tolerate arbitrary failures, we are not aware of any zero-error protocol for computing SUM that is better than trivially having every node flood its id together with its value and thus requiring each node to send $O(N \log N)$ bits. For (ϵ, δ)-approximate results, researchers have proposed some protocols [5, 14, 25, 26, 30] 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 exponentially 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 [1, 16], 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 [3, 7, 12, 13, 20, 22, 23] on computing SUM via gossip-based averaging (also called average consensus protocols). They all rely on the mass conservation property [23], and thus are vulnerable to node failures. There have been a few efforts [15, 21] 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* (defined in Section 2), for b from 1 to ∞. For $b \leq N^{0.25-c}$ or $\frac{1}{\epsilon^{0.5-c}}$ 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 $\frac{1}{\epsilon^{0.5-c}}$, 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, the existing fault-tolerant SUM protocols (incurring $O(N \log N)$ or $O(\frac{1}{\epsilon^2})$ bits [5, 14, 25, 26, 30] per node) are actually optimal within polylog factors.

[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 $\frac{1}{\epsilon^c} = \Omega(\log N)$.

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 $\frac{1}{\epsilon^{0.5-c}}$ 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 theoretic arguments [4]. 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 6). 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 $\frac{1}{\epsilon^{0.5-c}}$ are obtained by drawing a strong connection to an interesting probing game, and then by proving a lower bound on the probing game.

Other related work. Despite the developments (e.g., [8, 10, 19, 27, 28]) on different models for communication complexity, to the best of our knowledge, fault tolerance has never been considered. Among them, the closest setting to fault tolerance is perhaps unreliable channels [8, 27, 28] that either flip the bits adversarially or flip each bit iid. The specific techniques and insights there have limited applicability to our fault-tolerant setting. Under the iid unreliable channel model, there have also been some information-theoretic lower bounds [2, 17] on the rates of distributed computations. We suspect that such a lack of prior work on fault tolerance is due to two reasons. First, one needs to define correctness in a meaningful way when failures are possible, since some of the inputs can be missing. For this work, recent applications in wireless sensor networks have shown us how to do so [5]. Second, communication complexity problems tend to be challenging to study, and taking failures into account only makes things harder. For this work, we rely on several quite recent results [9, 19].

2. MODEL AND DEFINITIONS

This section describes the system model and formal definitions used throughout this paper, except in Section 8. For clarity, we defer to Section 8 various relaxed/extended versions of the system model and definitions, under which our exponential gap results continue to hold. All "log" in this paper 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, and one of the N nodes is the *root*. We assume that the topology G (including the ids of each node in G) is known to all nodes. The system is synchronous and a protocol proceeds in synchronous rounds. 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. Our results are insensitive to whether collisions are possible, but to make everything concrete, we still adopt and stick to the following commonly-used collision model. By doing a send, a node (locally) broadcasts one message to all its neighbors in G. 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. The root never fails. Any other node in G may experience crash failures (but not byzantine failure), and the total number of failures can be up to $N - 1$. See Section 8 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.

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 [5], 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 paper 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 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 3, 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. This implies that an aggregation round for an FT protocol is either the same or longer than that for an NFT protocol, which makes our gap results stronger.

NFT and FT communication complexity of SUM protocols. Classic multi-party communication complexity problems [24] 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 $\mathcal{R}_0^{\text{syn}}(\text{SUM}, G, b)$ ($\mathcal{R}_{\epsilon,\delta}^{\text{syn}}(\text{SUM}, G, b)$, respectively) to be the smallest average-case (worst-case, respectively) communication complexity under G across all randomized SUM protocols that can generate, in a failure-free setting, a zero-error result ((ϵ, δ)-approximate result, respectively) 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 $\mathcal{R}_{\epsilon,\delta}^{\text{syn}}$ is standard practice [4, 24]. With respect to any topology G, we similarly define $\mathcal{R}_0^{\text{syn,ft}}(\text{SUM}, G, b)$ and $\mathcal{R}_{\epsilon,\delta}^{\text{syn,ft}}(\text{SUM}, G, b)$ across all fault-tolerant randomized SUM protocols.

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

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 paper, 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. Different from the classic setting [24] for two-party problems, we will need to consider a setting with synchronous rounds[3], adapted from [19]. 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 $\mathcal{R}_0^{\text{syn}}(\Pi, t)$ ($\mathcal{R}_{\epsilon,\delta}^{\text{syn}}(\Pi, t)$, respectively) to be the smallest average-case (worst-case, respectively) communication complexity across all randomized protocols for Π that can generate a zero-error result ((ϵ, δ)-approximate result, respectively) within a time complexity of at most t rounds.

3. 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.

[3]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.

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 is deterministic, the receiver can locally determine the sender. The function $\Lambda(G)$ is formally defined to be the number of rounds needed for the above deterministic protocol to finish on G. Thus by definition, the above 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 [1] (see our technical report [11]). 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 [19] 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 the above leads to:

THEOREM 1. *For any $b \geq 1$, we have $\mathcal{R}_0^{\text{syn}}(\text{SUM}_N, b) = O(a/\log(\frac{b}{a} + 2))$ where $a = \log N$, and $\mathcal{R}_{\epsilon,\frac{1}{3}}^{\text{syn}}(\text{SUM}_N, b) = O(a/\log(\frac{b}{a} + 2))$ where $a = \log\frac{1}{\epsilon} + \log\log N$.*

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

The UNIONSIZE problem. 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 [9, 19, 29] tells us that $\mathcal{R}_0^{\text{syn}}(\text{UNIONSIZE}_n, O(\text{poly}(n))) = \Omega(\frac{n}{\log n})$ and $\mathcal{R}_{\epsilon,\frac{1}{3}}^{\text{syn}}(\text{UNIONSIZE}_n, O(\text{poly}(n))) = \Omega(\frac{1}{\epsilon^2 \log n})$ for $\epsilon \geq \frac{1}{\sqrt{n}}$ [11].

Overview of our reduction and its novelty. While the well-known reduction [29] from UNIONSIZE to the (centralized) one-pass distinct element counting problem is almost trivial, we seek a reduction from UNIONSIZE to SUM, which is less obvious. 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 5. 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

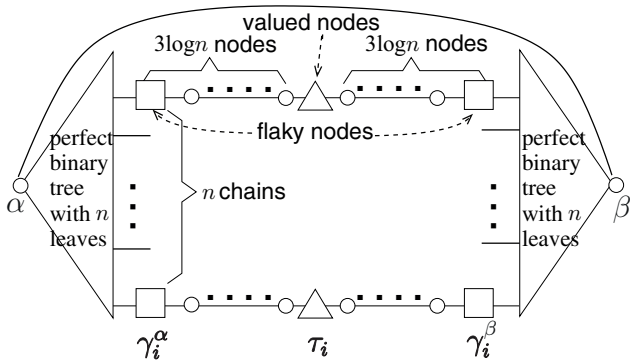

Figure 2: FT lower bound topology for $b \leq \frac{10}{9}$.

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* for Alice, in the sense that Alice cannot simulate τ. (See the formal framework established in our technical report [11] for the rigorous definition of a spoiled node.) 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 2 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.

Reducing from UNIONSIZE to SUM. For better understanding, the topology (Figure 2) we describe here works for $b \leq \frac{10}{9}$. See our technical report [11] for the topology for $b \leq 2 - c$, with the c there being any positive constant. Given UNIONSIZE$_n$ with n being a power of 2, the topology 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, 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.

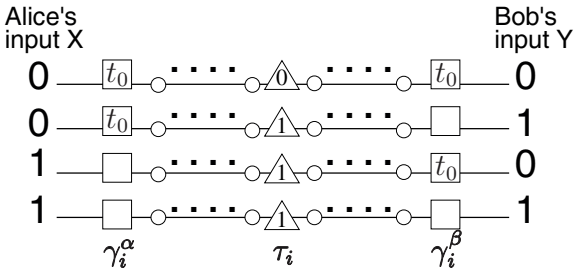

Figure 3: Values of valued nodes and failure times of flaky nodes, for $X = 0011$ and $Y = 0101$.

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. non-valued 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 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).

Alice will simulate the shrinking group of all the unspoiled nodes for Alice, which always contains node α. Bob similarly simulates the unspoiled nodes for Bob, including node β. (These two groups are made precise in our technical report [11].) 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.

To obtain some intuition, 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.

We obtain the following theorem by formalizing the above arguments, using an improved topology as in our technical report [11], and then trivially extending to those N values that currently do not map to any integer n for UNIONSIZE$_n$. The proof is in [11].

THEOREM 2. *For any $b \in [1, 2 - c]$ where c is any positive constant, we have $\mathcal{R}_0^{\text{syn,ft}}(\text{SUM}_N, b) = \Omega\left(\frac{N}{\log^2 N}\right)$ and $\mathcal{R}_{\epsilon, \frac{1}{3}}^{\text{syn,ft}}(\text{SUM}_N, b) = \Omega\left(\frac{1}{\epsilon^2 \log N}\right)$ for $\epsilon \geq \frac{\sqrt{9 \log N}}{\sqrt{cN}}$.*

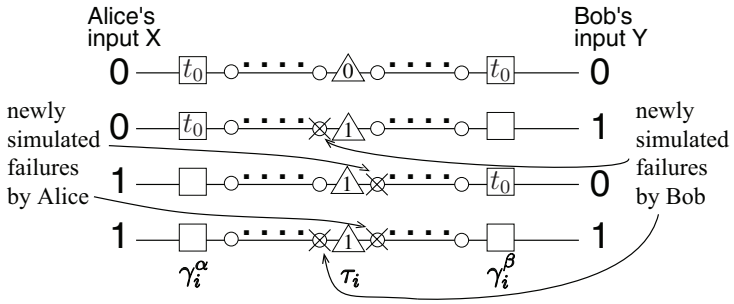

Figure 4: Why the construction from Section 4 cannot be extended to larger b.

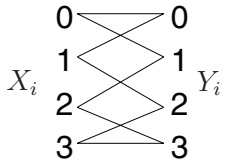

Figure 5: The cycle promise for $q = 4$.

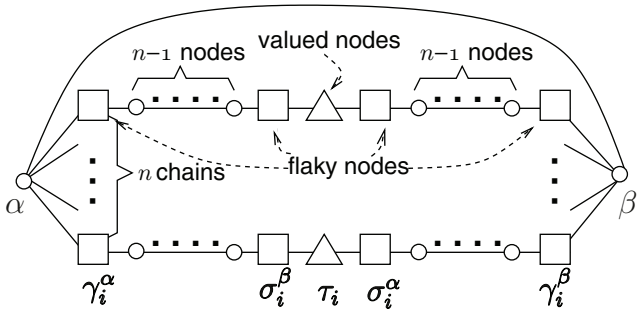

Figure 6: FT lower bound topology for $b \leq N^{0.25-c}$ or $b \leq \frac{1}{\epsilon^{0.5-c}}$.

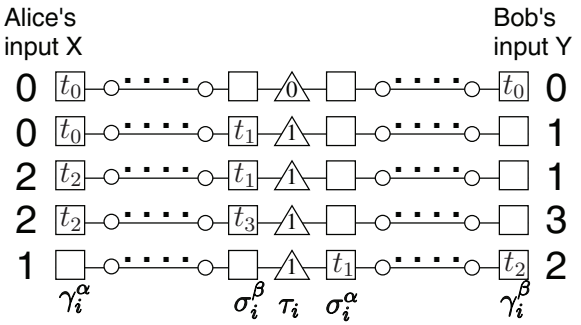

Figure 7: Values of valued nodes and failure times of flaky nodes, for $q = 4$, $X = 00221$, and $Y = 01132$.

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

Why the previous construction cannot be extended. The FT lower bounds in the previous section 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 4. Here when $Y_i = 1$, Bob simulates a new failure to the left of τ_i, to prevent the propagation of spoiled nodes due to γ_i^α. 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 4, this prevents us from using the SUM result to determine UNIONSIZE. 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, again 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 $O(\log n)$, making the final results trivial.

The UNIONSIZECP problem. To overcome the above problem, we will introduce and reduce from a new two-party communication complexity problem called 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 5 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 make the communication complexity of UNIONSIZECP trivial. 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.

Reducing from UNIONSIZECP to SUM. Figure 6 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.[4] 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 7, the 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 4 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 4 nodes, respectively. Let $t_j = (j+1)n + 1$ for all $0 \leq j \leq q-1$. The flaky node γ_i^α fails

[4]Using binary trees will not work here. Consequently, here an aggregation round will contain more rounds than in Section 4, and in turn each chain needs to have more nodes.

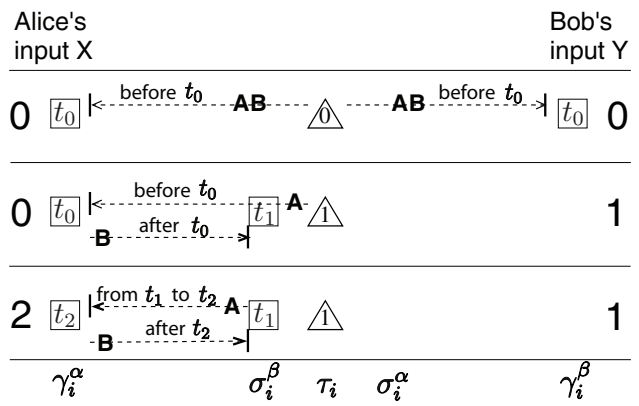

Figure 8: Failures prevent the spreading of spoiled nodes. Dashed arrows labeled A (B) indicate the spreading of spoiled nodes for Alice (Bob).

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 7). 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 oblivious to the SUM protocol, and fails only a vanishingly small fraction (i.e., $o(N)$) of all the nodes in G.

To gain some intuition, consider the example in Figure 8. We say that a node 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. Similarly define epicenters for Bob's input Y. 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 scenario in Figure 8). 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 scenario). 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 (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 next), as expected, our lower bounds here will be a function of b. We obtain the following theorem via formalizing the above reduction, using our lower bound later (Theorem 4) from UNIONSIZECP, and then trivially extending to all N values. See our technical report [11] for the proof.

THEOREM 3. *For any $b \geq 1$, we have $\mathcal{R}_0^{\text{syn,ft}}(\text{SUM}_N, b) = \Omega(\frac{\sqrt{N}}{b^2 \log N})$ and $\mathcal{R}_{\epsilon,\frac{1}{5}}^{\text{syn,ft}}(\text{SUM}_N, b) = \Omega(\frac{1}{\epsilon b^2 \log N})$ for $\epsilon \geq \frac{1}{\sqrt[4]{N}}$.*

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. On the surface, it may appear that the complexity of UNIONSIZECP should not be very different from that of UNIONSIZE. This first thought turns out to be incorrect. For $q \leq n$, our technical report [11] presents an $O(\frac{n}{q})$ upper bound protocol for $\mathcal{R}_0^{\text{syn}}(\text{UNIONSIZECP}_{n,q}, \text{poly}(n))$, implying that its communication complexity drops at least linearly with $\frac{1}{q}$. In this protocol, Alice finds the integer j with the smallest occurrence count in X, and sends Bob j and the set $\{i \mid X_i = j\}$. This takes $O(\frac{n}{q} \log n)$ bits in one round, or $O(\frac{n}{q})$ bits in $\text{poly}(n)$ rounds [19]. Now we only need to worry about indices not in the set. For those indices, the promise graph (Figure 5) degrades to a chain, since two edges are removed from the cycle. This makes the UNIONSIZECP problem easy to solve after we apply a mapping trick [11]. To lower bound UNIONSIZECP's communication complexity, we find that the cycle promise makes it challenging to apply classic arguments based on rectangles [24].[5] But we also find that UNIONSIZECP is rather amenable to information theoretical arguments [4], which lead to the following theorem whose proof is in [11]:

THEOREM 4. *$\mathcal{R}_0^{\text{syn}}(\text{UNIONSIZECP}_{n,q}, O(\text{poly}(n))) = \Omega(\frac{n}{q^2 \log n})$ and $\mathcal{R}_{\epsilon,\frac{1}{5}}^{\text{syn}}(\text{UNIONSIZECP}_{n,q}, O(\text{poly}(n))) = \Omega(\frac{1}{\epsilon q^2 \log n})$ for $\epsilon \geq \frac{1}{\sqrt{2n}}$.*

6. 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 $\frac{1}{\epsilon^{0.5-c}}$ 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 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.

Reductions and oblivious reductions. Consider any two-party communication complexity problem Π, where 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,

[5] Leveraging some strong results on the sperner capacity of the cyclic q-gon [6], we managed to obtain some results on $\mathcal{R}_0(\text{UNIONSIZECP})$, but not on $\mathcal{R}_{\epsilon,\delta}(\text{UNIONSIZECP})$.

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 4, 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,[6] while β can be any other node. During the reduction, Alice (Bob) may only send to the other party all those messages sent 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.

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. We now present a strong result on the completeness of UNIONSIZECP:

THEOREM 5. *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 $t \geq 1$, $\mathcal{R}_0^{\text{syn}}(\Pi, t) \leq \mathcal{R}_0^{\text{syn}}(\text{UNIONSIZECP}_{N,\lfloor\sqrt{b/3}\rfloor}, t)$ and $\mathcal{R}_{\epsilon,\delta}^{\text{syn}}(\Pi, t) \leq \mathcal{R}_{\epsilon,\delta}^{\text{syn}}(\text{UNIONSIZECP}_{N,\lfloor\sqrt{b/3}\rfloor}, t)$.*

The full proof is in our technical report [11], and we provide some intuition here. 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 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 9. 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.

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

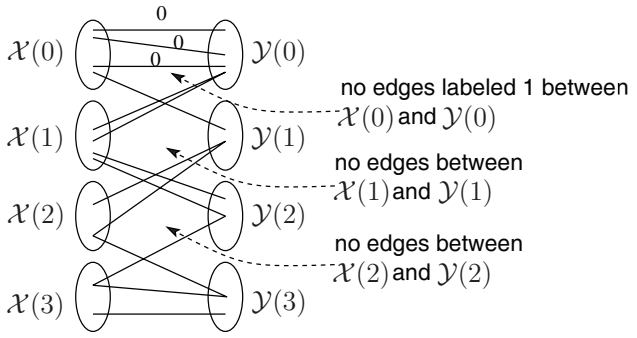

Figure 9: Example assignment graph for a given node τ and for $b' = 4$. $\mathcal{X}(0), \mathcal{Y}(0), \ldots, \mathcal{X}(3)$, and $\mathcal{Y}(3)$ are the 8 subsets, which may have different sizes and different numbers of incidental edges. All edges without labels indicated have a label of 1.

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

Our previous FT lower bounds become trivial when $b > N^{0.25}$ or $\frac{1}{\epsilon^{0.5}}$. 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. We first provide some intuition under a strong *gossip assumption*. Later we will remove this key assumption, which is the key technical challenge addressed by our proof.

Under the *gossip assumption*, the root computes the sum by explicitly collecting from each node a gossip containing its value. We will 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 10). 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 [11] once we insert some dummy nodes into each edge. Our adaptive adversary waits until exactly $\frac{N-1}{2}$ non-root nodes have sent a message (e.g., nodes 1 and 2 in Figure 10). Call these $\frac{N-1}{2}$ nodes as *marked* nodes. The adversary then fails enough edges so that each unmarked non-root 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 nodes has already sent a message, and yet it has one new gossip (from the

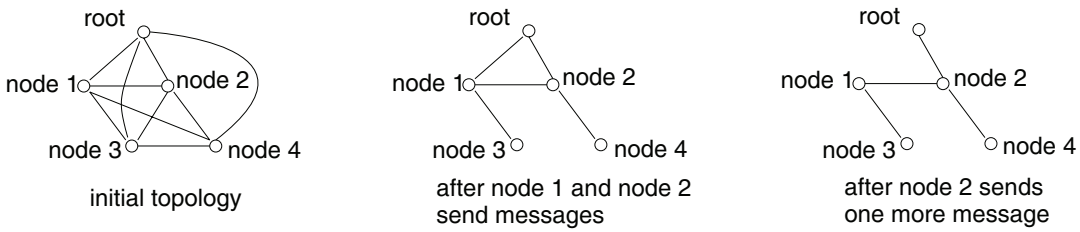

Figure 10: Example FT lower bound topology for $n = 4$ and unrestricted b.

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 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.

The gossip assumption is rather strong. 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. A key challenge in our proof is to properly capture all such possibilities. To do so, we explore a single-player *probing game*, and prove a strong connection between SUM protocols and strategies in this game. We then prove a lower bound on the probing game, which eventually leads to the following FT lower bounds on SUM. See our technical report [11] for the proof of the following theorem:

THEOREM 6. *For any $b \geq 1$, we have $\mathcal{R}_0^{\text{syn,ft}}(\text{SUM}_N, b) = \Omega(\log N)$ and $\mathcal{R}_{\epsilon,\frac{1}{3}}^{\text{syn,ft}}(\text{SUM}_N, b) = \Omega(\log \frac{1}{\epsilon})$ for $\epsilon \geq \frac{1}{N}$.*

8. DISCUSSIONS AND EXTENSIONS

Putting together the NFT upper bounds (Theorem 1) and FT lower bounds (Theorem 2, 3, and 6) will directly give us the exponential gaps, as summarized in Figure 1 from Section 1. Specifically, one only needs to apply Theorem 2 for $1 \leq b \leq 2-c$, Theorem 3 for $2-c < b \leq N^{0.25-c}$ or $\frac{1}{\epsilon^{0.5-c}}$, and Theorem 6 for $b > N^{0.25-c}$ or $\frac{1}{\epsilon^{0.5-c}}$, 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 2.

Total number of failures. Section 2 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.

Private-coin and deterministic protocols. Section 2 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 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 2 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 1 can simply broadcast the result to all nodes along some spanning tree.

Unknown topology. Assuming a known topology, as in Section 2, strengthens our FT lower bounds. For the upper bounds obtained via tree-aggregation, with unknown topologies, it suffices to simply add a distributed pre-processing phase for building a spanning tree.

Defining time complexity over average coin flips. Section 2 defined the time complexity of a protocol to be the number of rounds needed under the worst-case coin flips. Considering worst-case coin flips there was largely for clarity, as in the standard practice [4, 24] of using worst-case coin flips for defining randomized non-zero-error communication complexity. Our technical report [11] shows that defining time complexity using average-case coin flips only affects our results slightly, and our exponential gap continues to hold.

Excluding the communication complexity of the root. Section 2 defined the communication complexity of a SUM protocol to be the number of bits sent by the bottleneck node. Here it is possible for the bottleneck node to be the root. In some scenarios, one may want to exclude the root in this definition. For example, we may be concerned with communication complexity due to the power consumption of the nodes, while the root node (e.g., a base station) may not be operating on battery power. Doing so will not affect any of our theorems, once we extend the lower bound topology by attaching a new degree-1 node to the old root and letting this new node be the root [11].

9. CONCLUSIONS AND FUTURE WORK

Tolerating crash 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?" This paper reveals 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 traditional NFT communication complex-

ity. Since this paper is only the first step along this new direction of FT communication complexity, as one would imagine, the topic is rife with interesting open questions such as:

- Our lower bound 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. Similarly, our completeness result for UNIONSIZECP is for $q = \Theta(\sqrt{b})$, while our reduction actually uses a weaker $q = \Theta(b)$.
- 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 paper (e.g., on the role of failures in the reduction and on the cycle promise) can be valuable.

10. ACKNOWLEDGMENTS

We thank Cheng Yeaw Ku and Y. C. Tay for their valuable help and pointers, and the PODC anonymous reviewers for helpful feedbacks. This work is 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 the research grant MOE2011-T2-2-042 "Fault-tolerant Communication Complexity in Wireless Networks" from Singapore Ministry of Education Academic Research Fund Tier-2, and partly supported by the Intel Science and Technology Center for Cloud Computing (ISTC-CC).

11. REFERENCES

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

[2] O. Ayaso, D. Shah, and M. Dahleh. Information theoretic bounds for distributed computation over networks of point-to-point channels. *IEEE Transactions on Information Theory*, 56(12):6020–6039, 2010.

[3] T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*, 57(7):2748–2761, July 2009.

[4] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, June 2004.

[5] M. Bawa, A. Gionis, H. Garcia-Molina, and R. Motwani. The price of validity in dynamic networks. *Journal of Computer and System Sciences*, 73(3):245–264, May 2007.

[6] A. Blokhuis. On the sperner capacity of the cyclic triangle. *Journal of Algebraic Combinatorics*, 2(2):123–124, June 1993.

[7] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, June 2006.

[8] M. Braverman and A. Rao. Towards coding for maximum errors in interactive communication. In *STOC*, June 2011.

[9] A. Chakrabarti and O. Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. In *STOC*, June 2011.

[10] A. Chandra, M. Furst, and R. Lipton. Multi-party protocols. In *STOC*, April 1983.

[11] B. Chen, H. Yu, Y. Zhao, and P. B. Gibbons. The Cost of Fault Tolerance in Multi-Party Communication Complexity. Technical Report TRA5/12, School of Computing, National University of Singapore, May 2012. Also available at `http://www.comp.nus.edu.sg/~yuhf/TRA5-12.pdf`.

[12] J. Chen and G. Pandurangan. Optimal gossip-based aggregate computation. In *SPAA*, June 2010.

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

[14] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *ICDE*, March 2004.

[15] I. Eyal, I. Keidar, and R. Rom. LiMoSense — Live Monitoring in Dynamic Sensor Networks. In *ALGOSENSORS*, September 2011.

[16] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, September 1985.

[17] A. Giridhar and P. R. Kumar. Towards a theory of in-network computation in wireless sensor networks. *IEEE Communications Magazine*, 44(4):98–107, April 2006.

[18] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[19] R. Impagliazzo and R. Williams. Communication complexity with synchronized clocks. In *CCC*, June 2010.

[20] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, August 2005.

[21] P. Jesus, C. Baquero, and P. Almeida. Fault-tolerant aggregation by flow updating. In *DAIS*, June 2009.

[22] S. Kashyap, S. Deb, K. Naidu, R. Rastogi, and A. Srinivasan. Efficient gossip-based aggregate computation. In *PODS*, June 2006.

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

[24] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1996.

[25] D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In *PODC*, July 2006.

[26] S. Nath, P. Gibbons, S. Seshany, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. *ACM Transactions on Sensor Networks*, 4(2), March 2008.

[27] S. Rajagopalan and L. Schulman. A coding theorem for distributed computation. In *STOC*, May 1994.

[28] L. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996.

[29] D. Woodruff. Optimal space lower bounds for all frequency moments. In *SODA*, January 2004.

[30] H. Yu. Secure and highly-available aggregation queries in large-scale sensor networks via set sampling. *Distributed Computing*, 23(5):373–394, April 2011.