Truthful Information Dissemination in General Asynchronous Networks

Lior Solodkin
Blavatnik School of Computer Science, Tel Aviv University, Israel

Rotem Oshman
Blavatnik School of Computer Science, Tel Aviv University, Israel

Abstract

We give a protocol for information dissemination in asynchronous networks of rational players, where each player may have its own desires and preferences as to the outcome of the protocol, and players may deviate from the protocol if doing so achieves their goals. We show that under minimalistic assumptions, it is possible to solve the information dissemination problem in a *truthful* manner, such that no participant has an incentive to deviate from the protocol we design. Our protocol works in any asynchronous network, provided the network graph is at least 2-connected. We complement the protocol with two impossibility results, showing that 2-connectivity is necessary, and also that our protocol achieves optimal bit complexity.

As an application, we show that truthful information dissemination can be used to implement a certain class of *communication equilibria*, which are equilibria that are typically reached by interacting with a trusted third party. Recent work has shown that communication equilibria can be implemented in synchronous networks, or in asynchronous, complete networks; we show that in some useful cases, our protocol yields a lightweight mechanism for implementing communication equilibria in any 2-connected asynchronous network.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Algorithmic game theory and mechanism design

Keywords and phrases game theory, asynchronous networks, information dissemination

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.37

Funding *Rotem Oshman*: Research funded by the Israel Science Foundation, Grant No. 2801/20.

1 Introduction

Consider a network of n rational players trying to jointly carry out some distributed task, such as computing a spanning tree or electing a leader. Each player has its own desires and preferences: e.g., some players may want to be close to the root of the tree, while others may want to minimize their degree, to reduce traffic flowing through them. The players are given a protocol that can solve the task, but since they are self-interested, they may *deviate* from the protocol if doing so would serve their interests. Can we design a distributed protocol that solves the problem, such that faithfully following the protocol is an *equilibrium* – that is, no player has an *incentive* to deviate from the protocol? This question is at the heart of the recent line of work on *game theory in distributed computing* [2, 3, 1, 5, 6, 15, 4, 23]; we now know that it is possible, for example, to choose a leader on the ring uniformly at random, regardless of the players' individual preferences, and even to do it in a manner resilient to *coalitions* between the players (e.g., [4, 23, 15]).

In this paper we focus on one building block which is used in many distributed protocols: *information dissemination*. We assume that each player i has some piece of information x_i, initially known only to i, and we would like all players to learn all the information of all the players. Information dissemination is typically solved by *flooding* or by *pipelining* [16]. However, these protocols do not work well when the players are self-interested: what is to

prevent a player i from lying about its input, or lying about *other players'* inputs, which i is supposed to forward? In this work we study information dissemination with rational players in general asynchronous networks, and give a simple communication-optimal protocol, under some minimalistic assumptions.

Solving the information dissemination problem does require some assumptions about the *uilities* of the players – their desires and preferences. For example, if player 1's input is $x_1 \in \{0, 1\}$, and player 1's utmost desire is to have all players output that $x_1 = 0$, then whenever $x_1 = 1$, player 1 will lie about its input throughout the protocol; nothing we do can prevent that. Truthful information dissemination is only possible when players have no a-priori reason to lie about their inputs, in the absence of any information about the other players' inputs. However, as soon as anything is learned about the other inputs, a player may have an incentive to lie about its own input, so as to influence the final outcome. For example, mechanisms for auctions often assume *sealed bids*, and may no longer remain truthful if players have information about the other players' bids when placing their own bid.

In addition to the assumption above ("no a-prior reason to lie"), we also need to assume that players want the protocol to succeed, or at least not to abort; a similar assumption, called *solution preference*, is made in prior work (e.g., [4, 23, 5, 14, 6, 15]).

Our protocol shows that these two assumptions together – that player have no a-priori incentive to lie, and that they do not want the protocol to obviously fail – are sufficient to solve information dissemination in a truthful manner in any 2-connected communication network.

Application to communication equilibria. Part of the motivation for our work is to help extend the scope of rational distributed algorithms, beyond algorithms that merely choose a *uniformly random* solution from a set of legal solutions (as in, e.g., [4, 23, 5, 6]). A uniformly random solution is *fair*, but it may not be *good*. For example, instead of electing a uniformly random leader, as was done in [4, 23], we might wish to take the players' preferences into account, and elect the leader that maximizes the players' happiness in some sense. This motivates us to search for efficient distributed implementations of game-theoretic mechanisms for finding a "good" solution, rather than a uniformly random solution.

The particular mechanism we consider here is *communication equilibria* [12], a type of equilibrium that is usually reached with the help of a *mediator*. In classical game theory, a mediator is a trusted entity that helps the players reach an equilibrium: the players communicate with the mediator, and the mediator then suggests a course of action to each player. Even though the players are free to lie to the mediator or ignore its recommendation, it is known that adding a mediator can yield better payoffs for the players – it can enable us to reach an equilibrium in games where the same equilibrium could not be reached without a mediator [12, 8, 9]. For example, mediators are useful in *congestion games* [22], which can be used to model routing of packets in a network (see, e.g., [21]).

In [2, 1] and others it is shown that in some scenarios, a distributed network of rational players can *simulate* a mediator even when we do not have access to a real one; this enables us to implement the rich class of communication equilibria in the distributed setting. This type of simulation is referred to in the game theory community as "cheap talk" – the players can communicate freely before deciding what actions to take. However, so far, all existing literature on implementing a mediator in distributed networks has either assumed that the network is *synchronous*, or that it is *fully-connected*. In addition, previous solutions are heavily based on tools from cryptography – e.g., cryptographic primitives such as *envelope* and *ballot-box* in [17, 19], or *secure multi-party computation* (secure MPC) in [2, 1]. These

tools enable the players to simulate the mediator by securely computing the mediator's output, in such a way that no player finds out the private information of any other player. However, this privacy comes at a cost: the cryptographic primitives used in the literature are fairly heavyweight to implement, and require larger messages and higher running time compared to plain computation where the inputs are not kept private. As a result, in addition to the limitation of having a network that is either synchronous or fully-connected, these protocols also incur a high overhead.

As an application of our algorithm, we identify several natural scenarios where privacy of the inputs is not required to achieve an equilibrium, and information dissemination can be used to implement a mediator, by simply having the players collect all inputs and compute the mediator's recommendations. In these scenarios, our information-dissemination algorithm yields a lightweight, communication-optimal mediator implementation in general, 2-connected asynchronous networks. We prove that our algorithm is indeed communication-optimal, by proving a lower bound on the number of bits required to implement a mediator in a general network (even in synchronous networks).

1.1 Our Results

A truthful information-dissemination protocol. In Section 4, we construct a randomized protocol that solves information dissemination in any 2-connected network, and has the property that no player can improve their expected payoff by deviating from the protocol (that is, following the protocol is an *equilibrium*). Our protocol uses $O(n)$ messages, $O(n^2 \cdot b)$ total communication bits, and $O(D)$ asynchronous rounds, where D is the diameter of the network graph and b is the number of bits required to represent a single input.

Following [10, 13], we do not consider coalitions of players, or Byzantine attacks; these are interesting questions to consider in future work. We conjecture that our protocol extends to coalitions of $k > 1$ players, assuming the network is at least $(k+1)$-connected. We also assume that the network graph is known in advance. This assumption is common to most of the work on game-theoretic distributed computing, with some exceptions, e.g., [5, 6]; these, too, often focus on particular *classes* of graphs, such as rings.

Application to computing communication equilibria. We refer to the type of equilibria that can be computed by our protocol as *full-information communication equilibria*. In Section 6, we give two natural scenarios where a given communication equilibrium is also a full-information equilibrium: the first is any game where a player's payoff depends only on its own output (and not the outputs of the other players), and the second is any distributed task where

(a) the players prefer to reach a valid solution, and
(b) in any valid solution, given the outputs of any set of $n-1$ players, there is a unique valid output for the last player.

The second scenario includes *agreement* problems, such as leader election, but also other useful distributed primitives.

Lower bounds. One might wonder whether it is *necessary* to collect all players' inputs in order to achieve a communication equilibrium. To complement our results, we prove that the answer is "yes": there is some game with a full-information communication equilibrium that requires $\Omega(n)$ messages and $\Omega(n^2 \cdot b)$ total bits of communication to achieve, where b is the number of bits required to represent the players' types and actions. This means that using our information-dissemination protocol to compute full-information communication

equilibria is optimal, in the worst-case. We also prove that the assumption of 2-connectivity, which is made by our protocol, is necessary: in any network that is not 2-connected, there is some game with a full-information communication equilibrium that cannot be achieved.

A remark about our solution concept. In the game theory community, many flavors of equilibria have been considered. Following [4], the solution concept we adopt in this paper is *sequential equilibrium* [18]. Our protocol, like that of [4], involves "collective punishment": if a player detects that another player has deviated from the protocol, it aborts the protocol, thereby punishing *itself* as well as all the other players. One may view this as a "non-credible threat" – why should any rational player actually carry out such an action? Following [4], we showed that our protocol achieves a *sequential equilibrium* [18], a solution concept formulated to handle non-credible threats. However, for the sake of accessibility, in the current paper we focus on the weaker but more familiar concept of a *Nash equilibrium*.

2 Related Work

Information dissemination. Information dissemination is a central building block in distributed algorithms, and a goal in itself. In fault-free sychronous networks, a communication and time-efficient solution is *pipelining* [16], and the problem has also been studied in a variety of other network models.

To our knowledge, the only work to consider information dissemination in the game-theoretic setting is [5], where the problem is referred to as *knowledge sharing*. Several protocols for knowledge sharing are given in [5]: for synchronous rings, for synchronous or asynchronous complete networks, and for synchronous general networks. General asynchronous networks are not considered in [5]. Moreover, [5] makes a strong assumption, called *full knowledge*, about the protocol that *uses* knowledge sharing as a building block: "for each agent that does not know the values of **all other agents**, any output of the protocol is still equally possible" [5] (emphasis ours). In particular, this implicitly assumes that the protocol that uses the knowledge-sharing building block chooses a uniformly random solution. In contrast, here we assume only that agents have no incentive to lie about their input in the absence of *any* information about the other players' inputs; as soon as a player has learned even a single bit of another player's input, it may have an incentive to lie about its own input.

Implementing a mediator. Our paper studies distributed implementations of a mediator by cheap-talk protocols. This topic was first investigated by the economics community; the most directly relevant work to ours is [10], which solves the same problem in a similar setting, but it requires a fully-connected, synchronous network, and has high message complexity, $\Theta(n^2)$. [13] gave a protocol that implements rational secret sharing and secure multi-party computation (secure MPC), and these results were extended in [2] to protocols that withstand a coalition of rational players, or some players that are Byzantine instead of rational. A mediator was implemented in [2] using secure MPC, but [2] required again a fully-connected synchronous network. In [3], lower bounds were given for the size of a coalition we can withstand when implementing a mediator.

The first work to implement a mediator in an asynchronous network was [1], which showed that in a fully-connected asynchronous network, we can implement a mediator and even handle coalitions and non-rational players. Since the protocols of [1] again rely on secure MPC, their message complexity is $O(n \cdot N \cdot c)$, where n is the number of players, N is the number of messages in the mediated game, and c is a bound on the number of

arithmetic gates required to implement the mediator as an arithmetic circuit. In this paper, by considering the restricted class of *full-information* communication equilibria, we obtain an improved message complexity of $O(n)$, and handle general asynchronous networks.

We remark that some of the works mentioned above achieve different solution concepts than the type of equilibrium we aim for in this paper (sequential equilibrium). For example, [13, 2] both obtain a Nash equilibrium that is resilient to the iterated deletion of weakly-dominated strategies. This type of equilibrium, unlike sequential equilibrium, is not concerned with what action the player should take in states that occur off the equilibrium path.

Other related work. Several recent works considered fair solutions to specific distributed problems: electing a uniformly random leader was studied in [4, 23], and other problems and building blocks were considered in [5, 14, 6, 15]. The solution concept we suggest in this paper can solve some of these problems more generally: for example, choosing a uniformly random leader is one type of full-information communication equilibrium, which ignores the utilities of the players, but there are other equilibria that take the utilities into account (for example, we can maximize the *social welfare*, the sum of the players' utilities). On the negative side, our protocol is not resilient against coalitions, while several of the aforementioned works can handle coalitions and even Byzantine players. Also, like much of the prior work, we do assume that the network graph is known to all the players in advance. (This is also assumed in the secure MPC implementation of [20].)

3 Preliminaries

Games and protocols. In this paper we are concerned with two types of games: the game we are trying to implement – that is, the problem specification – is a *normal-form Bayesian game*, where players choose one action, and then their utility is determined by the actions that all the players have taken. Our implementation, a protocol for asynchronous networks, is formally modeled as an *extensive-form Bayesian game*, which represents all possible executions in the asynchronous network, and captures the interactive nature of such executions. For the sake of brevity, we define only normal-form Bayesian games here, as extensive-form games are not necessary to understand our protocol. (We touch more on extensive-form games in Section 5, where we give a high-level overview of the correctness proof for our protocol.)

Notation. Given an n-tuple $s = (s_0, \ldots, s_{n-1})$, we let $s_{-i} = (s_0, \ldots, s_{i-1}, s_{i+1}, \ldots, s_{n-1})$ denote the $(n-1)$-tuple obtained by omitting s_i. When i is clear from the context, the notation (s_i, s_{-i}) represents the original n-tuple s.

Normal-form Bayesian games. A normal-form (or strategic-form) Bayesian game is defined as follows. Note that the *types* referred to in the definition correspond to *inputs* in our case.

▶ **Definition 1.** *A* normal-form Bayesian game, $\Gamma = (N, (T_i)_{i\in N}, p, (A_i)_{i\in N}, (u_i)_{i\in N})$, *consists of:*

1. *A set of players N. We usually assume for simplicity that $N = [n]$, where $n \geq 2$ is the number of players.*
2. *For each player $i \in N$, a set of possible types (inputs) T_i. Let $T = T_0 \times \ldots T_{n-1}$.*
3. *A probability distribution $p : T \to [0, 1]$ over the types of the players.*
4. *For each player $i \in N$, a set of possible actions A_i. Let $A = A_0 \times \cdots \times A_{n-1}$.*
5. *For each player $i \in N$, a utility function $u_i : T \times A \to \mathbb{R}$.*

The game is played as follows: first, the types are drawn from their distribution, $t = (t_0, \ldots, t_{n-1}) \sim p$. Each player i is given only its own type t_i, but the prior p is known to all players. Next, each player i independently chooses an action $a_i \in A_i$, and the utility of each player is then determined by $u_i((t_0, \ldots, t_{n-1}), (a_0, \ldots, a_{n-1}))$.

Strategies and equilibria. A *mixed strategy* for player i in the game Γ is a function mapping player i's type $t_i \in T_i$ to a probability distribution over player i's actions A_i; a *strategy profile* is an n-tuple $s = (s_0, \ldots, s_{n-1})$ assigning to each player $i \in N$ a mixed strategy s_i. Player i's *expected utility* for the game Γ under the strategy profile $s = (s_0, \ldots, s_{n-1})$ is given by $\mathbb{E}_{t\sim p|t_i, a\sim s(t)}\left[u_i(t, a)\right]$. Here, t_i denotes player i's type, and $s(t)$ denotes the distribution on n-tuples of actions where the i-th element is sampled independently from $s_i(t_i)$.

A strategy s_i for player i is called *best response* to $s_{-i} = (s_0, \ldots, s_{i-1}, s_{i+1}, s_{n-1})$ if it maximizes player i's expected utility, assuming that the other players play according to s_{-i}.

A strategy profile $s = (s_0, \ldots, s_{n-1})$ is called a (mixed) *Nash equilibrium* if for each player i, the strategy s_i is player i's best response to s_{-i}.

Mediators. Some normal-form games may have strategy profiles that are desirable for some reason – for example, they might lead to high social welfare – but which are not Nash equilibria. In such cases, it can be helpful to enlist the help of an external *mediator*. Intuitively, a mediator is a trusted entity; the players tell their types to the mediator, and the mediator then provides each player with a recommended action. The players are free to lie to the mediator or ignore its recommendation, but a good mediator will render such deviations not profitable (in expectation).

Formally, a mediator d is a mapping from n-tuples of types to distributions on n-tuples of actions, and we would like it to satisfy the following:

▶ **Definition 2.** *A mediator d is a* communication equilibrium *of normal-form Bayesian game Γ if for all $i \in N, t_i, t_i' \in T_i, \alpha_i : A_i \to A_i$:*

$$\sum_{t_{-i}\in T_{-i}} p(t_{-i}|t_i) \sum_{a\in A} d(a|t) u_i(t, a) \geq \sum_{t_{-i}\in T_{-i}} p(t_{-i}|t_i) \sum_{a\in A} d(a|t_{-i}, t_i') u_i(t, (\alpha_i(a_i), a_{-i})).$$

Here, $d(a|t)$ represents the probability that the mediator returns $a \in A$ when the players send it $t \in T$, and $d(a|t_{-i}, t_i')$ represents the probability that the mediator returns $a \in A$ when player i sends t_i', and the other players send t_{-i}.

Intuitively, the definition asserts that for each type t_i, there is no "lie" t_i' that player i could tell the mediator, or different action $\alpha(a_i)$ that player i could take instead of the recommendation a_i of the mediator, that would increase player i's expected payoff.

Full-information equilibria. We introduce a subclass of communication equilibria, characterized by being resilient to the revelation of the other players' types and recommended actions:

▶ **Definition 3** (Full-information communication equilibrium)**.** *A mediator d is a* full-information communication equilibrium *for a game $\Gamma = (N, (T_i)_{i\in N}, p, (\tau_i)_{i\in N}, (A_i)_{i\in N}, (u_i)_{i\in N})$ if for each player $i \in [n]$ and type $t_i \in T_i$,*

1. *The player has no incentive to lie about its type: for each $t_i' \in T_i$, and $a_i' \in A_i$,*

$$\sum_{t_{-i}\in T_i} p(t_{-i}|t_i) \sum_{a\in A} d(a|t) u_i(t, a) \geq \sum_{t_{-i}\in T_i} p(t_{-i}|t_i) \sum_{a\in A} d(a|t_{-i}, t_i') u_i(t, (a_i', a_{-i})).$$

2. *After receiving the mediator's recommendation and learning all the other players' types, the player is still not incentivized to deviate: for each* $t_{-i} \in T_{-i}, a \in \sup(d(\cdot|t_i, t_{-i}))$, $\alpha_i : T \times A_{-i} \to A_i$,

$$u_i((t_i, t_{-i}), a) \geq u_i((t_i, t_{-i}), (\alpha_i((t_i, t_{-i}), a), a_{-i})).$$

Here, $\sup(d(\cdot|t)) \subseteq A$ *denotes the support of the mediator's distribution* d *when the players send it the types* t.

The difference from the standard definition of communication equilibria is in the second condition, where we reveal to player i the types and actions of the other players, not just its own recommended action.

Punishment actions. When implementing a general communication equilibrium, we need to assume that players have some means to retaliate against players that deviate from the protocol. As is standard (see, e.g., [10]), we assume that the game contains a *punishment action*, and any outcome where some player carries out this action yields the worst possible payoff for all players.

The information dissemination problem. We model information dissemination as a Bayesian game, Γ_{info}, where the *actions* of the players consist of outputting n-tuples of types, or aborting the protocol; that is, if $T_1, \ldots, T_n$ are the possible types (i.e., inputs) of each of the n players (respectively), then the actions of each player i are $A_i = (T_1 \times \ldots \times T_n) \cup \{\mathsf{abort}\}$. The utility function u_i of each player i then takes the true types $t \in T_1 \times \ldots \times T_n$ of the players, and the outputs $t'_1, \ldots, t'_n \in T_1 \times \ldots \times T_n$ of the players, and returns the payoff $u_i(t, (t'_1, \ldots, t'_n))$ of player i.

As we explained in Section 1, a few assumptions about the utilities of the players are necessary. First, we assume that abort is a *punishment action*, as explained above (that is, players prefer to avoid it above all else). The second assumption is that players have no a-priori incentive to lie about their input, and also that, having learned the other players' inputs, they will correctly output what they have learned (instead of outputting something else). The cleanest and most general way to model this assumption is to define a specific mediator, which we call d_{info}, which gives each player the types of all the other players, and require that the utilities be such that d_{info} is a full-information communication equilibrium. This captures the requirement that in the absence of any information about the types of the other players, no player has an incentive to lie about its own type, and that once the types are learned, the player should output them correctly.

▶ **Definition 4** (The information-dissemination mediator, d_{info}.)**.** *Let* $d_{\mathsf{info}} : T_1 \times \ldots \times T_n \to ((T_1 \times \ldots \times T_n) \cup \{\mathsf{abort}\})^n$ *be the mediator for* Γ_{info} *that returns to each player the types of all the other players:* $d_{\mathsf{info}}(t_1, \ldots, t_n) = (t_1, \ldots, t_n)^n$.

We say that Γ_{info} is a *feasible information-dissemination game* if d_{info}, the mediator from Definition 4, is a full-information communication equilibrium for Γ_{info}.

4 The Protocol

We now present the protocol that players are supposed to follow. Our protocol can implement any given full-information communication equilibrium d for a normal-form Bayesian game Γ, and in particular, the protocol solves the information-dissemination problem: if Γ is a feasible information-dissemination game (as defined in Section 3), then we can have the players simulate the mediator d_{info} to output the types of all the other players.

Throughout the protocol, if any player i detects a deviation from the protocol by any other player, player i irrevocably decides to take its punishment action, guaranteeing the worst possible payoff for all players. In particular, player i chooses to punish if it receives a message from player j that *could not have been sent* in accordance with the protocol in any execution (e.g., if player j is not supposed to message player i at this stage of the protocol, or if the message is ill-formatted).[1]

4.1 High-Level Overview

The goal of our protocol is to get all players to truthfully reveal their types to the entire network, and then have each player locally simulate the mediator and output the action the mediator recommends (or, if we simply want to solve information dissemination, output the types that it has learned). We must manage the process of revealing the players' types carefully: for example, if player i learns the type of player j before player i has said anything about its own type, then player i may have an incentive to lie about its type in order to improve its expected utility. (Recall that the definition of a communication equilibrium guarantees that the players have no incentive to lie in the absence of *any* information about the other players' types; once they have even partial information, all bets are off.)

The main idea underlying our protocol is to

(a) use a basic form of secret sharing to have players *commit* to their type, so that they cannot lie about their type in the future; and, at the same time,

(b) ensure that no player i can learn anything about the type of another player $j \neq i$ before player i has committed to its type.

The protocol has four main stages, which we describe here at a high level, omitting many details. A more detailed description will be given next.

Commitment on a cycle. We fix in advance some shortest cycle C in the graph (recall that the graph must be 2-connected). Assume w.l.o.g. that cycle C comprises players $0, 1, \ldots, \ell$, in this order (with $\ell + 1$ being the length of the cycle).

In the first stage of the protocol, players 0 and 1 *commit to their types*, by "splitting" each type into two shares, each of which reveals nothing about the type by itself. Together both shares reveal the type. The shares are sent along opposite sides of the cycle, in a way that ensures that neither player can receive both shares of the other player before it has sent out both of its own shares, thereby committing to its type in a way that will reveal any future attempt to lie about it.

Commitment on a tree. For the next part of the protocol, we fix in advance a spanning tree T rooted at player 0 and excluding player 1. We proceed through the tree top-down, and have each player interact with its children in the tree in a pairwise commitment protocol, where the parent and the child reveal their types to one another. We ensure that a player never reveals its type to its parent in the tree before the parent is committed – when we start this phase, player 0 is already committed, and we maintain this invariant as we proceed

[1] We note that player i cannot always detect the fact that player j sent it a message when j was not supposed to, because player i does not know where player j is in its execution of the protocol, but in some cases it is obvious – e.g., if player j is supposed to wait for a message from player i before sending a message to i, and i has not yet sent that message. We address the "undetectable" cases in our proof that the protocol is an equilibrium.

down the tree. At the same time, we make sure the player does not *know* its parent's type, only that the parent has already committed to the type. Thus, neither parent nor child have an incentive to deviate from the protocol.

Revealing the types. By the end of the previous part of the protocol, each player has revealed its type to at least one other player. We now use the spanning tree to share all the types with all the players, by simply collecting them up the tree and then broadcasting them downwards.

Detecting deviations. The last stage of the protocol checks whether any player has been "two-faced" and claimed that it has different types at different points in the protocol, or whether some player has tried to lie about the type of another player. To do this, we fix a sparse 2-connected subgraph G' of the network graph in advance, and simply have each player send all the types it received in the previous stage to its neighbors in G'. Each player verifies that the types its neighbors have learned match what it has learned itself. Since G' is 2-connected, between every two players $j, k \neq i$ there is a path π that does not contain i; if player i has given "two different versions" of its type to players j, k, this inconsistency will be discovered, as two neighboring nodes along the path π will have received different versions of i's type.

4.2 Detailed Description

We now give a detailed description of the protocol. Let b be the number of bits required to represent the type of a player, and let R be the number of random bits used by the mediator.

Each player i starts the protocol with a *secret*, denoted s_i. For all players except players 0 and 1, the secret is simply the player's type, $s_i = t_i$. Players 0 and 1 play a special role in the protocol, and at the beginning of the protocol, they generate random strings that will be used later in the protocol. These random strings are part of the secrets of players 0 and 1: for each $i \in \{0, 1\}$, the secret s_i of player i is a tuple consisting of

- The type $t_i \in \{0,1\}^b$ of player i;
- A private random string $r_i \in \{0,1\}^R$, which will later be used to simulate the mediator; and
- In the case of player 1, an additional private random string $v \in \{0,1\}^b$, which will be used in the next stage of the protocol.

Each player $i \in [n]$ has a local array called *values*, of length $n + 1$, where player i stores the secrets of the other players as it learns them. Each player i initializes all cells in the range $\{0, \ldots, n-1\}$ to $\perp$, except for cell i, which player i initializes to $values[i] = s_i$. The last cell, $values[n]$, is initialized to $\perp$ by all players except player 1, who sets $values[n] = v$.

During the algorithm, cells are updated only by calling the subroutine **Store**(i, x), which ensures that once a value is written to the *values* array, any future attempt to overwrite it with a different value will cause the player to execute the punishment action. The notation $values_i[j]$ refers to cell j of the local *values* array of player i.

Step 1: Commitment in a Cycle

Let C be some fixed shortest cycle in the network graph, and assume w.l.o.g. that the nodes of C are consecutively named $0, 1, \ldots, \ell$. In this part of the protocol, players 0 and 1 commit to each other, as follows: first, each player splits its secret s_i into two shares – a uniformly

random "key", $k_i \in \{0,1\}^{|s_i|}$, and an "encryption" of its secret, $e_i = s_i \oplus k_i$. The secret s_i can be reconstructed by taking $k_i \oplus e_i$, but each share k_i, e_i by itself is uniformly random and conveys no information about s_i.

The goal now is for players 0,1 to exchange their shares in such a way that neither player receives *both* of the other player's shares before it has sent out both of its own shares. That is, player 0 can receive *either* k_1 or e_1 before sending out both of its shares, but it should not receive both (and vice-versa). To achieve this, we have each player send one of its shares directly to the other player, and the other share is sent along the other side of the cycle. The order in which shares are sent is orchestrated carefully:

1. First, player 0 sends e_0 to player 1.
2. After receiving e_0, player 1 releases both of its shares: it sends k_1 to player 0, and e_1 to player 2.
3. After receiving k_1, player 0 sends k_0 to player ℓ. At the same time, e_1 is forwarded along the cycle from player 2 to player ℓ.
4. Agent ℓ waits until it has received both k_0 and e_1. Only then does it forward the two messages, sending e_1 to player 0 and k_0 to player $\ell - 1$.
5. Finally, k_0 is forwarded along the cycle from player $\ell - 1$ to player 1.

After receiving e_0 and k_0, player 1 stores the secret $s_0 = e_0 \oplus k_0$ in $values_1[0]$, and player 0 stores $s_1 = e_1 \oplus k_1$ in $values_0[1]$.

Step 2: Commitment in a Tree

Let T be a precomputed BFS tree over the network $G\backslash\{1\}$, rooted at player 0 and excluding player 1. Let H be the height of T. In this part of the protocol, we proceed through the breadth-first layers of T in a top-down manner, and at each step, each player in the current layer executes a short type-exchange protocol with its children in the tree, learning their types and revealing its own type to them. The result is that after $0 \leq h \leq H$ steps, each player i at distance at most h from player 0 in T has revealed its type t_i to some other player $j \neq i$ (where j is the parent of i in the tree if $i \neq 0$, and $j = 1$ if $i = 0$). We will later use this commitment to verify that player i has truthfully revealed its type to the entire network.

Recall that at the beginning of the protocol, player 1 chose a random string v, and this string was revealed to player 0 at the end of the cycle-commitment stage (because it is part of player 1's secret). We now use v as a "secret key" that parents use to commit to their type when they interact with their children in the tree. The value of v itself is also propagated down the tree, so that at the end of this stage, all nodes of the network know it; thus, we can later verify that all commitments were honored.

Let $i \neq 0$ be some player in the tree, and let p be the parent of i. The parent-child exchange protocol between players i and p is executed as follows:

1. The parent p commits to its type by sending its child the message $e_p = t_p \oplus v$.
2. The child i responds by sending its type t_i to p. The parent uses **Store** to store this value in $values_p[i]$. Note that at this point, i knows nothing about the types of the other players, so it has no incentive to lie about its type.
3. The parent "unpacks" its commitment by sending its type t_p to the child, and the child stores t_p in $values_i[p]$ and $e_p \oplus t_p$ in $values_i[n]$.

We point out one subtlety of the protocol: there is nothing to stop a player p in the tree from "fishing for information" by prematurely contacting its child i, sending it a garbage message, and eliciting in return the type of player i. However, our protocol safeguards against this behavior, by making sure that player p has no incentive to do so: if p sends a "garbage message" instead of $e_p = t_p \oplus v$, then the child will discover this later on, when both t_p and v are revealed to all players, and it will punish p.

Steps 3 and 4: Revealing the Types and Detecting Deviations

Once the second stage of the protocol completes, every player in the network is committed to its type, and it is now safe to reveal all the types to everyone. This is carried out by first collecting all the secrets up the tree to the root, and then broadcasting all the values down the tree, so that each player in the network learns the secrets of all the other players. As they learn new types (or more generally, secrets), players use the **Store** function to store them in their local *values* array. Player 0, who is the root of the tree, sends $values_0$ to player 1, who is not in the tree. At the end of this step, every player verifies that its *values* array is full: if player i has some cell j such that $values_i[j] = \perp$, then player i takes its punishment action in the underlying game.

In the final stage of the protocol, we disseminate the secrets collected in the previous stages on a sparse 2-connected subgraph G' of G, and then have each agent verify that in the previous stage, it received the same values as its neighbors did.

To construct G', we show that any 2-connected graph on n vertices has a spanning, 2-connected subgraph with $2n$ edges:

▶ **Proposition 5.** *Suppose that G is a 2-connected graph on n vertices. Then there is a spanning 2-connected subgraph G' of G with $2n$ edges.*

The protocol for detecting whether any player has been inconsistent is simple: each player i compares its *values* array with its two neighbors in G', and if any neighbor has a different *values* array (that is, if even one cell is different), player i executes its punishment action. If this step succeeds at all players that follow the protocol, and at most one player deviates, then all these players will use the same *values* array when they simulate the mediator (next).

In addition, player i verifies that, if $values_i[1] = (\tilde{t}_1, \tilde{r}_1, \tilde{v})$ (recall that player 1's secret has three fields), then $values_i[n] = \tilde{v}$. If this step succeeds, it ensures that i's parent p has honored its commitment, that is, it correctly sent $e_p = t_p \oplus v$ when it first communicated with player i.

Taking action. All players know in advance the mediator function d, which takes a vector $t \in \{0,1\}^{n \cdot b}$ of n types and a string $r \in \{0,1\}^R$, and returns an n-tuple of the mediator's recommended actions on types t using r as the mediator's randomness.

Once the protocol is completed, each player i extracts from its *values* array the types of all the players, and the random strings r_0, r_1. It now simulates the mediator by computing $a = d(t, r_0 \oplus r_1)$, and outputting its recommended action a_i.

5 Sketch of the Correctness Proof

We give an informal overview of the proof that our protocol correctly implements the given full-information communication equilibrium.

Protocols as extensive-form games. As we mentioned in Section 3, we model a distributed protocol as an *extensive-form Bayesian game*, which represents all possible executions of the protocol. An extensive-form game is a tree, where each vertex represents a possible state of the system, and the edges of the tree represent actions of players or the environment (the scheduler). An extensive-form game Γ' is called an *extension* of a normal-form game Γ if the types (inputs) in Γ' are the same as in Γ, and each leaf of Γ' is labeled with an action profile $(a_1, \ldots, a_n) \in A_1 \times \ldots \times A_n$ from Γ. In other words, we think of Γ' as "filling in" what happens in Γ between the point where players are assigned their types, and the point where players output their actions.

Informally, we say that a protocol P *implements* a communication equilibrium d for a normal-form game Γ if, in the extensive-form game induced by P and Γ, for every possible schedule,

(a) following the protocol P is a Nash equilibrium (no player has an incentive to deviate), and

(b) the actions output at every leaf are chosen according to d (that is, if the players' types are t, then the actions output are distributed according to $d(\cdot|t)$).

Proving that our protocol implements the desired communication equilibrium. It is not difficult to see that if all players follow the protocol, then the distribution of their actions is exactly the distribution that the mediator would produce; we focus on proving that our protocol is a Nash-equilibrium, that is, no player has an incentive to deviate from the protocol at any point.

The key idea in our proof is that at certain points in the execution of the protocol, a player becomes "locked in" to a type (and the random strings r_0, or r_0 and v, in the case of players 0 and 1 respectively). Following this point, the player can no longer fool the other players about its type. Formally, given a vertex u of the game tree, we say that *player i is committed to value x in u* if player i has already executed the following actions in u:

- Agent 0: has sent messages m_1, m_ℓ to players 1 and ℓ (resp.), and the first such messages sent have $m_1 \oplus m_\ell = x$.
- Agent 1: has sent messages m_0, m_2 to players 0 and 2 (resp.), and the first such messages sent have $m_0 \oplus m_2 = x$.
- Agent i for $i \neq 0, 1$: either
  - Agent i has sent at least one message to its parent in the tree, and the first message sent to the parent is x, or
  - Agent i has sent at least one message m to one of its children in the tree, and the contents of the first such message satisfies $m \oplus v = x$.[2]

We point out a subtlety in the definition above: under the protocol, a player $i \notin \{0, 1\}$ is not meant to contact its children in the tree before it has finished executing the parent-child protocol with its own parent. Thus, if i follows the protocol, it commits to its type by revealing its type to its parent, not by sending a message to some child. However, if i decides to deviate, it might try to gain some advantage by contacting its children before it is supposed to, in order to elicit their types from them before it commits to its own value. The children cannot detect this, as they cannot know when i has already revealed its type to its parent in the tree. Nevertheless, i cannot gain by doing so: as soon as i sends a message m to some child, it is effectively "locked in" to the value $m \oplus v$, since v will be revealed at the end of the protocol. Even if the child then responds with its true type, it is too late for i to "change its mind" about the value to which it committed.

We prove that "commitments are real", in the sense that if player i is committed to some value, then this value will eventually appear in the *values* array of some other player, even if player i deviates from the protocol:

▶ **Lemma 6.** *Let u be a leaf that is reached in a run where only player i deviates, and let u' be a vertex on the path to u. Suppose that player i is committed to value x at u'. Then in u, either some player carries out a punishment, or there is a player $j \neq i$ such that $values_j[i] = x$.*

[2] We assume for convenience that the private randomness of the players is fixed at the beginning of the run, so that the value v is defined even if player 1 has not yet taken a step.

Observe that players that do not deviate from the protocol cannot be stopped from committing to their true type, even if some other player deviates: under the protocol, the value to which a player commits does not depend on any messages it receives.

Next, we show that if the protocol completes without any player carrying out a punishment, then even if player i has deviated, all other players agree on their *values* arrays:

▶ **Lemma 7.** *Let u be a leaf that is reached by a run where only player i deviates, and suppose that no player takes a punishment action in u. Then in u, any two players $j \neq k \in [n] \setminus \{i\}$ have $values_j = values_k$.*

Intuitively, this is because the *values* arrays are propagated along the edges of the two-connected subgraph G', so any inconsistency will be detected along a path that does not include player i. Combined with the previous lemma, we now see that if player i is committed to some type, then this type will eventually appear in *all* players' *values* arrays, even if player i deviates (unless some player carries out a punishment, which is never in player i's interest). This means that the value x will be the value used when calling the mediator, and after committing, player i can do nothing to change that, short of deviating from the protocol in an obvious way that would cause some player to punish it. For all players that follow the protocol, their true types will be used (as they commit to those values). It follows that after committing to a type, player i has no incentive to deviate from the protocol, because doing so cannot improve its expected utility.

The next part of our proof deals with deviations that might occur prior to committing to a type. In particular, we must rule out the possibility that a player i that has not yet committed to its type decides to lie, and commit to a value other than its true type. To rule out such deviations, we prove that for each player i, at any point in the run where player i has not committed to its type, its belief about the other players' types is unchanged from the prior (in other words, player i does not gain any information before committing – even if player i deviates from the protocol).

This means that before player i commits to a type, it is effectively in the same situation that it would be in at the beginning of the mediated game: it knows only its own type and the prior distribution of the other players' types. Since the mediator is assumed to be a communication equilibrium, player i has no incentive to lie about its type in this situation. Formalizing this intuition and making its precise is somewhat involved, since player i can deviate in many ways that do not immediately translate to "lying about its type" under the protocol; nevertheless, we show that any strategy that player i might employ can be translated into a distribution on types that player i might send to the mediator in the mediated game, while following the protocol translates to telling the mediator player i's true type. We are therefore able to show that no strategy other than following the protocol can improve player i's expected payoff.

6 Examples of Full-Information Communication Equilibria

We point out two classes of normal-form Bayesian games Γ, such that any communication equilibrium for Γ is also guaranteed to be a *full-information* communication equilibrium (that is, any mediator d that satisfies Definition 2 for Γ also satisfies Definition 3.) The proofs are straightforward, and they are omitted here.

Constrained games. The first class we consider are games where given the behavior of all the other agents, there is only one "good" action that player i can take.

The games in this class involve a set of *legal outcomes* $L \subseteq A_1 \times \ldots \times A_n$, and we require that players prefer to reach a legal outcome above all other considerations. This is referred to as *solution preference* [4], and is assumed by most work on rational distributed computing. Formally, the requirement is that for any player i, type profile $t \in T_1 \times \ldots T_n$, legal outcome $a \in L$ and illegal outcome $a \notin L$, we have $u_i(t, a) \geq u_i(t, a')$.

▶ **Definition 8** (Constrained games). *We say that a set of legal outcomes $L \subseteq A_1 \times \ldots \times A_n$ is* constrained *if for any player i and legal outcome $(a_i, a_{-i}) \in L$, there does not exist any action $a'_i \neq a_i$ such that $(a'_i, a_{-i}) \in L$.*

A game Γ is called constrained *if there exists a set of outcomes $L \subseteq A_1 \times \ldots \times A_n$, such that Γ has the solution-preference property with respect to L, and L is constrained.*

Examples of this class include agreement problems such as leader election and consensus, where the set of legal solutions requires all players to agree on an output, but also other problems, depending on the output specification; we can turn any problem into a constrained game by simply having each node output its neighbors' actions in addition to its own. For instance, computing a spanning tree can be cast as a constrained game, by having the output (action) of each player include both its children and its parent in the tree. This version of the spanning-tree problem makes explicit the intuition that in any legal solution, if node u claims v as its parent, then v should agree that u is its child, and vice-versa.

Games with local utility functions. The second class we consider are games where each player's payoff depends only on its own type and action, that is, for any two type profiles $t, t' \in T$ and action profiles $a, a' \in A$, if $t_i = t'_i$ and $a_i = a'_i$, then $u_i(t, a) = u_i(t', a')$. Examples of games in this class include resource-allocation problems (e.g., dynamic spectrum allocation), where players only care about the resources allocated to them, and do not care about the allocation of the remaining resources to the other players. In local games, revealing the other players' recommended actions does not provide any additional incentive for player i to deviate from the mediator's recommendation; it is therefore not hard to show that any communication equilibrium is also a full-information communication equilibrium.

7 Communication Lower Bound

In this section we show that for some network graphs and games, every protocol that achieves a given full-information equilibrium must send $\Omega(n^2 \cdot b)$ bits in total. This holds even if the actions of the players are "short", requiring b bits to represent.

▶ **Theorem 9.** *For every $n \geq 1, b \geq 1$, there is a $2n$-player normal-form Bayesian game Γ with b-bit types and actions, and a full-information communication equilibrium d for Γ, such that any protocol that implements d must send $\Omega(b \cdot n^2)$ bits in expectation on a ring.*

Proof sketch. Consider the following $2n$-player Bayesian game Γ, where the players are given by $\{0, \ldots, 2n-1\}$, and each player has 2^b possible types, denoted $\{0, \ldots, 2^b - 1\}$. The types of the players are iid uniformly random. The actions available to each player are also given by $\{0, \ldots, 2^b - 1\}$.

Given a player $i \in [2n]$, the *opposite player* of i, denoted $-i$, is player $(i+n) \bmod 2n$. We define the utility u_i of player i to be 1 if player i outputs the type t_{-i} of its opposite player, and 0 otherwise. It is not hard to show that for this game, $(1, \ldots, 1)$ is a full-information communication equilibrium, which is achieved by having the mediator tell each player the opposite player's type.

Now consider a distributed implementation of this equilibrium. Suppose the $2n$ players are arranged consecutively in a ring, and consider only *synchronous* executions, where the scheduler lets the players run in round-robin order, and every message that is sent by player i to player j is delivered the next time player j takes a step. Fix a protocol P which achieves utility $(1, \ldots, 1)$ in all runs (in fact, our proof can be extended to protocols that achieve utility $(1, \ldots, 1)$ with constant probability, see Appendix A). Every run of P must end with player i correctly outputting the type of the opposite player, t_{-i}. Intuitively, this means b bits of information must flow from $-i$ to i, and they must be repeated along every edge between $-i$ and i; the distance between players i and $-i$ is n, so the total number of bits sent is $\Omega(n \cdot b)$, and these bits only "help" players $i, -i$, so every other pair of players must also send $\Omega(n \cdot b)$ bits of their own. The total communication complexity is therefore $\Omega(n^2 \cdot b)$.

The formal argument is given in Appendix A. It uses the technique of [11], where we consider each balanced cut in the graph, and argue by reduction to two-party communication complexity that $\Omega(n \cdot b)$ bits must flow across the cut. Since there are $\Theta(n)$ balanced cuts in the ring, and each edge appears in exactly one, the total communication is $\Omega(n^2 \cdot b)$ bits. ◀

8 Necessity of Two-Connectivity

Finally, we show that there is some full-information communication equilibrium that cannot be implemented in any network that is not 2-connected. We start from a two-player game, as follows:

▶ **Theorem 10.** *There is a normal-form Bayesian two-player game Γ, which has a welfare-maximizing full-information communication equilibrium d, such that no asynchronous protocol P implements d or any other welfare-maximizing communication equilibrium.*

(Recall that a *welfare-maximizing* equilibrium is one where the expected sum of the players' utilities is maximized.)

Theorem 10 implies that for any graph G that is not 2-connected, there is a normal-form Bayesian game Γ' such that no protocol can achieve maximum-welfare in Γ': since G is not 2-connected, it has a *bridge*, an edge (u, v) whose removal disconnects the graph. We take Γ' to be the game where players u, v take on the roles of the two players in Γ, and all other players have only one possible type and action. It is not hard to see that the impossibility result of Theorem 10 carries over to Γ'.

The proof of Theorem 10 is inspired by a game from [7, Ch. 4], and essentially involves a game of "chicken": the utility function incentivizes each player to learn the other player's type, but not to reveal its own true type. However, the only welfare-maximizing equilibrium is achieved when both players reveal their true types to one another. In any run of an asynchronous protocol, some player must be the first to reveal some information about its type, but neither player wants to "go first", because revealing this information gives an advantage to the other player, and decreases the revealing player's expected payoff. Therefore, there is no protocol that implements the welfare-maximizing equilibrium: players always have an incentive to deviate from the protocol. There is some subtlety involved in capturing what it means to "reveal information about the type" (for example, suppose player 1 sends its true type with probability 1/100, and otherwise sends a random type), and proving that this indeed gives the other player an advantage. See Appendix B for the details.

References

1 Ittai Abraham, Danny Dolev, Ivan Geffner, and Joseph Y. Halpern. Implementing mediators with asynchronous cheap talk. In *PODC 2019*, pages 501–510, 2019.

2 Ittai Abraham, Danny Dolev, Rica Gonen, and Joe Halpern. Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In *PODC 2006*, pages 53–62, 2006.

3 Ittai Abraham, Danny Dolev, and Joseph Y Halpern. Lower bounds on implementing robust and resilient mediators. In *TCC 2008*, pages 302–319, 2008.

4 Ittai Abraham, Danny Dolev, and Joseph Y. Halpern. Distributed protocols for leader election: A game-theoretic perspective. *ACM Trans. Economics and Comput.*, 7(1):4:1–4:26, 2019.

5 Yehuda Afek, Yehonatan Ginzberg, Shir Landau Feibish, and Moshe Sulamy. Distributed computing building blocks for rational agents. In *PODC 2014*, pages 406–415, 2014.

6 Yehuda Afek, Shaked Rafaeli, and Moshe Sulamy. The role of a-priori information in networks of rational agents. In *DISC 2018*, pages 5:1–5:18, 2018.

7 Mor Amitai. Cheap-talk with incomplete information on both sides. Discussion paper 90, Center for Rationality, The Hebrew University of Jerusalem, 1996.

8 Itai Ashlagi, Dov Monderer, and Moshe Tennenholtz. On the value of correlation. *J. Artif. Int. Res.*, 33(1):575–613, 2008.

9 Robert J. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96, 1974.

10 Elchanan Ben-Porath. Cheap talk in games with incomplete information. *J. Economic Theory*, 108(1):45–71, 2003.

11 Arkadev Chattopadhyay, Jaikumar Radhakrishnan, and Atri Rudra. Topology matters in communication. In *FOCS 2014*, pages 631–640, 2014.

12 Francoise Forges. An approach to communication equilibria. *Econometrica*, 54(6):1375–1385, 1986.

13 Joseph Halpern and Vanessa Teague. Rational secret sharing and multiparty computation. In *STOC 2004*, pages 623–632, 2004.

14 Joseph Y Halpern and Xavier Vilaça. Rational consensus. In *PODC 2016*, pages 137–146, 2016.

15 Itay Harel, Amit Jacob-Fanani, Moshe Sulamy, and Yehuda Afek. Consensus in Equilibrium: Can One Against All Decide Fairly? In *OPODIS 2019*, pages 20:1–20:17, 2019.

16 Juraj Hromkovič, Claus-Dieter Jeschke, and Burkhard Monien. Optimal algorithms for dissemination of information in some interconnection networks. *Algorithmica*, 10(1):24–40, 1993.

17 Sergei Izmalkov, Silvio Micali, and Matt Lepinski. Rational secure computation and ideal mechanism design. In *FOCS 2015*, pages 585–594, 2015.

18 David M Kreps and Robert Wilson. Sequential equilibria. *Econometrica: Journal of the Econometric Society*, pages 863–894, 1982.

19 Matt Lepinski, Silvio Micali, Chris Peikert, and Abhi Shelat. Completely fair sfe and coalition-safe cheap talk. In *PODC 2004*, pages 1–10, 2004.

20 Merav Parter and Eylon Yogev. Secure distributed computing made (nearly) optimal. In *PODC 2019*, pages 107–116, 2019.

21 Ryan M. Rogers and Aaron Roth. Asymptotically truthful equilibrium selection in large congestion games. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, EC '14, pages 771–782, 2014.

22 R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

23 Assaf Yifrach and Yishay Mansour. Fair leader election for rational agents in asynchronous rings and networks. In *PODC 2018*, pages 217–226, 2018.

A Missing Details from the Communication Lower Bound

We prove that any protocol implementing the equilibrium $(1, \ldots, 1)$ in the game from Section 7 must send $\Omega(n^2 \cdot b)$ bits in total (in expectation). In fact, this holds even if the protocol only achieves global output $(1, \ldots, 1)$ with probability $1 - \epsilon$, where $\epsilon \in (0, 1)$ is some constant error probability (which may exceed $1/2$).

Proof. Consider the family of cuts $\mathcal{S} = \{S_j\}_{j \in [n]}$, where S_j is the cut containing edges $\{j, (j+1) \bmod 2n\}$ and the "opposite edge" $\{(j+n) \bmod 2n, (j+n+1) \bmod 2n\}$. We argue that across each such cut, $\Omega(nb)$ bits of communication must flow.

Let A_j, B_j be the players on the two sides of the cut S_j, and let $\mathbf{M}_j$ be a random variable representing the messages that flow across the two edges in the cut S_j. Given a tuple X of players, let t_X represent the tuple consisting of the types of all players $i \in X$.

Consider a game $\mathcal{G}_j$ between two parties, Alice and Bob, where Alice receives the types $\mathbf{t}_{A_j}$ of all players on one side of the cut, and Bob receives the types $\mathbf{t}_{B_j}$ of all players on the other side of the cut. The goal is for Alice and Bob to each output the other party's input.

For information-theoretic reasons, in order for Alice to output $\mathbf{t}_{A_j}$ correctly with probability $1 - \epsilon$, Bob must send her $\Omega(n \cdot b)$ bits in expectation: the input of each player i is a uniformly random string of length $n \cdot b$, so the entropy of $\mathbf{t}_{B_j}$ is $\mathbb{H}(\mathbf{t}_{B_j}) = nb$. However, by Fano's inequality, conditioned on the communication $\mathbf{M}_j$ between Alice and Bob, we have

$$\mathbb{H}(\mathbf{t}_{B_j} | \mathbf{M}_j) \leq \mathbb{H}(\epsilon) + \epsilon \cdot \log(2^{nb} - 1) < 1 + \epsilon nb.$$

(Here, $\mathbb{H}(\epsilon) = -\epsilon \log(\epsilon) - (1-\epsilon)\log(1-\epsilon)$ is the binary entropy of $\epsilon \in (0, 1)$, which satisfies $\mathbb{H}(\epsilon) \in (0, 1)$.) For symmetric reasons, we also have $\mathbb{H}(\mathbf{t}_{A_j} | \mathbf{M}_j) \leq 1 + \epsilon nb$. This means that $\mathbb{I}(\mathbf{M}_j; \mathbf{t}) = \mathbb{H}(\mathbf{t}_{A_j} \mathbf{t}_{B_j}) - \mathbb{H}(\mathbf{t}_{A_j} \mathbf{t}_{B_j} | \mathbf{M}_j) = 2nb - 2\epsilon nb - 2 = \Omega(nb)$, assuming n is sufficiently large. Mutual information is symmetric: we can also write $\mathbb{I}(\mathbf{M}_j; \mathbf{t}) = \mathbb{H}(\mathbf{M}_j) - \mathbb{H}(\mathbf{M}_j | \mathbf{t}) \leq \mathbb{H}(\mathbf{M}_j)$, which implies that $\mathbb{H}(\mathbf{M}_j) \geq 2nb$. Entropy is never greater than the expected number of bits required to represent $\mathbf{M}_j$, and the claim follows.

Alice and Bob can *simulate* the protocol P to win the game $\mathcal{G}_j$: each party locally simulates the vertices on their side of the cut, and sends to the other party the messages crossing the cut edges. Under P, with probability at least $1 - \epsilon$, each vertex outputs the type of the opposite vertex, as this is the only scenario where the utilities of the players are $(1, \ldots, 1)$. Thus, with probability $1 - \epsilon$, Alice and Bob learn the correct output for $\mathcal{G}_j$ by simulating P. The communication between the two parties in this simulation is exactly $\mathbf{M}_j$.

Observe that each edge of the ring appears in exactly one cut in the family $\mathcal{S} = \{S_j\}_{j \in [n]}$. Thus, by linearity of expectation (summing over all cuts), the total communication that P sends on all edges is, in expectation, $\Omega(n^2 \cdot b)$. ◀

B Necessity of Two-Connectivity: Proof Overview

In this section we give a detailed overview of the proof of Theorem 10.

Consider the following 2-player Bayesian game Γ: there are two possible types, $\{1, 2\}$, and the type of each player is chosen uniformly and independently of the other player. The game has two possible actions, denoted $\{1, 2\}$ (the same as the set of types). The utilities are given by the following matrices, $\left\{u^{i,j} \;\middle|\; i, j \in \{1, 2\}\right\}$, where element (a_1, a_2) of matrix $u^{i,j}$ represents the utilities of the two players when their types are i, j (respectively) and the actions they take are a_1, a_2 (respectively).

$$u^{1,1} = \begin{pmatrix} 1,1 & 2,-2 \\ -2,2 & 0,0 \end{pmatrix},\ u^{1,2} = \begin{pmatrix} 0,0 & 0,0 \\ 1,1 & 0,0 \end{pmatrix},\ u^{2,1} = \begin{pmatrix} 0,0 & 1,1 \\ 0,0 & 0,0 \end{pmatrix},\ u^{2,2} = \begin{pmatrix} 0,0 & -2,2 \\ 2,-2 & 1,1 \end{pmatrix}.$$

It is not difficult to verify that the information-dissemination mediator d_{info} defined in Section 3 is a full-information communication equilibrium for Γ (i.e., it satisfies the conditions of Definition 3), and furthermore, it is the only communication equilibrium that achieves social welfare of 2 (the *social welfare* of an equilibrium with utilities $(u_1, \ldots, u_n)$ in an n-player game is the sum $\sum_{i=1}^{n} u_i$ of the players' utilities).

Now fix a protocol P which achieves utility $(1, 1)$ in all runs, and let us show that P *cannot* be an equilibrium.

While we have so far avoided giving the formal definitions associated with extensive-form games with imperfect information, here we cannot avoid them completely. However, to simplify matters, we consider a restricted set of executions, and give only the simplified definitions that are necessary to understand the proof. Throughout, we use bold-face letters to denote random variables, and plain letters to denote concrete values.

We consider only runs of P resulting from a scheduler that schedules the two players in alternating order, and immediately delivers every message that is sent (e.g., if player 1 sends a message to player 2, then in the next step the scheduler immediately schedules player 2 and delivers the message). We also assume w.l.o.g. that both players send a message every time they take a step, as any protocol that does not do this can be transformed into one that does, without affecting whether or not the protocol is an equilibrium (under this specific set of runs). The history h of such a run is represented by the sequence of messages sent and received by the two players; both players know the entire history, since they know what they sent and what they received. Moreover, since the types are initially independent, they remain independent conditioned on any history[3]. We emphasize that a *run* consists of the types of the two players and all the steps they have taken (messages sent and received), while a *history* consists only of the steps taken, as those are visible to both players.

After fixing the scheduler, for every assignment of types, the protocol P induces a probability distribution over histories of every given length. The *belief* of player 1 about player 2's type given the history h (and similarly for player 2) is the distribution $p_1(\cdot|t_1, h) : \{1, 2\} \to [0, 1]$, where

$$p_1(\mathbf{t}_2 = j|t_1, h) = \frac{\Pr[\mathbf{t}_2 = j|\mathbf{t}_1 = t_1, \mathbf{h} = h]}{\Pr[\mathbf{t}_2 = j|\mathbf{t}_1 = t_1]}.$$

The probability is taken with respect to the protocol's distribution over histories of length $|h|$ and the random assignment of types. However, since the player's types are independent, and they remain so conditioned on the history, we can omit t_1 from our notation and write $p_1(\cdot|h)$ (and for player 2, we omit t_2 and write $p_2(\cdot|h)$). By Bayes' law, we can also write

$$p_1(\mathbf{t}_2 = j|h) = \frac{\Pr[\mathbf{h} = h|\mathbf{t}_2 = j]}{2\Pr[\mathbf{h} = h]},$$

where the probability is again with respect to the protocol's distribution over histories of length $|h|$. In particular, $p_1(\mathbf{t}_2 = j|h) > 0$ iff there exists a run r where $\mathbf{t}_2 = j$ and the history is h.

[3] It is well-known that no communication protocol can create dependence between its inputs if they were initially independent.

Now we are ready to show that P cannot be an equilibrium. We begin by observing that since P is welfare-maximizing, every run must end with each player outputting the other player's type, to reach the maximum total payoff of 2. Thus, if P has a run with types (t_1, t_2) that ends after some history h, then

$$p_2(\mathbf{t}_1 = t_1|h) = 1, \qquad p_1(\mathbf{t}_2 = t_2|h) = 1.$$

Next, we show that at any history h where $p_2(\mathbf{t}_1 = a|h) > 0$ (that is, "player 2 still believes that $t_1 = a$ is possible"), player 1 can force player 2 to output a by deviating from the protocol (and vice-versa):

▶ **Observation 11.** *Let r be a run with a history h, and let $a \in \{1, 2\}$ be a type such that $p_2(\mathbf{t}_1 = a|h) > 0$. Then there is a strategy s_1 for player 1 starting from r, such that in every extension from r where player 2 plays according to P and player 1 plays s_1, player 2 always outputs a. The same holds with the roles of players 1 and 2 reversed.*

Proof. Since $p_2(\mathbf{t}_1 = a|h) > 0$, there exists a run r where $\mathbf{t}_1 = a$ and the history is h. In every extension of r where both players follow the protocol, player 2 eventually outputs a, as we assumed that the protocol always terminates with each player outputting the other player's type. Therefore, player 1's strategy s_1 is to simply behave the same way that it would under P when $\mathbf{t}_1 = a$, regardless of its true type, as this will always end with player 2 outputting a (if player 2 follows P). ◀

Now consider a run r of P, where the types of both players are 1. Observe that when a player *sends* a message, this does not change its belief; only receiving a message can change a player's belief about the other player's type.

The run begins with the prior $p_1(\mathbf{t}_2 = 1) = p_2(\mathbf{t}_1 = 1) = 1/2$, and it ends at a history h where $p_1(\mathbf{t}_2 = 1|h) = p_2(\mathbf{t}_1 = 1|h) = 1$. Let h be the *longest* history during r such that $p_1(\mathbf{t}_2 = 1|h) = p_2(\mathbf{t}_1 = 1|h) = 1/2$. Let h' be the history following h in r; then either $p_1(\mathbf{t}_2 = 1|h') \neq 1/2$ or $p_2(\mathbf{t}_1 = 1|h') \neq 1/2$, and we assume w.l.o.g. that $p_1(\mathbf{t}_2 = 1|h') > 1/2$. This implies that $p_2(\mathbf{t}_1 = 1|h') = 1/2$, because at each step, only one player's belief changes (the player that *receives* a message).

Since $p_2(\mathbf{t}_1 = 2|h') = 1 - p_2(\mathbf{t}_1 = 1|h') = 1/2$, by Observation 11, there is a strategy s_1 for player 1 from h' that always leads to player 2 outputting 2. Player 1's expected payoff is improved by following this strategy and outputting 1: under P, player 1 always receives a payoff of 1, but if player 1 follows s_1 and outputs 1, the expected payoff is

$$\begin{aligned} p_1(\mathbf{t}_2 = 1|h') \cdot u^{1,1}(1,2) + p_1(\mathbf{t}_2 = 2|h') \cdot u^{1,2}(1,2) \\ = p_1(\mathbf{t}_2 = 1|h') \cdot 2 + p_1(\mathbf{t}_2 = 2|h') \cdot 0 > \frac{1}{2} \cdot 2 = 1. \end{aligned}$$

Note that the strategy s_1 is not the strategy player 1 is supposed to follow under P, since playing according to P always ends with both players earning 1. Therefore, at h' player 1's rational choice is not to follow P, meaning P is not an equilibrium.