arXiv:2301.04817v2 [cs.DC] 8 Oct 2023

未知参与消息对手模型中的共识

GIULIANO LOSA,Stellar Development Foundation,
美国ELI GAFNI,加利福尼亚大学洛杉矶分校，美国

我们提出了一种新的分布式计算模型，其灵感来自比特币和以太坊等无需许可的分布式系统，使得能够在数学上规范的环境中研究无需许可共识。与 Pass 和 Shi 的休眠模型类似，我们考虑一个同步的、按轮次进行的消息传递系统，其中在线处理器集合每轮都会变化。不同于休眠模型，处理器集合可能是无限的。此外，处理器永远不会发生故障；相反，对手可以暂时或永久冒充某些处理器。最后，处理器可以使用一种强形式的消息认证，不仅能认证消息的发送者，还能认证消息发送的轮次。

假设对手每轮冒充的在线处理器少于一半，我们提出了两种共识算法。第一种算法在假设存在概率领导者选举预言机的情况下，保证确定性安全性，并使延迟期望为恒定值。第二种算法在假设存在不可撤销的冒充且参与最终稳定的情况下，保证确定性安全性和确定性活性。

该模型在完全一般的情况下并不现实。然而，若假设进程数量有限，且故障进程集合保持不变，该模型便与一个具有实际动机的模型一致：休眠模型的静态版本。

1 引言

无许可系统（如比特币和以太坊）的一个共同特点是它们具有动态可用性。这意味着，即使大量参与者意外离线或重新上线，只要任意时刻在线参与者中有足够比例（通常超过 1/2）行为正常，系统就能继续正确运行。

相比之下，传统的拜占庭容错（BFT）共识算法在这种情形下会失去活性，因为它们依赖固定规模的法定人数。借助工作量证明，比特币的中本共识成为首个动态可用共识算法，并且能够容忍 1/2 的故障比例（这是最优的 [14, 定理 6.2]）。然而，在中本共识中，对手总能走运，并以某种概率造成不一致，该概率取决于协议的安全参数。Pass 和 Shi 关于休眠模型的研究——该模型是动态可用系统的形式化模型——开启了一系列共识算法研究。这些算法与中本共识一样，能在故障比例为 1/2 时以概率保证一致性，但不使用工作量证明 [20, 8, 2, 7, 13]。

最近，Momose 和 Ren [19] 提出了一个问题：在动态可用环境中，能否确定性地保证安全性。随后，多个研究提出了在各种假设下具有确定性安全性的共识算法 [16, 21, 22, 17], ，但确定性地能够实现什么的确切边界仍不清楚。

本文提出在一种称为迭代无限认证拜占庭模型（IIAB）的新模型中研究具有确定性安全性的动态可用共识。该模型并不现实，但适合我们的目标：在数学上规整的模型中研究共识。举例来说，我们所说的规整性是指，在 IIAB 模型中，实现具有确定性安全性的共识是否可行，并不取决于受故障影响的处理器集合是保持不变、单调增长，还是不断波动。

“迭代”一词，如迭代即时快照（IIS） [5], 中的“迭代”所示，表示重复进行同步的通信封闭轮次。事实上，IIAB 模型是一种同步、逐轮进行的

作者地址：Giuliano Losa，giuliano@stellar.org，Stellar Development Foundation，美国；Eli Gafni，eli@ucla.edu，加利福尼亚大学洛杉矶分校，美国。

计算模型，其中一个可能无限大的处理器集合通过点对点链路相连，构成完全图；每一轮中，所有已发送的消息都会在该轮结束前收到。

然而，每一轮 r，对手都会将处理器划分为一个离线处理器集合（这些处理器不发送消息）和一个有限且非空的在线处理器集合 O_r。这与休眠模型类似，只是我们的情形中处理器集合可能是无限的。

更令人惊讶的是，IIAB 模型中的处理器永远不会发生故障；相反，对手会以下述方式控制它们的通信链路。每一轮，对手都会将在线处理器集合 O_r 划分为一个非空的行为良好的处理器集合 W_r 和一个被冒充的处理器集合 F_r，其中包含 $|F_r| < |W_r|$，由对手冒充。所有处理器都会忠实地遵循其算法，永不崩溃，但每一轮，对手都可以代表被冒充的处理器伪造任意消息。

处理器还可以使用一种强形式的消息认证，使任何处理器都能检查某条给定消息是否由另一指定处理器在某一特定轮次发送。请注意，这种强形式的认证并不现实。例如，在采用公钥签名方案和公钥基础设施（PKI）的系统中，一旦对手获得私钥 k，对手就能使用 k 签署消息，并声称该消息是在对手获得密钥之前的某一轮发送的。在 IIAB 模型中，这是不可能的。

IIAB 模型或许看起来不切实际，但它涵盖了以往无许可系统研究中使用的一些模型。例如，在进程集合有限，且参与是动态的、被冒充的处理器集合固定的假设下，Pass 和 Shi [20]的休眠模型静态版本与 IIAB 模型相同。如果在线处理器和被冒充的处理器都固定，且处理器还知道这些集合，那么我们得到的就是传统的同步认证拜占庭协议模型。因此，在 IIAB 模型中解决共识，也就能在那些模型中解决共识。

在少数派腐化假设下，于 IIAB 模型中解决共识并非易事。针对带有消息认证的同步系统，解决共识有两种著名方法：Dolev-Strong 算法 [9] 和基于仲裁组的方法，例如阶段之王算法 [3]。遗憾的是，这些方法似乎都不适用于 IIAB 模型。

首先，我们关于参与的假设使得任何行为良好的处理器都可能至多只在线一轮。因此，处理器保留本地状态毫无用处。这就排除了使用 Dolev-Strong 算法 [9]，因为该算法关键依赖本地状态。

其次，由于故障比例为 1/2，我们只能依赖多数法定人数（不能更多），而且看来多数法定人数无法在 IIAB 模型中阻止意见分歧：在给定轮次中，即使假设每个处理器都广播唯一消息，也可能出现这种情况：在线且行为良好的处理器 p 从其所听闻的处理器中严格多数的处理器那里收到消息 m，而另一个在线且行为良好的处理器 p' 则从其所听闻的处理器中严格多数的处理器那里收到消息 $m' \neq m$。例如：

示例 1. 考虑某一轮次 r，其中在线处理器为 $\{p_1, p_2, p_3\}$，并假设 $F_r = \{p_1\}$ 被冒充，而 $W_r = \{p_2, p_3\}$ 行为良好。假设 p_2 广播 v，而 p_3 广播 $v' \neq v$。如果对手代表 p_1 向 p_2 发送 v，并代表 p_1 向 p_3 发送 v'，那么 p_2 会从严格多数的处理器那里收到 v，而 p_3 会从严格多数的处理器那里收到 v'。

1.1 贡献总结

本文的主要贡献如下。

(1) 我们提出了 IIAB 模型。IIAB 模型是一个抽象模型，灵感源自参与度不断变化的无许可系统，使得人们能够在数学上规范的环境中研究共识。IIAB 模型是 Pass 和 Shi 的休眠模型静态版本的一种推广 [20], ；在进程集合有限，且参与度动态变化而故障为静态的假设下，该模型与 IIAB 模型一致。(2) 与共识问题通常一样，主要难点在于解决提交-采用任务 [11] （具有两个等级的分级一致 [6] ）。在第 4.2 节中，我们提出了一种对 Gafni 和 Zikas 算法进行推广的提交 - 采用算法 [12]。(3) 我们提出了一种依赖概率领导者选举预言机的概率共识算法。在对该预言机作出合理假设的条件下（在静态休眠模型中可使用可验证随机函数 [18] 实现），该算法的期望终止时间为 20 轮。(4) 我们提出了一种确定性共识算法，假设参与度最终趋于稳定，即从某个未知轮次 R之后保持不变，并且对手不断增长（即，对于每一轮 $r, F_r \subseteq F_{r+1}$）。该算法在第 R 轮之后经过 $O(|O_R|)$ 轮终止。

在补充材料中[15],我们还提供了算法的 TLA+ 规范，以及在 Isabelle/HOL 中对关键引理进行的机械化检查证明，该引理是提交-采纳算法正确性的基础。

1.2 技术概要

为了在 IIAB 模型中解决共识问题，我们分两步进行。首先，我们模拟一种更易于处理的模型，称为无 equivocation 的 IIAB 模型（简称无矛盾模型）；然后，我们介绍无矛盾模型中的共识算法。

与 IIAB 模型相比，无矛盾模型对对手的行为施加了额外限制：在任意给定轮次中，对手在其发送消息的所有信道上都只能发送相同的消息（不能通过向不同处理器发送不同消息来制造矛盾）；此外，如果对手向处理器 p 发送消息，但没有向处理器 p_1发送消息，那么对手必须从 p向 p_2发送一条特殊的故障通知，记作 λ。

关键的是，无矛盾模型的性质意味着，在同一轮次中，两个处理器不再可能各自见证两个相互冲突的严格多数，因此更容易设计基于多数法定人数的算法。

我们在第 2.4 节介绍无矛盾模型，并在第 3 节给出一个在 IIAB 模型中模拟无矛盾模型的模拟算法。

接下来，在第 4 节，我们将关注如何在无矛盾模型中解决共识问题。我们遵循第 4.1 节介绍的经典模式，即执行协调阶段与提交-采用 [11] 阶段交替组成的序列。图 1 展示了这一模式；它通过分离不同关注点，让共识算法更易于设计。每个协调阶段都必须在条件有利时促使处理器达成一致（例如，所有处理器恰好都同意一位合适的领导者，或系统以某种方式趋于稳定时），但在其他情况下，即使未能促使处理器达成一致，也可以接受。每个提交-采用阶段都必须检测是否已经存在一致性（若有），并作出共识决策。

在第 4.2 节中，我们介绍一种只需 2 个无 equivocation 轮次的提交-采纳算法。该算法将 Gafni 和 Zikas [12] 的算法推广到了动态参与的情形。

在第 4.3 节中，我们介绍了一种概率协调器，其保证达成一致性的概率为 1/2。该协调器依赖一种概率领导者选举预言机，该预言机在每一轮中以

概率为 1/2 向所有行为正常的处理器提供一名唯一行为正常领导者的身份。在静态休眠模型中，可以使用可验证随机函数（VRF）[18]实现该预言机。借助该概率协调器，我们得到了一种共识算法，其期望终止轮数为 20 轮。

最后，在第 4.4 节中，我们介绍一种受 Dolev-Strong 算法启发、适用于恒定参与情形的确定性协调器。该协调算法以自然数 $N > 0$ 为参数，运行 $N + 1$ 轮；如果参与人数保持恒定，且在前 N轮中被冒充的参与者少于 N 个（即 $\left|\bigcup_{0<r\leq N} F_r\right| < N$），则可保证一致性。

利用这一确定性协调器，我们得到一种共识算法，在增长型对手下可保证确定性安全性和确定性活性（即，对于每一轮 $r, F_r \subseteq F_{r+1}$），前提是存在一个未知轮次 R，称为稳定轮次，此后参与人数保持恒定。在这种情况下，算法会在第 R 轮之后的 $O(|O_R|)$ 轮内终止。

1.3 本文的前一版本

与本文的前一版本相比，我们澄清了内容的阐述，简化了提交-采纳算法的证明，并修正了若干错误：

- 我们删除了关于 IIAB 模型包含具有动态腐化的 Sleepy 模型这一错误说法（IIAB 模型仅包含静态休眠模型）。
- 我们修正了第 4.4 节中的确定性共识算法，并修正了该算法能够保证确定性终止性的条件。

2 IIAB 模型与无等价类 IIAB 模型

2.1 IIAB 模型

我们考虑一个可能无限的处理器集合 P，它们在一个同步的消息传递系统中运行分布式算法，系统中的每对处理器之间都通过点对点链路连接。

2.1.1 在线/离线集与行为正常/被冒充集。处理器执行一系列同步通信轮次，编号为 1、2、3、……。在每一轮 r 中，一个非空的有限处理器集合 O_r 在线，而集合 $\mathrm{P} \setminus O_r$ 离线；此外，在线处理器集合 O_r 被划分为一个非空的行为正常处理器集合W_r和一个被冒充的处理器集合F_r，且在线的被冒充处理器数量少于在线的行为正常处理器数量（即 $|F_r| < |W_r|$）。集合 O_r、W_r 和 F_r 对所有轮次 r 都保持不变，但处理器不知道这些集合。

我们将被冒充视为由对手实施。若对于每个轮次 r 都有 $F_r \subseteq F_{r+1}$，则称该对手在增长。

2.1.2 处理器与对手行为。在操作层面，在轮次 $r = 1$中，每个处理器 p（无论在线还是离线）首先接收一个外部输入 in_p。随后，每个轮次 r由发送阶段和接收阶段组成。在发送阶段，每个在线处理器按照算法的规定，根据其在上一轮次收到的消息以及它向某些预言机发出的查询所返回的结果（如果 $r > 1$），或者仅根据其外部输入（如果 $r = 1$），发送一组消息。然而，对手会删除在线且被冒充的处理器发出的所有出站消息（集合F_r），并在被冒充处理器的出站通信链路中注入任意消息，但须遵守以下限制。最后，离线处理器不发送消息。

在每一轮 r 的接收阶段，每个处理器（无论是否在线）都会收到同一轮 r 发送阶段发送给它的所有消息。此外，每个处理器还可以按照算法的规定，选择性地产生不可撤销输出；该输出取决于它在该轮收到的消息集合，以及它可能调用的任何预言机调用的输出。

2.1.3 消息认证。消息可以是任意内容，但签名消息这一类型受到特殊限制。签名消息的形式为 p，r，m，其中 p 是该签名消息的签名者，r 是该签名消息所属的轮次，m 是该签名消息的消息内容。此外，在每一轮 r 中，只有当 $r\prime = r$，且该消息冒充了轮次 r 中消息的签名者 p 时，对手才能发送签名消息 p，$r\prime\, m$；除非 p，$r\prime\, m$ 已在先前轮次发送。形式化地说，如果签名消息 p，r，m 在第 r' 轮出现在通信链路（p_1，p_2）上（无论是单独出现还是包含在另一条消息中），那么：

- 如果 $p_1 \in W_{r\prime}$，那么要么 $p = p_1$ 和 $r\prime = r$，要么 $r < r\prime$ 且 p_1 在轮次 $r\prime\prime < r\prime$ 收到了 p, r, m（直接收到或作为另一条消息的一部分收到）。
- 如果 $p_1 \in F_{r\prime}$，那么要么 $p \in F_r$ 和 $r\prime = r$，要么 $r < r\prime$ 且签名消息 p, r, m 在轮次 $r\prime\prime < r\prime$ 通过通信链路发送。

上述消息限制的一个结果是，如果对手在轮次 r 冒充处理器 p，但未在轮次 $r\prime \neq r$ 冒充，那么对手就无法可信地声称 p 在轮次 r' 发送了消息 m，而实际上 p 并未发送该消息。这与基于公钥认证的系统形成对比：在处理器 p 的私钥遭到泄露后，对手便可以可信地声称 p 过去发送过消息，尽管事实上 p 并未发送。

2.1.4 领导者选举预言机。在指定情况下，处理器可以使用概率型领导者选举预言机，该预言机为每个处理器提供一个本地接口。

在每一轮 r，每个处理器 p 都可以查询领导者选举预言机。预言机返回某个处理器 l_p^r 的身份，该处理器称为领导者，并保证以概率为1/2，存在一个行为良好的处理器 l_r，使得每个处理器 p 获得的领导者都相同 $l_p^r = l_r$。

领导者选举预言机是对使用可验证随机函数 [18]（VRF）选举领导者的一种抽象。可验证随机函数可用于从有限的处理器集合中选出领导者，方法是选取可验证随机函数输出最大的处理器；即使少数处理器有选择地向其他处理器公开其输出，所有行为正常的处理器仍能以概率为1/2就同一领导者达成一致。

2.1.5 算法组合。在后续章节中，我们将通过依次组合子算法来构造算法。在这种情况下，所有子算法都使用固定数量的轮次，因此组合只需按顺序执行这些算法，并将一个算法的输出作为下一个算法的输入。

2.1.6 有用术语。我们称 p 在第 r 轮听到 q，当且仅当 p 在第 r 轮的接收阶段通过链路 (p, q) 收到某条消息。我们称 p 在第 r 轮收到来自严格多数的消息 m，当且仅当 p 收到来自其在第 r 轮听到的处理器中严格多于一半的处理器的消息 m。

2.1.7 关于严格多数的说明。需要注意，处理器集合 P 可能是处理器 p 在第 r 轮听到的处理器中的严格多数，但 P 不一定是在线处理器的严格多数。

示例 2。取 *4* 个处理器 p_1 到 p_4，其中 $W_r = \{p_1、p_2、p_3\}$，以及 $F_r = \{p_4\}$（因此 $O_r = \{p_1、p_2、p_3、p_4\}$）。假设 p_1 只在第 r 轮听到 $\{p_1、p_2、p_3\}$。那么，$P = \{p_1、p_2\}$ 是 p_1 在第 r 轮听到的处理器中的严格多数，但不是在线处理器 $\{p_1、p_2、p_3、p_4\}$ 的严格多数。

同样，正如引言中的示例 1 所示，即使假设每个处理器在某一轮 r 中广播一条唯一的消息，也可能出现两个处理器各自从严格多数的处理器那里听到两条不同消息的情况。不过，我们可以依赖一个有用的性质：如果 p 在第 r 轮从严格多数的处理器那里收到消息 m，那么至少有一个行为良好的处理器在第 r 轮发送了 m。

2.2 与其他模型的关系

如果假设处理器数量有限，那么 IIAB 模型的特殊情形对应于现有模型：

(1) 如果 F_r 保持不变，那么我们得到 Pass 和 Shi 的 Sleepy 模型的静态版本 [20]。
(2) 如果我们假设 W_r 和 F_r 都保持不变（也就是说，$F_r = F_1$、$W_r = W_1$，因此对每个 r，$O_r = O_1$），并且处理器已知 W_r 和 F_r，那么我们得到传统的同步认证拜占庭协议模型。
(3) 如果 O_r 保持不变，$|F_r| \leq 1$，并且对手仅限于丢弃消息（而不能伪造消息），那么我们得到 Santoro 和 Widmayer 的模型 [23]

请注意，具有动态腐化的 Sleepy 模型在 IIAB 模型中没有对应模型。这是因为我们对签名消息施加的消息限制无法通过公钥密码学实现：一旦对手获得密钥，就可以伪造消息，并在不被察觉的情况下声称这些消息是在之前的轮次中发送的。我们的消息限制可以防止这种情况。

2.3 IIAB 模型中的共识可解性

Santoro 和 Widmayer [23] 证明，即使我们将对手限制为：a）每轮只能冒充一个处理器；b）只能丢弃消息而不能伪造消息，确定性共识（同时满足确定性安全性和确定性活性）仍然不可能。因此，在没有额外假设的情况下，IIAB 模型中的确定性共识是不可能的。

2.4 无 equivocation 的 IIAB 模型

现在，我们对 IIAB 模型施加额外限制，得到我们所称的无 equivocation 的 IIAB 模型（简称无矛盾模型）。

首先，无矛盾模型要求处理器每轮只能广播一条唯一消息（而在 IIAB 模型中，处理器可以自由地通过各自的输出信道发送任意数量的消息）。

其次，无矛盾模型对对手施加了额外限制：每轮 r，对于每个被冒充的处理器 p，对手选择一条消息 m，并通过其部分输出信道（也可能一个都不通过）发送 m；此外，如果对手将 m 从 p 发送给某些处理器而不发送给其他处理器，那么对手必须向所有未收到 m的处理器发送特殊故障通知 λ；最后，对手还可以通过其部分输出信道发送 λ，并在其他信道上保持沉默。

因此，对于每一轮 r和每个处理器 p，以下所有情况都有可能发生：
(1) 没有处理器收到来自 p 的消息（当 p 离线时可能发生，或者 $p \in F_r$ 而对手刚好丢弃了 p的所有消息）；或者 (2) 所有处理器都收到同一条消息 m，即 p 实际发送的消息（当 p 在线且行为正常，或者对手的行为仿佛 p 行为正常时可能发生）；或者 (3) 存在一条消息 m（可能不同于 p 实际广播的消息），使得所有处理器都收到来自 p的消息，但一些处理器收到 m，另一些处理器则收到 λ（当 p 被冒充时可能发生）；或者 (4) 一些处理器收到 λ，而另一些处理器完全没有收到来自 p 的消息（当 p 被冒充时可能发生）。

3 模拟无歧义模型

在本节中，我们将展示如何在 IIAB 模型中实现第 2.4 节定义的无矛盾模型。

算法 1（无矛盾模型的模拟）。 我们使用 *IIAB* 模型中的两个连续轮次来模拟无矛盾模型中的一轮。

(a) 在第一轮中，每个在线处理器 p 执行以下操作：对于 p 必须模拟广播的每条消息 m，p 广播签名消息 $\langle p, 1, m\rangle$。*(b)* 在第二轮中，对于每个进程 p' 以及从 p' 收到的每条签名消息 $\langle p', 1, m\rangle$，每个在线处理器 p 都会将 $\langle p', 1, m\rangle$ 转发给所有人（从而转告 p' 在第 *1* 轮发送了 m），方法是广播 $\langle p, 2, \langle p', 1, m\rangle\rangle$。我们称 $\langle p, 2, \langle p', 1, m\rangle\rangle$ 为 p 关于 p' 在第 *1* 轮发送消息 m 的声明。*(c)* 在第二轮结束时，每个处理器 p 都会模拟接收以下消息。对于每个满足以下条件的处理器 p'：p 收到一条声明，称 p' 在第 *1* 轮发送了消息 m，p 都会模拟交付以下消息：*(i)* 如果 p 在第 *2* 轮从严格多数处理器处收到声明，称 p' 在第 *1* 轮发送了 m，并且 p 没有收到任何声明，称 p' 在第 *1* 轮发送了其他任何消息 $m' \neq m$，那么 p 会模拟从 p' 处收到 m。*(ii)* 否则，p 会模拟从 p' 处收到 λ。

定理 1. 算法 *1* 正确地模拟无矛盾模型。

证明。根据无矛盾模型的定义，只需证明：

(1) 如果 p_1 模拟从 q 接收 $m \neq \lambda$，那么每个处理器 p_2 都模拟从 q 接收 m 或 λ。

(2) 如果 q 是第 1 轮中行为良好的处理器，那么每个处理器都会模拟从 q 接收 q 的输入。

我们先看条目 1。假设处理器 q 的 p_1 模拟接收 $m \neq \lambda$，并考虑一个处理器 p_2。由于 p_1 模拟从 q 交付 m，p_1 会收到其在第 2 轮中所听到的处理器中严格多数处理器的声明，称 q 在第 1 轮发送了消息 m。因此，至少有一个行为良好的处理器作出了该声明，因而 p_2 也会收到该声明；根据上文的条目 (i) 和 (ii)，p_2 会模拟从 q 接收 m 或 λ。

接下来，我们讨论条目 2。考虑处理器 q 在第 1 轮中行为正常，且其输入消息为 m。由于 q 行为正常，到第 1 轮结束时，每个处理器都会收到 m，该消息来自 q。随后，在第 2 轮中，所有在第 2 轮行为正常的处理器都会广播一项声明，称 q 在第 1 轮发送了 m。此外，根据 IIAB 模型的消息限制，没有处理器能广播一项声明，称 q 在第 1 轮发送了与 m 不同的消息。因此，到第 2 轮结束时，所有处理器都会模拟收到来自 q 的 m。□

4 在无歧义模型中求解共识

我们首先定义共识问题。

定义 1. 在共识问题中，每个处理器都会收到一个外部输入，并可以产生一个输出，满足以下条件：

- 一致性：任意两个处理器都不会产生不同的输出。
- 有效性：如果所有处理器的输入都相同，为 v，则没有处理器会输出 $v, \neq v$。

此外，还必须满足以下终止性质之一。

- 最终终止：最终，所有处理器都会输出一个值。
- 概率终止：以概率为*1*，所有处理器最终都会输出。

请注意，概率终止允许某些执行中有些处理器永不输出。不过，这些执行的概率必须为零。此外，请注意，一致性和

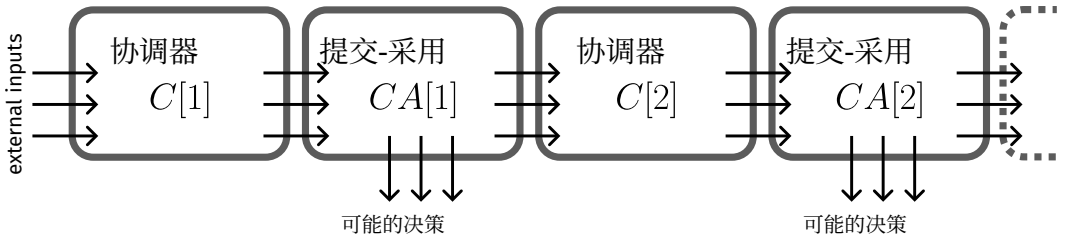
有效性必须在所有执行中得到保证（因此，像中本共识这样的算法不满足我们对共识的定义）。

4.1 通过交替使用调和器和提交-采用来解决共识

为了解决共识问题，我们采用经典方法：交替执行协调阶段和提交-采用阶段（提交-采用是一种有两个等级的分级一致协议），并从协调阶段开始。图 1 直观地展示了这一方法。

在协调阶段，处理器试图就某个值达成一致，但可能无法做到，最终产生分歧。在提交-采用阶段，如果处理器在前一个协调阶段成功达成一致，就会提交一个决策。正如我们将看到的，协调阶段和提交-采用阶段必须满足的确切性质，能够保证共识的有效性和一致性。不过，所能实现的终止性取决于处理器在什么条件下能够保证在协调阶段成功达成一致；这取决于算法以及其他假设（例如领导者选举预言机是否可用，或是否满足有界故障假设）。

图 1. 协调器与提交-采用阶段构成的无限交替序列。水平箭头表示处理器在本地将某一阶段的输出作为下一阶段的输入；垂直箭头表示处理器可能作出共识决策。无论是否已产生输出，处理器都会继续执行并进入下一阶段。



形式化地说，提交-采用算法必须遵循以下规范。

定义 2. 在提交-采用任务中，每个处理器以一个输入值开始，并必须产生形式为 $commit(v)$ 或 $adopt(v)$的输出，其中 v 是某个值，并且满足以下三个条件：

一致性 如果某个处理器输出 $commit(v)$，那么每个处理器都必须输出 $commit(v)$ 或 $adopt(v)$。有效性 如果所有处理器的输入均为 v，那么每个处理器都必须输出 $commit(v)$。终止性 存在某个轮次 R，使得所有处理器都在第 R 轮输出。

调和算法必须遵循以下规范。

定义 3. 在调解任务中，每个处理器以一个输入值开始，并且必须产生满足以下两个条件的输出。

有效性：如果存在一个值 v，使得所有处理器的输入都是 v，那么所有处理器的输出都是 v。终止性：存在一轮 R，使得所有处理器都在第 R 轮输出。

请注意，我们在后续章节中介绍的协调器算法还满足第4.3节和第4.4节所述的进展性质；这些性质要求协调器以一定概率产生一致性，或者根据算法的某个参数，确定性地产生一致性。

现在，我们给出一个通用算法，并将其定义为由两个任意的协调器算法和提交-采纳算法构成。

算法 2（通用共识算法）。我们使用一个无限的协调器实例序列 $C[1,2,\dots]$ 和一个无限的提交-采用实例序列 $CA[1,2,\dots]$。在通用共识算法中，每一轮 r，每个在线处理器 p 都执行以下操作。

(a) 如果 $r=1$，则使用 p 的外部输入调用 $C[1]$。*(b)* 从协调器 $C[n]$ 获得输出 o 后，对于某个 n，以 o 为输入调用 $CA[n]$。*(c)* 从提交-采用 $CA[n]$获得输出 o 后，对于某个 n:

- 如果对于某个 v，$o=$提交（v），则将 v 输出为共识决策。
- 以满足以下条件的值 v 调用 $C[n+1]$：$o=$提交（v）或 $o=$采用（v）。

概括而言，我们再次强调，协调阶段的目标是在条件有利时（例如，所有处理器恰好都认同一位优秀的领导者，或者系统以某种方式趋于稳定时）促使处理器达成一致，而提交-采用阶段的目标是检测已有的一致，并产生共识决策。

定理 2. 算法 *2* 满足共识的有效性和一致性性质。

证明。我们先来看有效性性质。假设所有处理器都收到相同的外部输入 v。那么，根据调和器的性质，所有处理器都会在第一个调和器 $C[1]$ 中输出 v。因此，根据提交-采用的性质，所有处理器都会在第一个提交-采用 $CA[1]$ 中决定 v，至此证明完毕。

关于一致性性质，注意，根据调和器和提交-采用的规范，一旦某个处理器从一个提交-采用实例 $CA[n]$ 中输出 commit(v)（其中 v 为某个值），那么所有处理器都会将 v 作为输入传给下一个调和器 $C[n+1]$；因此，所有处理器都会在 $C[n+1]$ 中输出 v，这意味着所有处理器都会在提交-采用 $C[n+1]$ 中提交 v，并且都决定 v。□

我们将在后续章节介绍相应的调和算法时，讨论如何实现共识的终止性质（参见定理 7 和 9）。

4.2 实现提交-采用无矛盾模型

Algo算法 3(提交-采用**)**。该算法需要进行 *2* 轮无矛盾模型 .

(a) 在第一轮中，每个在线处理器广播其输入 in_p。*(b)* 在第二轮中，如果每个在线处理器 p 听到 v 来自严格多数的消息，就广播提议提交(v)；否则，p 广播不提交。*(c)* 在第二轮结束时，每个处理器 p 按如下方式确定其输出：*(i)* 如果 p 从严格多数处收到提议提交(v)，且 v满足某条件，则输出$commit(v)$。*(ii)* 否则，如果 p 从至少一个处理器处收到一个值 v，且收到该值的处理器数量严格多于收到任何其他值的处理器数量，那么 p 输出 $adopt(v)$。*(iii)* 否则，p 输出采用 (in_p)。

尽管该算法很简单，但要证明其正确性并不容易。我们先介绍无矛盾模型的一些有用性质。

4.2.1 无矛盾模型的关键性质。本节给出定理 3 和定理 4，它们概括了无矛盾模型的关键性质，我们将在下一小节中利用这些性质证明算法正确。

首先，在定理 3 中，我们证明同一轮中，没有两个处理器会从同一处理器接收到两条不同的消息。因此，尽管参与情况未知，我们仍能依赖一种类似于许多经典拜占庭容错算法所依赖的法定人数交集性质的性质。

其次，在定理 4 中，我们证明：如果某个处理器在轮次 r 中收到 m，且没有处理器在轮次 r 中广播 m'，那么每个处理器在轮次 r 中收到 m 的次数都严格多于 m'。我们利用这一性质来确保：如果某个处理器在听到一致严格多数的消息后采取行动，其他处理器就能得到某种迹象，表明这种情况可能已经发生。这与经典拜占庭容错算法中使用的性质类似：如果某个处理器听到一致超级多数的消息，即 $3f+1$ 个中的 $2f+1$ 个，那么最终所有处理器都会听到其中 $2f+1$ 个正确处理器的消息。

先介绍一些记号约定。当上下文能够明确轮次 r 时，令 H_p 表示在 r 中听到 p 的处理器集合，令 $H_{p'}$ 表示处理器 p' 在 r 中听到的处理器集合。

首先，我们需要一个简单的引理，用来证明定理 3 和定理 4。

引理 1. 考虑第 r 轮。假设处理器 p 从处理器集合 P_m 中的每个成员 q 都收到一条消息 m。那么，对于每个处理器 p'，我们有：

(1) $P_m \subseteq H_p \cap H_{p'}$，并且 *(2)* 如果 P_m 是 H_p 的严格多数，那么 P_m 是 $H_p \cap H_{p'}$ 的严格多数。

证明。我们先证明条目 1。首先注意，根据定义，$P_m \subseteq H_p$。根据无互相矛盾模型的定义，对于每个 $q \in P_m$，如果 p' 没有收到 m，那么 p' 会收到针对 q 的 λ。因此，p' 会听说每个满足以下条件的处理器 q：p 从 q 收到 m，即 $P_m \subseteq H_{p'}$。由此，我们有 $P_m \subseteq H_p \cap H_{p'}$。

对于条目 2，请注意，我们刚刚证明了 $P_m \subseteq H_p \cap H_{p'}$。因此，如果 P_m 是 H_p 的严格多数，根据集合的简单性质，我们可得 P_m 是 $H_p \cap H_{p'}$ 中的严格多数。□

定理 3. 考虑轮次 r。对于任意两个不同的值 m_1 和 m_2，如果某个处理器 p 从严格多数处理器那里收到 m_1，那么没有处理器 p' 会从严格多数处理器那里收到 m_2。

证明。反设 p 从其获知的处理器中严格多数的 Q_1 那里收到 m_1，且 p' 从其获知的处理器中严格多数的 Q_2 那里收到 m_2。由于在无矛盾模型中，对手无法发送不一致的信息，因此 Q_1 和 Q_2 不相交。

然而，根据引理 1.2，Q_1 由 $H_p \cap H_{p'}$ 中的严格多数构成。类似地，Q_2 由 $H_p \cap H_{p'}$ 中的严格多数构成。这与 Q_1 和 Q_2 不相交这一事实矛盾。□

定理 4. 考虑轮次 r。假设某个处理器 p 从严格多数处理器那里收到 m_1。此外，假设在轮次 r 中没有处理器广播 $m_2 \neq m_1$。那么，对于每个处理器 p'，如果 $m_2 \neq \lambda$，则 p' 收到 m_2 的处理器数严格少于它收到 m_1 的处理器数。

证明。首先，注意到 $m_1 \neq \lambda$，因为至少有一个行为良好的处理器发送了 m_1。

令 $H = H_p \cap H'_p$ 为同时收到 p 和 p' 消息的处理器集合。令 $F = F_r \cap H$ 为同时收到 p 和 p' 消息的被冒充处理器集合。注意，根据模型的定义，$W_r \subseteq H$；因此 $2|F| < |H|$ 和 $F \cap H$ 是 H 的严格少数。

令 M_1 为 p 接收 m_1 的处理器集合。根据引理 1，我们有 $M_1 \subseteq H$，且 M_1 是 H 的严格多数。鉴于 F 是 H 的严格少数，我们有 $|M_1| > |F|$。因此，根据有限集合的简单性质，我们有

$$|M_1 \setminus F| > |F \setminus M_1| \tag{1}$$

令 M'_1 为 p' 从中接收 m_1 的处理器集合。令 M'_2 为 p' 从中接收 m_2 的处理器集合。我们必须证明 $|M'_2| < |M'_1|$。

由于每个行为良好的处理器都会广播唯一的消息，因此我们有 $M_1 \setminus F \subseteq M_1'$。此外，由于没有处理器会广播 $m_2 \notin \{m_1, \lambda\}$，且对手无法说谎，因此我们有 $M_2' \subseteq F \setminus M_1$。利用方程 (1)，我们很容易得到 $|M_1'| > |M_2'|$，从而完成证明。□

在补充材料中，[15], 我们提供了在 Isabelle/HOL 中对定理 4 的机械化验证证明。

现在我们可以证明提交-采用算法是正确的。

4.2.2 提交-采用算法的正确性证明。

定理 5. 算法 *3* 实现了提交-采用。

证明。首先，我们证明提交-采用的安全性。反设处理器 p输出 $commit(v)$，且处理器 p'输出 $commit(v')$ 或 $adopt(v')$ 和 $v \neq v'$。

假设 p' 按照算法 3 中的第 (i) 项输出。那么 p' 会从严格多数进程处收到 提议提交(v')。此外，由于 p输出 $commit(v)$，p会从严格多数进程处收到 提议提交(v)。结合 $v \neq v'$，这与定理 3 矛盾。

现在假设 p' 并未按照算法 3 中的第 (i) 项输出。注意，如果某个处理器广播提议提交(v)，那么根据定理 3，没有处理器会针对$w \neq v$广播 提议提交(w)。此外，由于 p输出 $commit(v)$，p 在第 2 轮会从严格多数进程处收到 提议提交(v)。因此，结合 m_1 = 提议提交(v) 和m_2 = 提议提交(v')，我们满足定理 4 的前提。于是，根据定理 4，p' 收到 提议提交(v') 的频率严格低于收到提议提交 (v) 的频率。因此，p' 按照算法 3 中的第 (ii) 项输出 $adopt(v)$。结合 $v \neq v'$，这与假设矛盾：p'输出 $commit(v')$ 或 $adopt(v')$。

接下来，我们证明提交-采用算法的有效性属性。假设所有处理器的输入都是 v。那么，每个处理器在第一轮都会从严格多数的处理器那里收到 v。因此，每个处理器都会在第二轮广播提议提交(v)。同理，每个处理器都会在第二轮从严格多数的处理器那里收到提议提交(v)。因此，每个处理器都会输出 $commit(v)$，证明完毕。□

现在我们已经介绍了提交-采用算法，接下来还需要介绍我们在两个共识算法中使用的调和器，以及它们如何实现所需的活性性质。

4.3 带概率终止的共识

本节将介绍一种调和算法。将其与第 4.1 节通用共识算法中的第 4.2 节提交-采用算法结合后，即可在期望 20 轮内实现具有确定性安全性和概率终止性的共识。

除了满足定义 3 中的性质外，本节介绍的调和算法还满足以下概率一致性性质：

定义 4（概率一致性）。 以 *1/2* 的概率，每个处理器输出相同的值。

概率协调器的思路是依靠一个良好且经一致同意的领导者（领导者选举预言机以 1/2 的概率选出该领导者），使所有处理器达成一致。然而，盲目信任领导者会危及协调器的有效性属性，因为当我们运气不好、领导者被对手冒充时，可能会出现问题。因此，我们按如下方式进行。

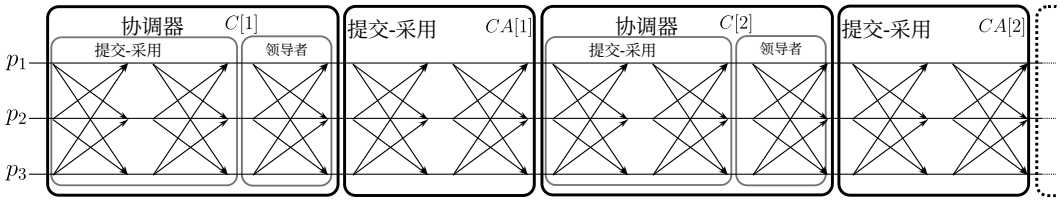
算法 4（概率协调器）。 该算法需要在无矛盾模型中执行 *3* 轮。在前两轮中，参与者按照算法 *3* 执行提交-采用。然后，在第三轮中，每个参与者 p 广播其提交-采用输出，并按如下方式输出：

(1) 如果 p 从严格多数处收到 $commit(v)$，则 p 输出 v。*(2)* 否则，如果 p 从其领导者 l_p^3 处收到 $commit(v)$ 或 $adopt(v)$，则 p 输出 v。

(3) 否则， p 输出其初始外部输入。

图 2 展示了以算法 4 作为协调器实例化的通用共识算法 r.

图 2. 三个处理器 p_1、 p_2 和 p_3 按照算法 2 执行协调阶段与提交-采用阶段交替进行的无限序列。每个协调阶段按照算法 4 包含三轮：前两轮用于执行提交-采用，第三轮中处理器尽可能输出其领导者的值。每个提交-采用按照算法 3 包含两轮。



定理 6. 算法 *4* 实现了协调器规范，并提供概率一致性保证。

证明。我们必须证明定义 3 中的有效性和终止性质，以及定义 4 中的概率一致性性质。

终止性显然成立：算法在 3 轮次内终止。

对于有效性，若所有进程都以 v 开始，那么根据提交-采用的性质，在第 2 轮次结束时，所有进程都会从提交-采用中输出 $commit(v)$，并据此广播 $commit(v)$；于是到第 3 轮次结束时，所有进程都会收到严格多数的 $commit(v)$，因此根据条目 1，所有进程都会输出 v。

对于活性性质，只需证明：若所有处理器在第 2 轮次都获得同一个行为良好的领导者 $l \in W_2$，那么它们都会输出相同的值。因此，假设所有处理器在第 2 轮次都获得同一个行为良好的领导者 l。首先注意，由于 l是行为正常的处理器，每个处理器都会收到 l的消息，并且没有处理器会根据条目 3 输出。因此，只需考虑以下两种情况：

(a) p_1 和 p_2 根据条目 1 输出。

(b) p_1 根据条目 1 输出，而 p_2 根据条目 2 输出。

首先，假设 p_1 和 p_2 都根据条目 1 输出，并反设 p_1输出 v_1，而 p_2 输出 $v_2 \neq v_1$。于是， p_1 在第 3 轮从严格多数处收到 $commit$（v_1），而 p_2 在第 3 轮从严格多数处收到 $commit$（v_2）。由于没有任何处理器会同时广播$commit$（v_1）和 $commit$（$v2$），根据定理 3，得到矛盾。

其次，假设 p_1 根据条目 1 输出，而 p_2 根据条目 2 输出。于是， p_1在第 3 轮从严格多数处收到 $commit$（v_1）。因此，至少有一个行为良好的处理器$p \in W_3$在第 3 轮广播 $commit$（v_1），所以所有处理器在第 2 轮结束时通过提交-采用输出的结果要么是 $commit$（v_1），要么是$adopt$（v_1）。因此，根据提交-采用的一致性性质（定义 2）， p_2从其领导者处收到 $commit$（v_1）或 $adopt$（v_1），并输出 v_1。□

现在我们准备证明，如果使用算法 4 的协调器，我们就能以期望 20 轮的复杂度解决共识。

定理 7. 在采用算法 *3* 作为提交-采用、采用概率协调器（算法 *4*）作为协调器的通用共识算法（算法 *2*）中，按照期望，每个处理器在 *IIAB* 模型运行 *20* 轮后都会产生共识输出。

证明。根据定理 6 和定义 4，每个协调器以概率 1/2 产生一致性；如果产生了一致性，那么后续的提交-采用必定会产生共识决策。因此，按照期望，我们会在 $CA[2]$内作出决策。这需要 2 个协调器（每个需要 3 个无 equivocation 轮次）和 2 次提交-采用（每次需要 2 个无 equivocation 轮次），总计 10 个无 equivocation 轮次，或等价地，20 个 IIAB 轮次。□

4.4 确定性安全性与确定性活性下的共识

本节将探讨如何在最终稳定假设下完全确定性地解决共识问题。

首先，我们给出一种确定性协调器算法。该算法以轮次数 $N > 1$ 为参数，并保证在 $N + 1$ 轮后达成一致性，前提是满足以下确定性协调假设：

定义 5 (确定性协调假设**)。** 满足以下条件时，确定性协调假设成立：

(1) 前 N轮中的参与情况保持不变，即对于每一轮 $r \le N$，都有 $O_r = O_1$；*(2)* 对手规模不断增长，即对于每一轮 r，都有 $F_r \subseteq F_{r+1}$；*(3)* 前 N轮中被冒充的参与者少于 N，即 $\left|\bigcup_{r \le N} F_r\right| < N$。

定义 6 (确定性一致性性质**)。** 假设确定性协调假设成立，则每个处理器在第 $N + 1$ 轮结束时输出相同的值。

接下来，我们通过使用通用共识算法（算法 2）来获得完全确定性共识算法：将其中的每个协调器 $C[n]$ 实例化为使用参数 $N = 2^n$ 的确定性协调器，并且与之前一样，将其中的每个提交-采用 $CA[n]$ 实例化为第 4.2 节中的算法。

完全确定性共识算法在以下最终稳定假设下保证确定性安全性和确定性活性：

定义 7 (最终稳定**)。** 当执行满足以下条件时，即满足最终稳定假设：

(1) 对手是递增的，即对于每一轮 r，$F_r \subseteq F_{r+1}$；并且 *(2)* 存在一轮 R，处理器并不知道这一轮，称之为稳定轮次，并且参与度在第 R 轮之后保持不变（对于每一轮 $r \ge R$，均满足$O_r = O_R$）。

其思路是，由于每次运行确定性协调器时，我们都会增加其轮次数量，因此最终会运行一个在第R 轮之后开始、且运行轮数多于故障数量的确定性协调器实例；因此，根据确定性协调器的性质，所有处理器都会输出相同的值，并在下一轮提交-采用中达成决定。

最后，通过指数级增加协调轮次，我们可以确保完全确定性共识算法在第 R轮之后的 $O\left(\left|\bigcup_{r \ge R} F_r\right|\right)$轮内终止，等价地说，即在 $O(|O_R|)$轮内终止。

4.4.1 确定性协调器。确定性协调器在 $N + 1$ 轮后输出，其中N是算法的一个参数；它必须保证协调器的有效性属性，并且在满足上述确定性协调假设的前提下，必须保证一致性，即每个处理器都输出相同的值。

为确保在上述确定性协调假设下实现一致性，我们只需使用 Dolev-Strong 算法 [9]，就第一轮中参与的处理器集合及其输入达成一致性，然后从这些输入中确定性地选择一个输出。在

确定性协调假设下，系统的行为实质上类似于传统的固定参与同步系统，该系统采用消息认证，且恶意处理器少于 N 个。因此，Dolev-Strong 算法可保证在 N 轮后达成一致性。

只有一个小问题：我们必须保证所有处理器都输出相同的值，包括被冒充的处理器。只需增加一轮即可轻松实现这一点：在这一轮中，处理器广播各自的候选输出，并在收到的值中存在严格多数时输出该值；否则输出自己的候选输出。

最后，即使确定性协调假设不成立，我们也必须保证有效性属性。幸运的是，只需调整用于选择输出的确定性规则即可轻松实现这一点：我们要求每个处理器在存在某个值被严格多数的处理器持有时，输出该值。

现在我们给出完整的算法。请注意，它就是 Dolev-Strong 算法，只是在最后一轮使用了特定的输出选择规则，并在末尾增加了一轮。

算法 5（确定性协调器）。 该算法以自然数 $N > 0$ 为参数，并运行 N 轮。每个处理器 p 都维护一个局部变量 e_p，其初始值为空集。

- 在第1轮的发送阶段，每个在线处理器 p 都会广播签名消息 $\langle p, 1, in_p\rangle$，其中 in_p 是 p的外部输入。
- 在每一轮 $1 \leq r \leq N$的接收阶段，每个在线处理器 p 执行以下操作：对于其收到的每条形如 $\langle p_r, r, \langle p_{r-1}, r-1, \ldots \langle p_1, 1, v\rangle\rangle\rangle$ 的消息，如果序列中没有处理器重复出现，且 p_1 未出现在集合 e_p中的任何元组里，则 p 将元组 (p_1, v) 添加到集合 e_p中。
- 在每一轮 $1 < r \leq N$的发送阶段，每个在线处理器 p 首先将局部变量 m_p^r初始化为空集。然后，对于上一轮 $r-1$收到的每条形如 $\langle p_{r-1}, r-1, \langle p_{r-2}, r-2, \ldots \langle p_1, 1, v\rangle\rangle\rangle$ 的消息 m，如果序列中没有处理器重复出现，则处理器 p将签名消息 $\langle p, r, m\rangle$添加到 m_p^r中。最后，处理器 p广播 m_p^r。
- 在第 $N+1$ 轮的发送阶段，每个在线处理器 p 都会按如下方式确定候选输出并将其广播。– 如果存在一个值 v，使得出现在元组中的处理器中有严格多数出现在元组中，且该元组的第二个分量为 v，那么 p 的候选输出就是 v。– 否则，p 的候选输出就是元组 e_p 中出现的最小值。
- 最后，在第 $N+1$ 轮结束时，每个处理器按如下方式输出：– 如果 p 从其所听闻的处理器中的严格多数那里收到值 v，那么 p 输出 v。– 否则，p 输出自己的候选输出。

引理 2。 如果确定性协调假设成立，那么每个未被冒充的处理器 p 在第 N 轮结束时都拥有相同的集合 e_p。

证明。首先注意，如果处理器 p 在第 N 轮将元组 p', v 加入其集合 e_p，那么它必定在第 N 轮收到了一条形如 $p_N, N, p_{N-1}, N-1, p_{N-2}, N-2, \ldots p_1, 1, v$ 的签名消息，其中序列中没有处理器重复。由于 $N > \left|\bigcup_{r \leq N} F_r\right|$，序列 $p_1, p_2, \ldots, p_N$ 必定包含一个在所有轮次中行为正常的处理器 p_j。因此，在第 j 轮，p_j 会在第 j 轮广播 $p_j, j, p_{j-1}, j-1, \ldots p', 1, v$。因此，每个处理器 p'' 最迟会在第 j 轮结束前，将元组 p', v 加入其集合$e_{p''}$。

接下来请注意，如果行为良好的处理器 p在第 $r < N$ 轮将元组 p', v 添加到其集合 e_p 中，那么它必定在第 r 轮收到过形如 $p_r, r, p_{r-1}, r-1, \ldots p', 1, v$ 的消息，其中序列中没有处理器重复。因此，p在第$r+1$轮广播 $p, r+1, p_r, r, \ldots p', 1, v$。此外，$p \notin \{p_r, p_{r-1}, \ldots, p'\}$ 或 p', v 本来就会在 e_p 中。因此，最迟到第 $r+1$ 轮结束时，每个处理器 p'' 都会将 p', v 添加到 $e_{p''}$ 中。

我们由此得出，在第 N 轮结束时，每个未被冒充的处理器 p 都拥有相同的集合 e_p。□

现在我们可以证明，调和算法是正确的。

定理 8. 算法 *5* 实现了一个具有额外确定性一致性性质的调和器。

证明。我们必须证明算法 5 满足调和器的有效性和终止性性质（定义 3），以及确定性一致性性质（定义 6）。

终止性显然成立：根据定义，该算法在 $N+1$ 轮次内终止。

对于有效性，注意，对于每个行为良好的处理器 p'，每个处理器都会在第 1 轮结束时将 $p', in_{p'}$ 添加到集合 e_p 中。此外，注意，对于任意两个处理器 p 和 p'，如果对于某个 v，有 $p', v \in e_p$，那么 p' 在第 1 轮处于在线状态。因此，如果所有行为良好的处理器具有相同的输入 v，那么对于每个处理器 p，v 都会出现在 e_p 中严格超过一半的元组中。因此，所有行为良好的处理器都会在第 $N+1$ 轮广播 v，并且都输出 v。

最后，对于确定性一致性性质，假设确定性协调假设成立。那么，根据引理 2，在第 N 轮结束时，每个未被冒充的处理器 p 都有相同的集合 e_p，因此所有此类处理器都会在第 $N+1$ 轮广播相同的候选输出。由于对手不断扩张且参与度保持不变，未被冒充的处理器数量必然多于被冒充的处理器。因此，在第 $N+1$ 轮，每个处理器都会从其听到消息的处理器中的严格多数处收到相同的值 v，从而每个处理器都会输出 v。□

4.4.2 完全确定性共识。

定理 9。 假设最终稳定假设成立。考虑采用算法 *3* 作为提交-采用、并对每个 n 采用本节的确定性协调器（算法 *5*）且参数为 $N = 2^n$ 的通用共识算法（算法 *2*），其中 $C[n]$ 为该确定性协调器。该算法保证每个处理器在稳定轮次之后最多经过 $O(|\bigcup_{r \geq R} F_r|)$ 轮即可产生共识输出 R。

证明。令 $b = \lceil \log_2 (|\bigcup_{r \geq R} F_r|) \rceil + 1$。若 $b \geq R$，由于 $2^b > |\bigcup_{r \geq R} F_r|$，根据定理 8，每个处理器都会在协调器 $C[b]$ 中产生相同的输出。否则，每个处理器都会在第 R 轮之后的第一个协调器中产生相同的输出。无论哪种情况，每个处理器都会在下一次提交-采用结束时作出共识决策。

注意，提交-采用固定需要 3 轮，而每个协调器 $C[n]$ 需要 $2^n + 1$ 轮。因此，最多经过 $3b + \sum_{i=1}^{b} 2^i + 1 = O(|\bigcup_{r \geq R} F_r|)$ 即可到达协调器 $C[b]$；并且我们可以保证，所有处理器都会在第 R 轮之后经过 $O(|\bigcup_{r \geq R} F_r|)$ 后输出。□

5 相关工作

为了简化对分布式计算问题的推理，许多早期研究采用了消息对手这一概念：它控制通信，而处理器本身不会发生故障。Afek 和 Gafni [1] 研究了一个存在消息遗漏的同步系统，其中出站消息的遗漏由消息对手造成。在该模型中，处理器不会发生故障，并会根据收到的消息执行算法。因此，在该模型中解决任务时，所有处理器都必须输出，而基于故障的模型中只有正确的处理器才必须输出。利用消息对手模型，Afek 和 Gafni 大大简化了异步可计算性定理的证明。

本文采用类似的设定：处理器永不发生故障，但恶意对手可能会暂时冒充该处理器，并代表它发送消息。Santoro 和 Widmayer [24]

还研究了存在良性瞬时通信故障时的共识问题；这类故障与 Afek 和 Gafni 的消息对手相似。他们证明，即使每轮只有一次良性故障，共识也是不可能实现的。

在一份写于2019年（现已发表于 arXiv [12]）的未发表手稿中，Gafni 和 Zikas 在参与情况固定且已知的认证拜占庭同步设置中引入了移动消息对手。他们展示了如何在由此产生的模型中求解提交-采用问题，而我们提出的提交-采用算法（算法 3）借鉴了其主要算法思想。然而，在 IIAB 模型中引入参与情况未知这一条件，使问题显著复杂化。Gafni 和 Zikas 还为其模型提出了一个有缺陷的共识算法（缺陷在于，他们的协调器不满足定义 3 中的有效性属性）。

Algorand [6] 通过逐轮推进，为每一轮选出一个认证委员会。委员会成员事先并不知道其他委员会成员是谁，但收到委员会成员发来的消息时，仍可验证对方是否有资格成为委员会成员。权益证明能够以概率方式确保委员会中诚实成员的比例为 1/3。与我们的设置相比，主要区别在于，在 Algorand 中，处理器知道委员会的基数，即使这一信息仅在期望意义上成立。这对算法设计有深远影响：在 Algorand 中，处理器可以高概率地检查委员会中是否有某个比例 f的成员支持某个值；而在 IIAB 模型中，处理器只能检查其获知的处理器中是否有某个比例 f支持某个值。

Pass 和 Shi [20] 提出了休眠模型，在该模型中，处理器会唤醒并参与，也可能重新休眠。因此，与 Algorand 一样，参与情况事先未知；但不同于 Algorand，参与者数量未知（甚至无法预期）。他们考虑少数派腐化，并采用最长链机制来实现概率安全性。Snow-White论文 [8] 将 Pass 和 Shi 的工作扩展到权益证明场景，并提出了实用算法。

Sandglass [21] 即使在故障集合可能逐轮扩大的情况下，也能在少数良性故障条件下实现确定性安全性。我们提出的算法也适用于 Sandglass 模型，因为其中不存在拜占庭故障。Gorilla [22]将 Sandglass 扩展为可容忍恶意故障。与当前工作相比，Gorilla 不使用消息认证，而是使用可验证延迟函数（可验证延迟函数） [4] ，以实现一种具有确定性保证的工作量证明形式。

Monmose 和 Ren [19] 在故障比例低于 1/2 的假设下，于休眠模型中实现了确定性安全性，但他们的算法需要最终稳定参与，才能以概率为1取得进展。Malkhi、Momose 和 Ren [16] 取消了最终稳定性要求，但代价是将腐化比例限制在至多 1/3。随后，Malkhi、Momose 和 Ren [17]在没有最终稳定化要求的情况下，实现了故障比例为 1/2 时的确定性安全共识。这与第 4.3 节中所述算法的保证相符。

在概率安全领域，从比特币的最长链协议开始，一系列研究以概率安全性解决了动态参与下的共识问题 [20, 8, 2, 7, 13]。

5.1 结论

就像所有优秀的基础研究一样，Gafni 和 Zikas 于 2019年 撰写但未发表的论文（现已发表于 arXiv [12]）当时被所有知情者认为优雅却晦涩，三年后人们发现其中包含一种适用于无许可系统的提交-采用算法思想。

本文基于 Gafni 和 Zikas 的工作，在一种称为 IIAB 模型的无许可系统通用模型中实现共识。在有界对手下（这涵盖了最终稳定参与的情形），我们确定性地实现共识；即使面对无界对手，我们也能以概率方式实现共识，但依赖于一个可通过可验证随机函数实现的领导者选举预言机。

我们提出结构化的模块化算法，并未试图降低消息复杂度或轮次复杂度。因此，我们得到了极其简单的算法，明天就可以拿到课堂上讲授。为此，我们遵循关注点分离这一由来已久的传统：首先以尽可能清晰的方式探究可行性，然后再优化效率。后者留待未来工作。

最后但同样重要的是：著名的 FLP 结果 [10] 表明，如果要求处理器每轮都继续执行，而每轮恰有一个处理器缺席，那么n 就可能阻止共识。Santoro 和 Widmayer [24] 在一个同步系统中证明了类似的不可能性结果：每轮有一个处理器遭受消息遗漏故障，且受影响的处理器可能每轮都不同。

这两种设定引出了两种不同的最终同步性概念（后一种更适合称为最终稳定性）。第一种是 FLP 意义上的最终同步性，即系统先是异步的，之后最终变为同步。第二种是 Santoro 和 Widmayer 意义上的最终稳定性，即系统是同步的，但存在移动故障；之后故障最终变为固定不变（也就是说，最终至少有一个处理器不受影响）。

前一种最终同步性概念在数学上表现出不规则性——认证拜占庭模型中，即使存在少数故障处理器，也能在同步系统中解决共识[1], ；但在最终同步的情形下，故障比例必须低于三分之一。最终稳定性概念不存在从二分之一到三分之一的这种差距：正如我们所展示的，对于少数腐化，IIAB 模型在最终稳定性条件下可以解决共识。

参考文献

[1] Yehuda Afek 和 Eli Gafni。2015年。异步计算的简单刻画。理论计算机科学，561，B辑。doi: 10.1016/j.tcs.2014.07.022。[2] Christian Badertscher、Peter Gaži、Aggelos Kiayias、Alexander Russell 和 Vassilis Zikas。2018年。Ouroboros Genesis：具备动态可用性的可组合权益证明区块链。载于*2018*年*ACM SIGSAC*计算机与通信安全会议论文集。doi: 10.1145/3243734.3243848。[3] P. Berman、J. A. Garay 和 K. J. Perry。1989年。迈向最优分布式共识。载于第*30*届年度计算机科学基础研讨会论文集。doi: 10.1109/SFCS.1989.63511。[4] Dan Boneh、Joseph Bonneau、Benedikt Bünz 和 Ben Fisch。 [无日期。] 可验证延迟函数。载于密码学进展 – *CRYPTO 2018* （计算机科学讲义）。Hovav Shacham 和 Alexandra Boldyreva（编）。Springer International Publishing，Cham。doi: 10.1007/978-3-319-96884-1_25。[5] Elizabeth Borowsky 和 Eli Gafni。1997年。无等待计算的一种简单算法推理刻画（扩展摘要）。载于第十六届*ACM*分布式计算原理年度研讨会论文集。doi: 10.1145/259380.259439。[6] Jing Chen 和 Silvio Micali。2019年。Algorand：安全高效的分布式账本。英文。理论计算机科学，777。doi: 10.1016/j.tcs.2019.02.001。[7] Francesco D'Amato、Joachim Neu、Ertem Nusret Tas 和 David Tse。2022年。不再有针对权益证明以太坊的攻击？*arXiv*预印本 *arXiv:2209.03255*。[8] Phil Daian、Rafael Pass 和 Elaine Shi。2019年。Snow White：稳健的可重配置共识及其在可证明安全的权益证明中的应用。英文。载于金融密码学与数据安全。doi: 10.1007/978-3-030-32101-7_2。[9] D. Dolev 和 H. R. Strong。1983年。用于拜占庭协议的认证算法。*SIAM*计算杂志，12。doi: 10.1137/0212045。[10] Michael J. Fischer、Nancy A. Lynch 和 Michael S. Paterson。1985年。存在一个故障进程时分布式共识的不可能性。*ACM*杂志，32，（1985年4月）。doi: 10.1145/3149.214121。[11] Eli Gafni。1998年。逐轮故障检测器（扩展摘要）：统一同步与异步。载于第十七届*ACM*分布式计算原理年度研讨会论文集。doi: 10 2 77724 1145 277697。/ . .[12] Eli Gafni 和 Vasileios Zikas。2023年。同步/异步与静态/移动？后者在理论上更胜一筹……（2023年）。doi: 10.48550/arXiv.2302.05520。[13] Vipul Goyal、Hanjun Li 和 Justin Raizes。2021年。休眠模型中的即时区块确认。英文。载于金融密码学与数据安全。doi: 10.1007/978-3-662-64331-0_4。[14] Andrew Lewis-Pye 和 Tim Roughgarden。2023年。无需许可共识。（2023年）。doi: 10.48550/arXiv.2304.14701。

[1]有人声称，Dolev-Strong 算法可实现 99% 的容错性，但只要至少一半进程出现故障，该算法就无法满足共识的有效性属性；此外，如果至少一半进程出现故障，实现共识也没多大用处，因为外部观察者无法判断未故障的进程作出了什么决定。

[15] [SW] Giuliano Losa，nano-o/dynamic-participation-supplemental：TLA+ 规范与 Isabelle/HOL 理论，2023 年 10 月 8 日。doi：10.5281/zenodo.8418165，网址：https://zenodo.org/record/8418165。[16] Dahlia Malkhi、Atsuki Momose 和 Ling Ren。2022 年。完全波动式参与下的拜占庭共识。报告编号：1448。（2022 年）。检索于 2022 年 11 月 6 日，网址：https://eprint.iacr.org/archive/2022/1448/20230910:162058。[17] Dahlia Malkhi、Atsuki Momose 和 Ling Ren。2023 年。迈向实用型休眠式拜占庭容错。出版信息：发表于其他刊物。ACM CCS 2023。（2023 年）。检索于 2023 年 9 月 12 日，网址：https://eprint.iacr.org/2022/1448。[18] S. Micali、M. Rabin 和 S. Vadhan。1999 年。可验证随机函数。载于 第*40*届计算机科学基础年会。doi：10.1109/SFFCS.1999.814584。[19] Atsuki Momose 和 Ling Ren。2022 年。休眠式共识中的恒定延迟。载于 *2022*年 *ACM SIGSAC*计算机与通信安全会议论文集。doi：10.1145/3548606.3559347。[20] Rafael Pass 和 Elaine Shi。2017 年。休眠式共识模型。英文。载于 密码学进展 – *ASIACRYPT 2017*。doi：10.1007/978-3-319-70697-9_14.[21] Youer Pu、Lorenzo Alvisi 和 Ittay Eyal。2022 年。安全的无许可共识。载于 第*36*届国际分布式计算研讨会（*DISC 2022*）。第 246 卷。doi：10.4230/LIPIcs.DISC.2022.33。[22] Youer Pu、Ali Farahbakhsh、Lorenzo Alvisi 和 Ittay Eyal。2023 年。Gorilla：安全的无许可拜占庭共识。（2023 年）。doi：10.48550/arXiv.2308.04080。[23] N. Santoro 和 P. Widmayer。1989 年。时间并非良药。载于 第六届计算机科学理论方面年会论文集（*STACS 89*）。检索于 2023 年 2 月 3 日，来源未注明。[24] Nicola Santoro 和 Peter Widmayer。2007 年。普遍故障下同步网络中的一致性。英文。理论计算机科学，384。doi：10.1016/j.tcs.2007.04.036。