# Coordination Through Stochastic Channels

**Pierre Fraigniaud**
Inst. de Recherche en Informatique Fondamentale (IRIF), CNRS and Université Paris Cité, France

**Boaz Patt-Shamir**
School of Electrical Engineering, Tel Aviv University, Israel

**Sergio Rajsbaum**
Instituto de Matemáticas, UNAM, Mexico City, Mexico

## Abstract

We consider a *stochastic network* model consisting of a set of $n$ synchronous processes communicating by message passing. In each round, processes send messages directly to each other over a complete communication graph. The processes do not fail, but messages can be lost. Each message is delivered with probability $p$, for a given parameter $p \in [0, 1]$. We study the following optimization version of approximate agreement in this model. We assume that processes start with binary input values, execute an algorithm for a fixed number of rounds, and decide values in $[0, 1]$ satisfying the usual *validity* requirement stating that if all processes start with the same input value, then they should all decide that value. We propose deterministic algorithms that minimize the expected *discrepancy*, namely, the expected maximum distance between the decided values. We also present lower bounds on the expected discrepancy, which demonstrate the optimality of our algorithms for two processes. Finally, we present applications of our algorithms to solve randomized consensus and randomized approximate agreement.

**2012 ACM Subject Classification** Theory of computation → Probabilistic computation; Theory of computation → Distributed computing models; Theory of computation → Random network models; Theory of computation → Distributed algorithms

**Keywords and phrases** Approximate agreement, randomized consensus, stochastic models, topology

**Digital Object Identifier** 10.4230/LIPIcs.DISC.2025.32

**Funding** *Pierre Fraigniaud*: Additional support from ANR projects DUCAT (ANR-20-CE48-0006), ENEDISC (ANR-24-CE48-7768-01), and PREDICTIONS (ANR-23-CE48-0010), and from the InIDEX Project METALG.
*Boaz Patt-Shamir*: This research was supported by the Israel Science Foundation, grant No. 1948/21.
*Sergio Rajsbaum*: Part of this work was done while visiting IRIF, France, supported by ENEDISC (ANR-24-CE48-7768-01) and Ecole Polytechnique Gaspard Monge.

## 1 Introduction

We consider a set of $n$ synchronous processes communicating by sending messages to each other along the edges of a fully connected graph, i.e., for every two processes $i$ and $j$, there is a directed channel from $i$ to $j$, and a directed channel from $j$ to $i$. The processes do not fail, but the channels may fail to deliver messages. In each round, for every directed channel, the message sent through that channel is delivered with probability $p$, for a given parameter $p \in [0, 1]$.

This model has been called in the past *stochastic (dynamic) network*, and also sometimes referred to as evolving or temporal stochastic graph model as, in each round, the set of channels through which the messages have been delivered may define a different directed graph. Such a model has been studied since the 1980s, including variants where the probability of the directed graph at each round my depend on the graphs that occurred during the previous rounds, where the random graph at each round may be undirected, or where a


random graph distribution is assumed that guarantees that each instantaneous graph, or the union of a few consecutive graphs, is connected with high probability. We refer to the surveys and recent papers [5, 7, 11, 14, 16] for details and references. The stochastic network model is of theoretical interest (e.g., the directed Erdős-Réni random graph model can be viewed as a specific instance of the model), and also of practical interest, e.g. it has been argued that assuming that every link delivers a message independently with some probability $p$ in every round is actually a quite realistic model for uncorrelated transient channel or network interface failures in homogeneous system architectures, e.g., wireless and ad hoc networks [17, 20, 21].

Various problems have been considered before in a stochastic dynamic network, mainly related to broadcast, gossiping, connectivity and routing, as described in the above cited surveys. In this paper we consider the following problem, which can be viewed as an optimization variant of the classic approximate agreement problem [6]. We assume that processes start with binary input values, execute an algorithm for a fixed number of rounds, and decide values in $[0, 1]$ satisfying the standard *validity* requirement stating that (1) if all processes start with the same input value, then they should all decide that value, and (2) otherwise, each process can decide any value in $[0, 1]$. The *discrepancy* of the algorithm in an execution is the largest difference between any two decided values. Since executions are stochastic, the discrepancy is a random variable, and we seek algorithms with smallest *expected discrepancy* over all input assignments. For instance, in the case of two processes, and letting $q = p - 1$, the directed graph induced by the correct links (i.e., the channels through which the messages were delivered) is picked from $\mathcal{G} = \{\leftarrow\bullet, \rightarrow\bullet, \leftrightarrow\bullet, \circ\!\!-\!\!\bullet\}$, where the first two graphs are selected with probability $pq$ each, the third with probability $p^2$ and the fourth with probability $q^2$.

## 1.1 Our Results

We describe algorithms minimizing the expected discrepancy, showing several interesting, surprising behaviors.

- We first study the case of two processes in detail, proving tight upper and lower bounds on the expected discrepancy. In particular, we prove a lemma, referred to as the Integrality Lemma, stating that, for optimal algorithms, it is sufficient to assume that the processes always decide integral values, either 0 or 1. We present two algorithms, one that is optimal for $p \leq 1/2$, and one that is optimal for $p \geq 1/2$. Our lower bound technique shows that stochastic settings can be analyzed by following the algebraic topology approach although the latter was initially designed for analyzing protocols in non-stochastic settings [12].
- When one considers executions conditioned on the event that at least one message is delivered, it is intuitively clear that, when $p$ is large, i.e., when the probability of message delivery is high, it should be possible to obtain small expected discrepancy, namely, going down to 0 as $p$ gets closer to 1. We confirm this intuition with our algorithms. However, we additionally show that this is also the case when $q = 1 - p$ is large, i.e. when the probability of message loss is high. As $q$ gets closer to 1, the expected discrepancy of our algorithms also goes down to 0. In particular, when we discuss the application to randomized consensus, we will see that the probability of error goes down to 0, both when the probability of message loss is either very low or very high (conditioned on the event that at least one message is delivered).
- We then move on to the case of $n > 2$ processes. We show that the Integrality Lemma does not hold anymore, even for three processes. That is, an optimal algorithm must decide fractional values. As for the case of two processes, we show that there are thresholds for

the values of $p$ such that, depending on whether the value of $p$ is smaller or larger than each threshold, a different algorithm minimizes the expected discrepancy. As mentioned above, for two processes, there is a unique threshold $\theta = 1/2$, but we show that, for three processes, there are two thresholds $\theta_1 \approx 0.35$ and $\theta_2 = 1/2$. We provide optimal algorithms for three processes, one for each interval $[0, \theta_1]$, $[\theta_1, \theta_2]$, and $[\theta_2, 1]$ of message delivery probability $p$.

- For the case of a large number of processes $n$, we design a 1-round algorithm which guarantees an expected discrepancy that goes to 0 when $n$ grows to infinity, except for highly unbalanced input configurations, i.e., when the number of 0s is $\omega(\log n)$, or when the number of 1s is $\omega(\log n)$. We however show how to adapt our 1-round algorithm for handling all possible input configurations, at the mere expense of one additional round.
- Finally, to support our claim that minimizing the expected discrepancy could be useful in applications where processes must decide values within a small $\epsilon$ from each other in most executions, and tolerate few executions with large discrepancy – as for clock synchronization, sensor replication, among others (see, e.g. [9]) – we provide some concrete applications. Specifically, we design algorithms solving randomized binary consensus, and randomized approximate agreement, with small error probability. In fact, for two processes, our randomized approximate agreement algorithm also applies to the case of arbitrary inputs in $[0, 1]$, or even in $\mathbb{R}$, and we can also prove corresponding lower bounds.

Alon et al. [1] consider a model similar to ours, from the information theoretic perspective. They assume binary symmetric channels (BSC), i.e each message is a bit that may be flipped with some constant probability $\varepsilon > 0$. Their concern is computing a function (while ours is computing a task). Roughly, they show that for $n \to \infty$, any computation over failure-free channels can be emulated, with high probability, over BSC channels with a constant multiplicative overhead. These results are asymptotic and are not applicable to small $n$ values, but they inspired our Theorem 15.

### Organization

The model is presented in Section 2. Algorithms for the case of two processes are analyzed in Section 3, and corresponding lower bounds in Section 4, where the integrality lemma is proved. In Section 5 we present our results for the case of more than 2 processes. We discuss applications in Section 6. The conclusions are in Section 7. Some proofs and extensions (i.e. message delivery probabilities that may vary between different channels, and in different rounds, and variance analysis) are omitted from this extended abstract.

## 2 Model

The *stochastic dynamic network* model involves a set of $n \geq 1$ processes labeled from 1 to $n$. For each ordered pair $(i, j) \in [n] \times [n]$ of distinct integers, there is a directed channel from process $i$ to process $j$. Communication proceeds as a sequence of synchronous rounds. At each round $r \geq 1$, each of the $n$ processes can send one message of arbitrary size to each of the other processes. Every message may however fail to reach its destination: it only succeeds with some probability. Specifically, for each round $r \geq 1$, and for every message sent by process $i$ to process $j$,

$$\Pr[\text{the message sent by } i \text{ at round } r \text{ is received by } j] = p^r_{i,j} \ ,$$

where, for every $i, j \in [n]$ with $i \neq j$, and for every $r \geq 1$, $p^r_{i,j} \in [0, 1]$. When a message is not received, which occurs with probability $q^r_{i,j} = 1 - p^r_{i,j}$, it is lost. The sender of a message

is not informed of whether the message reached its destination or not. The probabilities $p^r_{i,j}$, for all $(i,j) \in [n] \times [n]$ and $r \geq 1$, are parameters of the model, and the $n$ processes are aware of their values. We concentrate on the *uniform* case where there exists $p \in [0,1]$ such that, for every $(i,j) \in [n] \times [n]$, and every $r \geq 1$, $p^r_{i,j} = p$, and discuss how to generalize our results in the full version. When there are just two processes, they are referred to as *players*, and called Alice and Bob.

All our algorithms are deterministic. Note however that the outputs of a deterministic algorithm in our stochastic model are random due to the probabilistic nature of the message delivery. We consider *one-shot algorithms*, i.e., algorithms that execute a fixed number of rounds, and then each process outputs a value.

We study the following optimization version of approximate agreement, that we refer to as the *Agreement Optimization* problem.

▶ **Definition 1** (Agreement Optimization). *Each process $i \in \{1, \ldots, n\}$ starts with a private input $x_i \in \{0,1\}$, and decides a value $y_i \in [0,1]$ subject to the following two conditions:*
**Termination:** *Every process must terminate in a finite number of rounds, $k$.*
**Validity:** *If all processes start with the same input value $x$ then all processes must decide $x$.*
*The objective is to minimize* discrepancy *defined as* $\max_{1 \leq i < j \leq n} |y_i - y_j|$.

In stochastic environments, the discrepancy is a random variable. We will concentrate on minimizing the *expected* discrepancy. In Section 6 we discuss how this implies minimizing probability of error in consensus and approximate agreement.

We will use several times the following consequence of the Validity condition.

▶ Remark 2. If processes have the same input value, then they all output that value in any execution. Furthermore, if some processes with input value $x$ receives the same input value $x$ from some set of processes, and nothing from other processes, then it has to decide $x$.

## 3 Two player algorithms

We assume in this section, without loss of generality, that Alice has input 0 and Bob has input 1. This assumption is solely for the purpose of presenting the algorithms; none of the two players initially know the input of the other player. We consider only initial configurations with different input values due to Remark 2.

We first consider single round algorithms in Section 3.1, and then extend our results to the case of multiple rounds, in Section 3.2.

### 3.1 Single round algorithms

In a single round, Alice and Bob send one message each, and hence there are four possible executions, according to whether each message was delivered or not. Remark 2 states that a player who did not receive a message must output its input value. So a single-round protocol can be completely characterized by the outputs made by the players as a response to the reception of a message containing a value different than their input (in all other cases, the output must be equal to the input).

Let us denote by $y_A \in [0,1]$ the output of Alice (with input 0) after receiving a message with value 1 from Bob, and similarly $y_B \in [0,1]$ denote the output of Bob (with input 1) after receiving value 0 from Alice. Each message is delivered with probability $p$, and dropped with probability $q = 1 - p$.

We consider two types of 1-round algorithms, which as we shall prove, are optimal, each one on its own range of $p$.

**Agreed Meeting Point $y$ (AMP$(y)$):** The algorithm specifies an arbitrary meeting point $y \in [0,1]$, and sets $y_A = y_B = y$. That is, if a player receives a value different than its input value, then it outputs $y$; otherwise it outputs its input value.

**Flip Value (FV):** The protocol is the following. If a player receives a value $x'$ different from its input value, then it outputs $x'$; otherwise it outputs its input value.

Figure 3(a) shows the expected discrepancy of the two algorithms as a function of $p$, as stated in the following theorem. If $p = q = 1/2$, either AMP or FV can be used, so long as both players use the same algorithm.

▶ **Theorem 3.** *The expected discrepancy of any single-round algorithm is at least* $\min\{q, p^2 + q^2\}$. *This discrepancy is achieved by the AMP algorithm whenever* $p > 1/2$, *and by the FV algorithm whenever* $p < 1/2$ *(and by both if* $p = 1/2$*).*

**Proof.** For any algorithm, let $\Delta \stackrel{\text{def}}{=} y_B - y_A$, the difference between the output values in an execution. Note that $-1 \le \Delta \le 1$. The adversary selects probabilistically in each round a directed graph representing the correct links from $\mathcal{G} = \{\circ\!\!\leftarrow\!\!\bullet, \circ\!\!\rightarrow\!\!\bullet, \circ\!\!\leftrightarrow\!\!\bullet, \circ\!\!-\!\!\bullet\}$, where the first two graphs are selected with probability $pq$ each, the second with probability $p^2$ and the fourth with probability $q^2$. Thus, the expected discrepancy is

$$\begin{aligned}\mathbf{E}[\mathsf{D}] &= (1-0)\cdot q^2 + (1-y_A)\cdot pq + (y_B - 0)\cdot pq + |y_B - y_A|\cdot p^2 \\ &= q^2 + pq(1+\Delta) + p^2|\Delta| \ . \end{aligned} \tag{1}$$

The expected discrepancy is thus a piecewise linear function of $\Delta$. In each "piece," i.e., in each interval of $\Delta$ values, $\mathbf{E}[\mathsf{D}]$ attains its smallest value at one of the endpoints of that interval, depending on the sign of $\frac{\mathrm{d}\mathbf{E}[\mathsf{D}]}{\mathrm{d}\Delta}$. We therefore now differentiate Eq. (1) with respect to $\Delta$ after conditioning on whether $\Delta$ is positive or not. We obtain:

$$\frac{\mathrm{d}\mathbf{E}[\mathsf{D}]}{\mathrm{d}\Delta} = \begin{cases} pq + p^2, & \text{if } 0 \le \Delta \le 1 \\ pq - p^2, & \text{if } -1 \le \Delta < 0 \end{cases} \tag{2}$$

We proceed by case analysis.

- If $\Delta \ge 0$, then $\frac{\mathrm{d}\mathbf{E}[\mathsf{D}]}{\mathrm{d}\Delta} > 0$, and therefore the minimum discrepancy is in the low end of the region $\Delta \ge 0$, i.e., at $\Delta = 0$, where $\mathbf{E}[\mathsf{D}] = q^2 + pq = q$ by Eq. (1).
- If $\Delta < 0$, there are two sub-cases, because, by Eq. (2), $\frac{\mathrm{d}\mathbf{E}[\mathsf{D}]}{\mathrm{d}\Delta}$ is positive if and only if $q > p$.
  - If indeed $q > p$, then the minimum discrepancy is obtained at the smallest possible value of $\Delta$, i.e., when $\Delta = -1$. In this case, by Eq. (1), $\mathbf{E}[\mathsf{D}] = q^2 + p^2$.
  - If $p > q$, then $\frac{\mathrm{d}\mathbf{E}[\mathsf{D}]}{\mathrm{d}\Delta} < 0$ and the minimum discrepancy is obtained at the high end of the region, i.e., at $\Delta = 0$, which we have already analyzed.

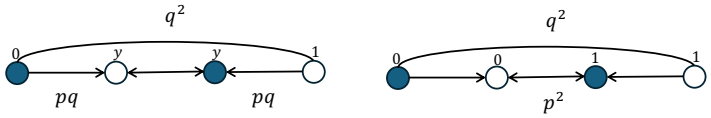
In summary, we have

$$\min \mathbf{E}[\mathsf{D}] = \begin{cases} q & \text{if } p > q, \text{ attained at } \Delta = 0 \\ q^2 + p^2 & \text{if } p < q, \text{ attained at } \Delta = -1 \end{cases} \tag{3}$$

In the case that $p = q$ we get that the minimum discrepancy is $q = q^2 + p^2 = 1/2$ for any $\Delta \in [-1, 1]$.

To see that these are the values achieved by the algorithms AMD and FV, consider Figure 1 (discussed in more detail in Section 4.1), where the four possible executions are depicted as edges with directions indicating that a message is delivered. A vertex corresponds

the the state of each one of the players at the end of the round. Black vertices correspond to Alice and white vertices to Bob, with their decisions on top of the corresponding vertex. For AMP, the undirected edge where no message arrives has discrepancy 1, while the two edges where exactly one message arrives have discrepancy $y$ and $1-y$, respectively. Thus, the expected discrepancy is $q^2+pqy+pq(1-y)$ which is equal to $q$, independently of the meeting point $y$. For FV, the bi-directed edge where both messages are delivered has discrepancy 1, while the two other directed edges have discrepancy 0, thus the expected discrepancy is $p^2+q^2$. ◀




**(a)** Algorithm AMP **(b)** Algorithm FV

**Figure 1** One round protocol complexes for two players starting with a single input assignment. Edges are oriented to indicate which messages arrived during the round, in each execution, and the probability of that execution. On the left are depicted the decisions of the AMP algorithm with meeting point $y$, on the right are the decisions of algorithm RV.

Assuming at least one message is always delivered, namely, the case when the adversary picks the actual communication graph in the set $\mathcal{G}' = \{\circ\!\!\not\leftarrow\!\!\bullet, \circ\!\!\rightarrow\!\!\bullet, \circ\!\!\not\leftrightarrow\!\!\bullet\}$ has been considered in the past, but for non-deterministic adversaries. For $p = 1/2$, our 1-round algorithms have expected discrepancy $1/3$, which is the optimal discrepancy when the adversary is non-deterministic e.g. [9, 13]. Remarkably, the expected discrepancy improves, both when the probability of message-loss is reduced, and when it is increased, bypassing the $1/3$ lower bound of the non-deterministic case.

We now consider the assumption that it is never the case that no messages arrive in the round. Remarkably, using our algorithms AMP and FV as before, we can see that the expected discrepancy conditioned on the event that at least one message is delivered is at most $1/3$ for any value of $p$, matching the worst case known bound for non-deterministic adversaries e.g. [9, 13], for two processes. Furthermore, the expected discrepancy improves more and more, both as $p$ gets smaller and smaller than $1/2$, and also as it gets larger than $1/2$, see Figure 2. Formally, we have the following.

▶ **Theorem 4.** *For any single-round algorithm, the expected discrepancy, conditioned on the event that at least one message is delivered, is at least* $\min\left\{\frac{p^2}{1-q^2}, \frac{pq}{1-q^2}\right\}$. *This discrepancy is achieved by the AMP algorithm if* $p > 1/2$, *and by the FV algorithm if* $p < 1/2$, *and by both if* $p = 1/2$, *at a maximum expected discrepancy value of* $1/3$.

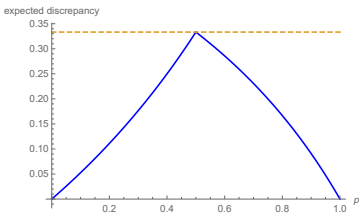



**Figure 2** The expected discrepancy of the optimal algorithm, conditioned on the event that not all messages are dropped. The dashed line indicates $1/3$.

## 3.2 Multiple rounds

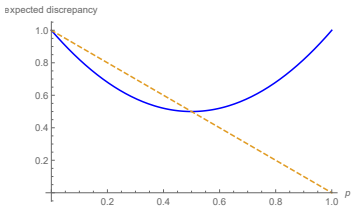
Consider now an algorithm running for $k$ rounds, for any fixed $k \in \mathbb{N}$, known by both processes. Given a single-round algorithm, one simple strategy is *replication*, i.e., sending the 1-round algorithm message $k$ times, so as to decrease the probability that the message is not received from $q$ to $q^k$. We shall see that the replication strategy is optimal only for $p \geq 1/2$. Instead, we use any of our single-round algorithms *recursively*. That is, we used the output of round $i \geq 1$ as the input of round $i+1$. For simplicity, we can force the inputs to always be in $\{0,1\}$, by using algorithm AMP with meeting point either at 0 or at 1 (algorithm FV satisfies the requirement in any case).

To analyze the expected discrepancy of multiple rounds, we note that the discrepancy of round $i+1$ is a random function (because the execution is stochastic) of a random variable, namely, the discrepancy of round $i$. The main part of the argument is stated as follows.

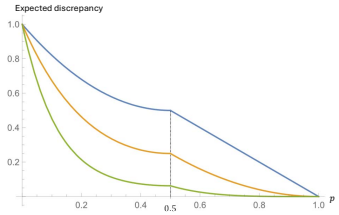
▶ **Lemma 5.** *Let $f_1, f_2, \ldots, f_r$ be a sequence of independently randomized functions, i.e., for every $1 \leq i \leq r$, and every $x \in \mathbb{R}$, $f_i(x)$ is a random variable. Suppose that, for every $i = 1, \ldots, r$, there exists $a_i \in \mathbb{R}$ such that $\mathbf{E}[f_i(x)] \leq a_i x$ for all $x \in \mathbb{R}$. Then $\mathbf{E}[f_r \circ \cdots \circ f_1(x)] \leq \left(\prod_{i=1}^r a_i\right)x$. Similarly, if, for every $1 \leq i \leq r$, $\mathbf{E}[f_i(x)] \geq a_i x$ for all $x \in \mathbb{R}$, then $\mathbf{E}[f_r \circ \cdots \circ f_1(x)] \geq \left(\prod_{i=1}^r a_i\right)x$.*

▶ **Theorem 6.** *The expected discrepancy of a $r$-round recursive algorithm is at most $q^r$ if $p \geq 1/2$ by using AMP, and at most $\left(p^2+q^2\right)^r$ if $p < 1/2$ by using FV.*

The result in Theorem 6 is illustrated in Figure 3(b).




**(a)** The expected discrepancy of the 1-round algorithms as a function of $p$. The solid line is for the FV rule, and the dashed line is for the AMP rule




**(b)** The expected discrepancy of recursively applying the optimal 1-round algorithm for 1 round (blue), 2 rounds (orange) and 4 rounds (green) as a function of $p$

**Figure 3** The expected discrepancy of the optimal algorithm.

A recursive algorithm for the case where it is assumed that at least one message is delivered in each round can be derived in a similar way, from Theorem 4 and Lemma 5.

▶ **Theorem 7.** *The expected discrepancy of a $r$-round recursive algorithm conditioned on the event that at least one message is delivered at each round is at most $\left(\frac{pq}{1-q^2}\right)^r$ if $p \geq 1/2$ by using AMP, and at most $\left(p^2/(1-q^2)\right)^r$ if $p < 1/2$ by using FV. Thus, the expected discrepancy is at most $1/3^r$ for any value of $p$.*

# 4 Lower Bound

In this section, we show that the simple two player algorithms consisting of recursively applying AMP or FV as defined in the previous section have the best possible expected discrepancy.

## 4.1 Protocol Complexes and Algorithms

We already discussed briefly the notion of protocol complex, only for one round, in Figure 1. We explain it in more detail and generalize it to multiple rounds in Section 4.1.1, and then define the notion of an algorithm for a protocol complex in Section 4.1.2.

### 4.1.1 The Weighted Protocol Complex

We consider the *protocol complex* representation of all executions after $r$ rounds used in the topology perspective of distributed computing [12]. In the case of two players, this *simplicial complex* is merely an undirected graph, but we stick to the terminology for consistency. Without loss of generality, we assume *full information protocols*, that is, every player sends its entire local state at each round. The protocol complex is essentially a representation of the local states of the players in all executions, together with a relation stating which executions are indistinguishable to a process. The local state of a player can be represented as a triple $(x_A, x_B, v)$, where

- $x_A$ and $x_B$ are the binary inputs of Alice and Bob, respectively, where a player sets the input of the other player only once it has received a message from the other, otherwise the value is set $\perp$, and
- $v$ is the *view* of the player, that is, its local state, including the sequence of messages it received (and marking those that it did not receive).

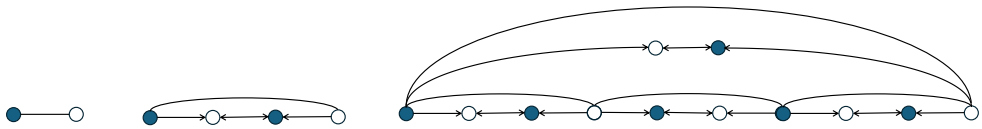
The initial state of Alice (resp., Bob) is thus $(x_A, \perp, v_0)$ (resp. $(\perp, x_B, v_0)$ where $x_A$ (resp., $x_B$) is the input of Alice (resp., Bob), and $v_0 = ()$ is an empty sequence. An initial configuration thus consists of two initial states, one for Alice and one for Bob. The protocol complex $\mathcal{P}_r$ after $r$ rounds then consists of a graph. Each vertex is a pair $(i, s)$, where $i \in \{A, B\}$ and $s$ is the local state of process $i$ after $r$ rounds, in some execution of the algorithm. Two states $(A, s_A)$, $(B, s_B)$ belong to an edge of the graph, if there is an $r$ round execution, where $s_A$ is the state of $A$, and $s_B$ is the state of $B$. Thus the graph is bipartite: each edge connects vertices of different players.

Notice that the protocol complex at round 0 with any *fixed* pair of input values $x_A, x_B$, consists of a single edge, whose endpoints are the initial states of $A$ and $B$ with those inputs. Figure 4(a) represent such an edge at round 0. The black vertex represents Alice in initial state $(x_A, \perp, v_0)$, and the white vertex represents Bob in initial state $(\perp, x_B, v_0)$. The protocol complex at round 1 depicted on Figure 4(b) is a path of three edges, plus an edge connecting the endpoints of the path. This latter edge depicted on top of the path corresponds to the execution of the (full information) protocol where no message arrives, denoted by $\circ\!\!-\!\!\bullet$. The path of three edges corresponds to the three scenarios $\bullet\!\!\rightarrow\!\!\circ$ (message from Alice reaches Bob), $\bullet\!\!\leftrightarrow\!\!\circ$ (both messages arrive), and $\bullet\!\!\leftarrow\!\!\circ$ (message from Bob reaches Alice). In general, the protocol complex at $r+1$ rounds is obtained from the one at $r$ rounds by replacing each edge by the round 1 protocol complex. Figure 4(c) represents the protocol complex after two rounds, starting from fixed inputs.

Note that, in each round,

$$\Pr(\bullet\!\!\leftrightarrow\!\!\circ) = p^2,\ \Pr(\bullet\!\!\rightarrow\!\!\circ) = \Pr(\bullet\!\!\leftarrow\!\!\circ) = p \cdot q, \text{ and } \Pr(\circ\!\!-\!\!\bullet) = q^2.$$

The probability of an edge in the complex corresponding to a multi-round execution is the product of the probabilities of all the successive scenarios leading to this edge. For instance, the probability of the top edge in Figure 4(c) is $q^4$ (all messages failed at both rounds), whereas the probability of the edge in the middle of the path at the bottom of Figure 4(c) is $p^4$ (all messages arrived during both rounds). More generally, the complex that corresponds



**(a)** Round 0 **(b)** Round 1 **(c)** Round 2

**Figure 4** Protocol complexes for two players starting with a single input assignment. Edges are oriented to indicate which messages arrived during the considered round, but the protocol complex is an undirected graph.

to the 0 round execution consists of two nodes and a single edge (see Figure 4(a)), which correspond to the two initial states, and the single possible 0-round (vacuous) execution, whose probability is 1. We denote this complex by $G_0$. Given a complex $G_i = (V_i, E_i)$, the complex $G_{i+1}$ is obtained by transforming each edge $e \in V_i$ to a 4-node, 4-edge "gadget" denoted $T(e)$, as shown in Figure 5 (recall that the protocol complex in an *undirected* graph).

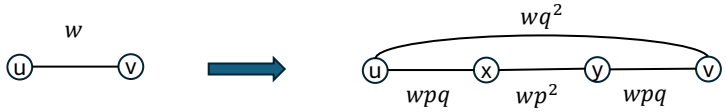



**Figure 5** Transformation of an edge by performing one more round. Edge labels indicate their weight.

In our complexes, the edges are weighted: the weight of an edge is the probability of the corresponding execution. Hence the sum of the edge weights must be 1. In particular, the weight of the single edge in $G_0$ is 1. Given an edge in $G_i$ with weight $w$, the edges of the gadget corresponding to that edge in $G_{i+1}$ have weights as shown in Figure 5. The rationale is that $T(e)$ corresponds to the executions starting in $e$ with one additional round, in which either both messages are dropped (the $(u, v)$ edge) with probability $q^2$, or both messages are delivered (the $(x, y)$ edge) with probability $p^2$, or just one message delivered (the edges $(u, x)$ and $(y, v)$) with probability $pq$.

### 4.1.2 Algorithm for a Protocol Complex

An $r$-round algorithm for approximate agreement is a deterministic *decision function* $\delta$ that assigns a real value $\delta(i, s)$ in $[0, 1]$ to each vertex $(i, s)$, $i \in \{A, B\}$, of the protocol complex at round $r$, where $s$ is the local state of player $i$. We denote the decision of process $i$ by $y_i \in [0, 1]$. Remark 2 (Validity) requires that $\delta(i, s) = x_i$, if no message has been received in $s$, or if a message has been received and the input of the other process is also $x_i$.

Notice that the decision of a player may depend on the name, $A$ or $B$, of that player. It is sometime more convenient to write $\delta_i(s)$ for the decision of player $i$ in state $s$. The algorithm is said to be *symmetric* if $\delta$ is solely a function of the local state $s$, i.e., if the decision functions of the players are equal, $\delta_A = \delta_B$. The *discrepancy* of the execution is $|\delta(A, s) - \delta(B, s)|$, and the expected discrepancy of the algorithm for this input configuration is taken over all possible $r$-round executions.

As discussed previously, one can assume, without loss of generality, that the input configuration is where $A$ has input 0, and $B$ has input 1.

## 4.2 Integrality of optimal algorithms

In Section 3.1 we analyzed single round algorithms, and showed that there always exists an optimal algorithm whose only possible output values are 0 and 1. In this section we consider the general case of two player, $r$-rounds algorithms, and show that the same integrality property holds as well. The idea is the following. For every given input assignment, the discrepancy of an approximate agreement algorithm is a random variable which we wish to minimize. We shall formalize the problem as an optimization problem, which we then convert into a linear program. The result will follow from the fact that the optimal solution is at a vertex of the polytope defined by the constraints.

▶ **Lemma 8** (Integrality Lemma). *There is an optimal algorithm with optimal expected discrepancy in which every player outputs only* 0 *or* 1*.*

**Proof.** Given an $r$-round algorithm, each possible execution $\Gamma$ (specifying which messages are delivered) has a probability $\Pr[\Gamma]$, which can be computed from the model parameters. Since the view at a process, along with its local input, determines the output value of that process, we know that given all inputs and local views, all outputs are determined, and therefore the discrepancy is determined as well. Let $\bar{x} = (x_A, x_B)$ denote the input assignment. There are 4 possible input assignments. If $x_A = x_B$, validity implies that both players output this value, and the discrepancy is 0. We hence focus on the case where $x_A \neq x_B$. Without loss of generality, we assume that $x_A = 0$ and $x_B = 1$. Given an algorithm, let us denote its outputs by $\bar{y} = (y_A, y_B)$. Let $\Gamma^*$ denote the set of all possible $r$-round executions for the given input. Given an execution $\Gamma \in \Gamma^*$, let $\Gamma[i]$ denote the view of process $i$ in $\Gamma$. Then an algorithm is optimal if its expected discrepancy for the given input has the same value as the solution to the following optimization problem.

$$\text{minimize } \mathbf{E}\left[\mathsf{D}_{\bar{y}}(\bar{x})\right] = \sum_{\Gamma \in \Gamma^*} \Pr[\Gamma] \cdot |y_A(x_A, \Gamma[A]) - y_B(x_B, \Gamma[B])| \tag{4}$$

subject to

$$\forall i \in \{A, B\} \text{ and } \forall \Gamma \in \Gamma^* \ : \min\{x \in \{x_i\} \cup \Gamma[i]\} \leq y_i(x_i, \Gamma[i]) \leq \max\{x \in \{x_i\} \cup \Gamma[i]\} \ . \tag{5}$$

(Abusing notation, we use $\Gamma[i]$ in Eq. (5) to also denote the set of values in the view.)

The constraints force the output values to be within the range spanned by their input value, and the values they received. We stress that the variables are $y_A$ and $y_B$. The values of $x_A$ and $x_B$ are fixed, and the probabilities are constants whose values are fixed by the stochastic environment model.

Observe that Eq. (4) and (5) do not specify a linear program, due to the absolute values in Eq. (4) – the min and max operators in Eq. (5) are applied to constants as far as the minimization of $\mathbf{E}\left[\mathsf{D}_{\bar{y}}(\bar{x})\right]$ is concerned as, for a given player $i \in \{A, B\}$ and a given execution $\Gamma \in \Gamma^*$, $x_i$ and $\Gamma[i]$ are constants, and so is the smallest (or largest) input value $x$ in $\{x_i\} \cup \Gamma[i]$.

We get around the non-linearity as follows. Consider an optimal solution $\bar{y}$ to the problem specified by Eq. (4) and (5). Let $Y$ denote the multiset of all possible $y_i$ values for the given input, one for each party and each execution. Now, let $z_1, \ldots, z_m$, for an appropriately large $m$, be the sequence of all values in $Y$ in *increasing order*. If we knew the order of the $z$ values, then we could have replaced the expression $|y_A(x_A, \Gamma[A]) - y_B(x_B, \Gamma[B]|$ with $z_\ell - z_s$, where $z_\ell$ and $z_s$ are the larger and the smaller values of $\{y_A(x_A, \Gamma[A]), y_B(x_B, \Gamma[B])\}$, respectively. To complete the transformation of the optimization problem defined by Eq. (4)

and (5) into a linear program, we make sure that the $z_i$ variables respect the assumed ordering by adding the constraints

$$z_i \leq z_{i+1} \quad \text{for all } 1 \leq i < m \ . \tag{6}$$

Now, the transformed target function Eq. (4), where a simple subtraction of the $z$ values replaces the absolute value expression in each term, and the constraints Eq. (5) and Eq. (6), where the original $y$ variables replace their $z$ pseudonyms, is a linear program. If the ordering according of the $z$ values agrees with an optimal solution, then the additional constraints of Eq. (6) do not change the target value. In other words, there exists an ordering of the $z$ values such that the optimal algorithm is a solution to the corresponding linear program.

Finally, as mentioned above, if both inputs are the same, the output equals the input value, and the claim is immediate. Otherwise, consider the polytope defined by Eq. (5) using the $z$ variables and Eq. (6). The vertices of this polytope are $\left\{0^\ell 1^{M-\ell} \mid 0 \leq \ell \leq M\right\}$, when the coordinates are ordered according to the $z$ ordering. Since there is always an optimal solution at a vertex, the result follows. ◀

▶ Remark 9. Note that an optimal solution may actually be attained at multiple vertices, in which case all their convex combinations (whose coordinates need not be 0 or 1) are also optimal solutions.

Lemma 8 narrows down the number of candidates for an optimal solution to a finite number of possible algorithms, so in principle one can enumerate all possible 0/1 output values for all inputs and executions to find an optimal solution. But this is hardly practical: The number of possibilities we need to enumerate for a $r$-round, $n$-player protocol, even in the case of symmetric algorithms, is doubly-exponential.

## 4.3 The Lower Bound

We have now all the ingredients to prove a lower bound on the expected discrepancy of any $r$-round protocol. To this end, fix an algorithm $\mathcal{A}$ and a number of rounds $r$. We focus on the instance where Alice gets input $x_A = 0$, and Bob gets input $x_B = 1$.

We study the protocol complex $\mathcal{P}$ of $\mathcal{A}$ after $r$ rounds. In this complex, each edge is associated with some probability (depending on the message deliveries in that execution), and a certain discrepancy (depending on the output values defined by the decision function $\delta$ of the protocol, in the local states at the edge endpoints). By Lemma 8 we may assume w.l.o.g. that the output values are only 0 and 1. We can therefore partition the vertices of the protocol complex into the set of vertices whose output is 0 and the set of vertices whose output is 1. All edges within a set correspond to executions with discrepancy 0, and all edges that connect vertices in different sets correspond to executions with discrepancy 1. The expected discrepancy is therefore simply the sum of the probabilities of the edges that connect the different sets. In graph theoretic terms, if we view the probability of an execution as the weight of the corresponding edge in the protocol complex, the expected discrepancy is the weight of the edges in the cut defined by the output values. Our goal of obtaining a lower bound on the expected discrepancy can be restated as bounding from below the weight of cuts in the protocol complex.

Let $s, t$ denote the two nodes in the 0-round protocol complex $G_0$ (see Figure 4(a)). These nodes correspond to the local states of $A$ and $B$, respectively, where no message is ever received. With these local states, by the validity requirement (Remark 2), $A$ and $B$ are forced to output their input values, namely 0 and 1, respectively. We are thus interested in bounding from below the weight of $s$-$t$ cuts in the $r$-round protocol complex $G_r$.

Let $G = (V, E, w)$ be a graph with edge weights, and let $s, t \in V$ be two distinct nodes in $G$. Recall that an $s$-$t$ cut of $G$, denoted $(S, \bar{S})$ is a partition of $V$ into two subsets $S$ and $\bar{S} = V \setminus S$ where $s \in S$ and $t \in \bar{S}$. The weight of $(S, \bar{S})$, denoted by $w(S; \bar{S})$, is the sum of weights of edges with one endpoint in $S$ and another in $\bar{S}$.

Recall from Section 4.1.1 that $T(e)$ is the protocol complex graph of one round executions starting in $e$.

▶ **Lemma 10.** *Let $e = \{u, v\}$ be an edge of weight $w$. Then the weight of the min-weight $u$-$v$ cut of $T(e)$ is $w \cdot \min\{q, p^2 + q^2\}$.*

**Proof.** Note that $\min\{q, p^2 + q^2\} = q$ if $p \geq 1/2$ and $\min\{q, p^2 + q^2\} = p^2 + q^2$ if $p \leq 1/2$. Also note that $\min\{q, p^2 + q^2\} < 1$ unless $p = 0$, that is, $\min\{q, p^2 + q^2\} < 1$ unless no message can ever be delivered. We refer to Figure 5 for an illustration. The edge $\{s, t\}$ in $T(e)$ must be in the cut, contributing $wq^2$ weight. Out of the three edge $\{u, x\}, \{x, y\}$ and $\{y, v\}$ we take an edge of minimal weight, i.e., $\{x, y\}$ if $p < q$ and otherwise either $\{u, x\}$ or $\{y, v\}$, adding a weight of $w \cdot \min\left\{p^2, pq\right\}$. In total, the cut weight is

$$w \cdot \left(q^2 + \min\left\{qp, p^2\right\}\right) \;=\; w \cdot \left(\min\left\{q^2 + qp, q^2 + p^2\right\}\right) \;=\; w \cdot \min\{q, p^2 + q^2\},$$

as claimed. ◀

▶ **Lemma 11.** *Let $i > 0$. If there exists an $s$-$t$ cut of weight $W$ in $G_i$ then there exists an $s$-$t$ cut in $G_{i-1}$ of weight at most $W/\min\{q, p^2 + q^2\}$, where $s$ and $t$ are the two nodes of $G_0$.*

**Proof.** Let $(S, \bar{S})$ be an $s$-$t$ cut of $G_i$. Let $S' = S \cap V_{i-1}$ and $\bar{S}' = \bar{S} \cap V_{i-1}$. Consider the cut $(S', \bar{S}')$ of $G_{i-1}$. This cut is well defined because $V_{i-1} \subset V_i$. For each edge $e' = \{u, v\}$ that crosses the cut $(S', \bar{S}')$ in $G_{i-1}$ we have that, in $G_{i-1}$ too, $u \in S$ and $v \in \bar{S}$. Therefore some of the edges of $T(e')$ cross the cut $(S, \bar{S})$ as well. By Lemma 10, min-weight cut that separates the endpoints of $e'$ in $T(e')$ have weight $\min\{q, p^2 + q^2\} \cdot w(e')$. Since the edges of different gadgets are disjoint, it follows that

$$w(S', \bar{S}') \;=\; \sum_{e' \in (S', \bar{S})} w(e') \;\leq\; \sum_{e \in (S, \bar{S})} w(e)/\min\{q, p^2 + q^2\} \;\leq\; w(S, \bar{S})/\min\{q, p^2 + q^2\},$$

and the result follows. ◀

We can now establish that our algorithms have optimal expected discrepancy.

▶ **Theorem 12.** *For every $r \geq 1$, every $r$-round algorithm has expected discrepancy at least $(\min\{q, p^2 + q^2\})^r$.*

**Proof.** It is sufficient to show that if $s, t$ denote the nodes of the protocol complex $G_0$, then the weight of any $s$-$t$ cut in the $r$-round protocol complex $G_r$ is at least $(\min\{q, p^2 + q^2\})^r$. We proceed by contradiction. If there was a cut of $G_r$ with weight $W < (\min\{q, p^2 + q^2\})^r$, then, by repeated application of Lemma 11, we would infer that there exists an $s$-$t$ cut of $G_0$ of weight smaller than $W/(\min\{q, p^2 + q^2\})^r < 1$, which is impossible, as the only $s$-$t$ cut in $G_0$ is $(\{s\}, \{t\})$, whose weight is 1 by definition. ◀

## 5 The case of more than two players

In this section we show that having more than 2 players changes the picture in a profound way. We first show that the integrality lemma does not hold, even for 3 players. We also show that when the number of players is large, 0 discrepancy can be achieved with overwhelming probability in two rounds, or in one round if the number of players with each value is slightly larger than a constant.

Consider a player with input $x$.

- $p \geq 1/2$: use AMP, with meeting point at $1/2$. I.e., if received $\neg x$ (once or twice), output $1/2$, otherwise output $x$.
- $\theta \leq p < 1/2$: if received $\neg x$ twice, output $\neg x$; if received $\neg x$ once, output $1/2$; otherwise output $x$.
- $p < \theta$: if received $x$ (once or twice) or received nothing, output $x$; otherwise (received no $x$ and at least one $\neg x$), output $\neg x$.

**Figure 6** Optimal protocol for three players, given $p \in [0, 1]$. The parameter $\theta \approx 0.35$ is the root of a certain fourth degree polynomial.

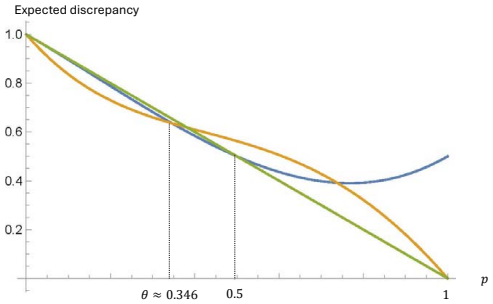



**Figure 7** The expected discrepancy of the strategies specified in Figure 6 as a function of $p$. The green curve is for large $p$ values, the orange for small $p$ values, and the blue for the middle range $\theta < p \leq 1/2$.

## 5.1 Integrality does not hold for more than 2 players

The linear program of Section 4.2 can be extended to any number of players as follows. While the constraints of Eq. (5) must always be maintained, the target function of Eq. (4) changes. In Eq. (4), we are minimizing the expected discrepancy of a single input assignment. This is fine for the two-party case, because there is only one non-trivial input in this case (namely $\bar{x} = (0, 1)$). However, this is no longer true for more than two players: For example, if there are 3 parties, there are two non-isomorphic nontrivial input assignments: $(0, 0, 1)$ and $(0, 1, 1)$. This means that our goal is to minimize the maximum of the discrepancies of the different inputs.

To deal with this, we introduce a new variable, say $v$ (the goal of the linear program is to minimize $v$), and a new constraint for each input assignment. Specifically, for each input assignment $\bar{x}$, let $\Gamma(\bar{x})$ denote the set of all executions with input assignment $\bar{x}$, and for any execution $\Gamma$, let $\mathsf{D}(\Gamma)$ be the discrepancy of the outputs in $\Gamma$. Then, for each input assignment $\bar{x}$, we introduce the constraint $v \geq \sum_{\Gamma \in \Gamma(\bar{x})} \Pr[\Gamma]\mathsf{D}(\Gamma)$ (similarly to the expression in the r.h.s. of Eq. (4).) Since the goal is to minimize $v$, the program will find the outputs which minimize the worst-case discrepancy, over all inputs.

However, the new constraints change the polytope and create vertices in non-integral coordinates. Indeed, we have computed the optimal 1-round for 3 players using the linear program formulation. The optimal protocol sometimes decides $1/2$: see Figure 6 for a specification of the optimal protocol according to the value of $p$. Interestingly, the output value $1/2$ is not used when $p$ is small. The expected discrepancy for the three strategies is plotted in Figure 7.

▶ **Theorem 13.** *The optimal expected discrepancy of a single-round algorithm for three players is achieved by the algorithm in Figure 6. The expected discrepancy, plotted in Figure 7, is*

$$\mathbf{E}[\mathsf{D}] = \begin{cases} q\ , & \text{if } p \geq 1/2 \\ \frac{1}{2}(2pq^2 + p^3 + pq(1-q^2) + p^2q^2) + q^2(1-p^2)\ , & \text{if } \theta \leq p < 1/2 \\ p(1-p^2) + q^2(1-pq)\ , & \text{if } 0 \leq p < \theta \end{cases}$$

*This discrepancy is achieved by the protocol of Figure 6.*

## 5.2 Asymptotic Analysis as the Number of Players Grows

We have seen that already for three players the optimal expected discrepancy behaves differently than for two players. At the extreme end of the spectrum we have the case of $n$ players, with $n \to \infty$. It turns out that 2 rounds suffice in this case to guarantee 0 discrepancy with high probability. In fact, 0 discrepancy can also be guaranteed w.h.p. in 1 round, if the minority is not too small.

Let us start with the case of one round. We use AMP with any meeting point. Note that the probability that a process does not receive a value decreases exponentially with the number of times that value is sent. Therefore, when one of the values (say 1) is held by a sublogarithmic number of players, there is non-negligible probability that at least one of the other players (with input 0) does not receive any 1 message. Formally, for any $n \geq 2$ and $0 \leq m \leq n$, let

$$I_{n.m} \stackrel{\text{def}}{=} \text{input instance of } n \text{ players, } m \text{ of them with input 0, } n-m \text{ with input 1.}$$

▶ **Theorem 14.** *The expected discrepancy of AMP($a$), satisfies* $\mathbf{E}[\mathsf{D}(I_{n.m})] \xrightarrow[n\to\infty]{} 0$*, for any meeting point* $a \in [0,1]$*, when* $\omega(\log_{1/q} n) \leq m \leq n - \omega(\log_{1/q} n)$*.*

**Proof.** To analyze the behavior of AMP, we use the following shorthand for $I_{n.m}$.

$$A(n,m) \stackrel{\text{def}}{=} \left(1 - q^{n-m}\right)^m\ .$$

Note that $A(n,m)$ is exactly the probability that in $I_{n.m}$, each of the $m$ 0-players received at least one 1 message (from the $n-m$ 1-players).

Note further that for constant $q < 1$, we have that $A(n,m) = \exp\left(-\Theta(mq^{n-m})\right)$. Therefore, for $n - m \geq \omega(\log_{1/q} n)$, i.e., $m \leq n - \omega(\log_{1/q} n)$ we have that $A(n,m) \xrightarrow[n\to\infty]{} 1$.

Now, for Algorithm AMP($a$), directly from definitions we have

$$\Pr[\mathsf{D}(I_{n.m}) = 0] = A(n,m)A(n,n-m) \tag{7}$$
$$\Pr[\mathsf{D}(I_{n.m}) = a] = (1 - A(n,m))A(n,n-m) \tag{8}$$
$$\Pr[\mathsf{D}(I_{n.m}) = 1-a] = A(n,m)(1 - A(n,n-m)) \tag{9}$$
$$\Pr[\mathsf{D}(I_{n.m}) = 1] = (1 - A(n,m))(1 - A(n,n-m)) \tag{10}$$

In particular, Eq. (7) implies the probability of consensus approaches 1 when $n$ grows and $m$ is neither too small nor too large. Regarding discrepancy, Eqs. 7–10 imply that the expected discrepancy of AMP($a$) is $\mathbf{E}[\mathsf{D}(I_{n.m})] = 1 - \Big(aA(n,m) + (1-a)A(n,n-m)\Big)$. ◀

To cover the case of small minority (of size $O(\log_{1/q} n)$) we "amplify" its presence with another round of communication. Specifically, we have the following algorithm, biased toward output 1.

Algorithm Boost1

1. Round 1: if the local input is 1, send it to all.
2. Round 2: if the local input or any value received in Round 1 is 1, send 1 to all.
3. If the local value is 1 or any value received is 1, decide 1. Otherwise decide 0.

▶ **Theorem 15.** *The expected discrepancy of Algorithm Boost1 approaches* 0 *as the number of players goes to infinity.*

**Proof.** We prove a stronger statement, namely that the probability that the discrepancy is 0 is $1 - \exp(-\Omega(n))$, where $n$ is the number of players. To see that, note first that if all inputs are 0 the statement is trivial: Boost1 does not send any message in this case, and all processes decide 0. So suppose, without loss of generality, that process 1 has input 1. We show that all other processes decide 1 after 2 rounds, regardless of their initial value. To this end, consider some process $i$. Process $i$ does not receive the 1 message if for every process $j$ (including $j = 1$ and $j = i$), either the first round message from process 1 to $j$ was dropped, or the send round message from $j$ to $i$ was dropped. Since this happens with probability at most $1 - p^2$ (considering also $j \in \{1, i\}$), we have

$$\Pr[\text{process } i \text{ does not hear "1"} \mid \text{ there exists a "1" input}] \leq (1-p^2)^n \ ,$$

and hence, by the Union Bound,

$$\Pr[\text{there exists a process that does not hear "1"} \mid \text{ there exists a "1" input}] \leq n(1-p^2)^n \ .$$

The discrepancy therefore satisfies $\mathbf{E}[\mathsf{D}] \leq 1 \cdot n(1-p^2)^n \xrightarrow[n\to\infty]{} 0$ for any constant $0 < p \leq 1$. ◀

## 6 Applications

In this section we present some examples about the use of agreement optimization to solve randomized consensus and approximate agreement.

### 6.1 Randomized consensus

Consensus in the case of $n = 2$ synchronous reliable processes when messages can be lost is called *coordinated attack* since [10] (also two generals' problem). The *randomized coordinated attack* [22], or for arbitrary number of processes, *randomized consensus*, see, e.g., [2, p.66], for a given desired upper bound error probability $0 < \rho < 1$, is defined by the following conditions, for processes starting with binary input values, and agreeing on binary output values.

Randomized Consensus

**Termination:** All processes terminate in a bounded number of rounds, $r$.

**Validity:** If all processes start with the same value, that is the only value that can be decided.

**Randomized agreement:** The probability that some process decides 0 and some other process decides 1 is at most $\rho$.

A straightforward application of Markov-like inequality gives the following result, for any number of players.

▶ **Theorem 16.** *Let $\mathcal{A}$ be an algorithm for agreement optimization with possible output discrepancy values $v_0 = 0 < v_1 < \ldots < 1$. If the expected discrepancy of $\mathcal{A}$ is $\mu$, then $\mathcal{A}$ solves randomized consensus with error probability $\rho \leq \frac{\mu}{v_1}$.*

Since our 2-party algorithms produce discrepancy either 0 or 1, we have the following result.

▶ **Corollary 17.** *For any given $r \in \mathbb{N}$, two-player randomized consensus can be solved in $r$ rounds with error probability at most $\min(p^2+q^2, q)^r$. Moreover, no protocol can solve randomized consensus in $r$ rounds with probability larger than $1 - \min(p^2+q^2, q)^r$.*

## 6.2 Approximate Agreement

The usual $\epsilon$-approximate agreement problem requires processes to decide values at most $\epsilon$ apart. Here we define a relaxed version, allowing an error of $\rho$. Formally, it requires the following to hold.

$(\epsilon, \rho)$-Approximate Agreement
**Termination:** All processes terminate in a bounded number of rounds, $r$.
**Validity:** The value decided by every processor is in the range spanned by all input values.
**Randomized $\epsilon$-Agreement:** The probability that the discrepancy is larger than $\epsilon$ is at most $\rho$.

In the classic version of approximate agreement [6], the input values are arbitrary real numbers. Input values in a bounded region such as $[0, 1]$, have also been considered e.g. [19]. The number of rounds $r$ depends on $\epsilon$. In fact, it may depend also on the discrepancy of the input values (the maximal distance between inputs) or even their magnitude, see e.g. [3]. The *relative discrepancy* agreement optimization can be defined as its discrepancy normalized by the discrepancy of the input values.

We have the following consequence of our lower bound on the expected discrepancy of agreement optimization. It is stated for inputs in $[0, 1]$ (it could be rephrased in terms of the discrepancy of the inputs)

▶ **Theorem 18.** *Any algorithm for two player $(\epsilon, \rho)$-Approximate Agreement that terminates in $r$ rounds when the inputs are binary must have probability of error of at least $\rho \geq \frac{(\min(p^2+q^2,q))^r - \epsilon}{1-\epsilon}$.*

We can also derive upper bounds on $(\epsilon, \rho)$-Approximate Agreement using agreement optimization, for the case of 2 players with arbitrary real input values, as follows.

First, note that non-binary inputs may be a problem for algorithms such as AMP: where should the meeting point be located? If we fix the location in advance, validity may be violated. And if a process computes it based on the values it has seen, the meeting point may be different at different processes because their views may be different. However, this is easily solvable in the case of two processes. The point is that the agreed meeting point is used only when two values are known, and in the case of two processes, if both their views contain two values, they must contain the same two values, so it is possible to specify a common meeting point.

Specifically, consider the following algorithm for $a \in [0, 1]$.

Algorithm Scaled AMP($a$) (2 players, arbitrary input values in $\mathbb{R}$)
- If no value is received, output the input value.
- Otherwise, denote the local input and the 4 value by $x$ and $y$. Output $\min\{x, y\} + a \cdot |y - x|$.

Clearly, scaled AMP is a generalization of AMP to any input values, assuming there are only two players. Since Algorithm FV also works with arbitrary input values, we arrive at the following result.

▶ **Theorem 19.** *Let $0 < \rho < 1$. Assume that $p > 1/2$. Then 2-player $(\epsilon, \rho)$-Approximate Agreement can be solved with probability $1 - \rho$ in $O(\log 1/\rho)$ rounds with relative discrepancy $\epsilon = q^{\Omega(\log(1/\rho))}$.*

We note that we have not attempted to optimize the constants. For the case of $p \leq 1/2$, we use the recursive FV algorithm and the following similar statement holds.

▶ **Theorem 20.** *Let $\rho > 0$. Assume that $p \leq 1/2$. Then 2-player $(\epsilon, \rho)$-Approximate Agreement can be solved in $O(\log(1/\rho))$ rounds with relative discrepancy $\epsilon = (p^2+q^2)^{\Omega(\ln(1/\rho))}$.*

The nice property of scaled AMP is that it allows us to use AMP$(a)$ with $0 < a < 1$ in the recursive version, thus obtaining a "continuous" algorithm, i.e., an algorithm in which the discrepancy shrinks (probabilistically) in each round, rather than algorithms in which the discrepancy is either 0 or 1.

▶ **Theorem 21.** *The optimal relative discrepancy of 2-party approximate agreement by a $r$-round algorithm is $\min(q, (p^2 + q^2))^r$, achieved by $r$ recursive applications of algorithm FV if $p \leq 1/2$ of scaled AMP$(a)$ (with any $0 \leq a \leq 1$) for $p \geq 1/2$.*

## 7 Conclusion

In this paper we have considered a stochastic dynamic network model consisting of synchronous, reliable processes communicating through channels that may drop messages with a given probability $p$. We defined the agreement optimization problem, where the goal is to minimize the expected discrepancy. For the case of two players, we provided a detailed characterization of the achievable expected discrepancy. To this end we developed the Integrality Lemma, showing that it is sufficient to consider algorithms with outputs in $\{0, 1\}$. Going beyond $n = 2$, we showed that the problem becomes much more complicated: this lemma no longer holds, and optimal algorithms may have more different behaviors depending on the parameter $p$, while in the case of two players, there are only two possible behaviors, for $p$ either smaller or larger than $1/2$. We also presented algorithms for large number of processes, whose expected discrepancy goes down to 0 exponentially fast. Finally, we showed the relevance of the agreement optimization problem to obtain algorithms for randomized consensus and randomized approximate agreement, with small probability of error.

We leave many interesting open problems for future work. A main question is to find optimal algorithms for agreement optimization for any number of processes $n > 2$. Our analysis of agreement optimization is mostly under the simplest stochastic assumption, with a single parameter $p$, known to all processes. The full version extends some of our results the case where the parameter may be different in different links or rounds, but there are many other stochastic models that have been considered in the past, as mentioned in the Introduction, for which agreement optimization has not been studied, e.g. a model where the links are unidirectional (and each one fails with probability $p$).

The analysis of our algorithms was performed by considering a *weighted* version of the protocol complex of the topology approach to distributed computing, which is, up to our knowledge, new. It would be interesting to extend it to $n > 2$ processes.

We have considered binary consensus and 1-dimensional approximate agreement. It would be interesting to consider multi-valued versions of consensus, e.g., [18], and multi-dimensional versions of approximate agreement [15]. Regarding other tasks, several have already been studied in dynamic networks, such as $k$-consensus [17] and set agreement [4, 8], but not in our stochastic setting.

## References


1 Noga Alon, Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Reliable communication over highly connected noisy networks. *Distributed Comput.*, 32(6):505–515, 2019. doi:10.1007/S00446-017-0303-5.

2 James Aspnes. Notes on theory of distributed systems, 2023. arXiv:2001.04235.

3 Hagit Attiya and Faith Ellen. The Step Complexity of Multidimensional Approximate Agreement. In Eshcar Hillel, Roberto Palmieri, and Etienne Rivière, editors, *26th International Conference on Principles of Distributed Systems (OPODIS 2022)*, volume 253 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:12, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.OPODIS.2022.6.

4 Martin Biely, Peter Robinson, Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. Gracefully degrading consensus and $k$-set agreement in directed dynamic networks. *Theor. Comput. Sci.*, 726:41–77, 2018. doi:10.1016/J.TCS.2018.02.019.

5 Michael Dinitz, Jeremy Fineman, Seth Gilbert, and Calvin Newport. Smoothed analysis of information spreading in dynamic networks. *J. ACM*, 71(3), June 2024. doi:10.1145/3661831.

6 Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33(3):499–516, May 1986. doi:10.1145/5925.5931.

7 Antoine El-Hayek, Monika Henzinger, and Stefan Schmid. Broadcast and Consensus in Stochastic Dynamic Networks with Byzantine Nodes and Adversarial Edges. In Dan Alistarh, editor, *38th International Symposium on Distributed Computing (DISC 2024)*, volume 319 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:15, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DISC.2024.21.

8 Pierre Fraigniaud, Minh Hang Nguyen, and Ami Paz. *A Simple Lower Bound for Set Agreement in Dynamic Networks*, pages 253–262. SIAM, 2025. 2025 Symposium on Simplicity in Algorithms (SOSA). doi:10.1137/1.9781611978315.20.

9 Matthias Függer, Thomas Nowak, and Manfred Schwarz. Tight bounds for asymptotic and approximate consensus. *J. ACM*, 68(6), October 2021. doi:10.1145/3485242.

10 Jim Gray. Notes on data base operating systems. In Michael J. Flynn, Jim Gray, Anita K. Jones, Klaus Lagally, Holger Opderbeck, Gerald J. Popek, Brian Randell, Jerome H. Saltzer, and Hans-Rüdiger Wiehle, editors, *Operating Systems, An Advanced Course*, volume 60 of *Lecture Notes in Computer Science*, pages 393–481. Springer, 1978. doi:10.1007/3-540-08755-9_9.

11 Sandra M. Hedetniemi, Stephen T. Hedetniemi, and Arthur L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988. doi:10.1002/net.3230180406.

12 Maurice Herlihy, Dmitry N. Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.

13 Gunnar Hoest and Nir Shavit. Toward a topological characterization of asynchronous complexity. *SIAM J. Comput.*, 36(2):457–497, 2006. doi:10.1137/S0097539701397412.

14 Fabian Kuhn and Rotem Oshman. Dynamic networks: models and algorithms. *SIGACT News*, 42(1):82–96, 2011. doi:10.1145/1959045.1959064.

15 Hammurabi Mendes, Maurice Herlihy, Nitin H. Vaidya, and Vijay K. Garg. Multidimensional agreement in byzantine systems. *Distributed Comput.*, 28(6):423–441, 2015. doi:10.1007/S00446-014-0240-5.

**16** Othon Michail and Paul G. Spirakis. Elements of the theory of dynamic networks. *Commun. ACM*, 61(2):72, January 2018. doi:10.1145/3156693.

**17** Henrique Moniz, Nuno Ferreira Neves, Miguel Correia, and Paulo Veríssimo. Randomization can be a healer: consensus with dynamic omission failures. *Distributed Computing*, 24(3):165–175, 2011. doi:10.1007/s00446-010-0116-2.

**18** Achour Mostéfaoui, Michel Raynal, and Frederic Tronel. From binary consensus to multivalued consensus in asynchronous message-passing systems. *Inf. Process. Lett.*, 73(5-6):207–212, 2000. doi:10.1016/S0020-0190(00)00027-2.

**19** E. Schenk. Faster approximate agreement with multi-writer registers. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 714–723, 1995. doi:10.1109/SFCS.1995.492673.

**20** Ulrich Schmid. Failure model coverage under transient link failures. Technical report, Research Report 2/2004, Technische Universität Wien, Institut für Technische Informatik, 2004, 2008. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=35b27320d6e1c34ef233f178411d20973d7813f5.

**21** Ulrich Schmid, Bettina Weiss, and Idit Keidar. Impossibility results and lower bounds for consensus under link failures. *SIAM Journal on Computing*, 38(5):1912–1951, 2009. doi:10.1137/S009753970443999X.

**22** George Varghese and Nancy A. Lynch. A tradeoff between safety and liveness for randomized coordinated attack. *Information and Computation*, 128(1):57–71, 1996. doi:10.1006/inco.1996.0063.