计算机科学
COMPUTER SCIENCE

异步网络模型下的共识协议研究

王迪, 雷航, 曹广平

引用本文

王迪, 雷航, 曹广平. 异步网络模型下的共识协议研究[J]. 计算机科学, 2025, 52(4): 310-326.

WANG Di, LEI Hang, CAO Guangping. Research on Consensus Protocols in Asynchronous Network Model [J]. Computer Science, 2025, 52(4): 310-326.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240600132

异步网络模型下的共识协议研究

王 迪[1,2] 雷 航[1] 曹广平[2]
1 电子科技大学信息与软件工程学院 成都 610054
2 中国西南电子技术研究所 成都 610036

摘 要 随着分布式系统的发展,共识问题受到了计算机领域的广泛关注。然而,FLP不可能结论指出:“在存在故障节点的异步系统中,没有确定的共识协议能够使分布式系统达成一致”。该结论成为阻碍设计异步共识协议的鸿沟。目前,研究者就如何规避FLP结论,已在异步共识领域进行了大量研究。首先,通过对分布式共识问题的相关定义与理论进行分析,总结出异步共识协议的定义;然后,根据规避FLP结论策略的不同,分别阐述了异步共识协议的发展脉络、实现方法与相关指标,分析总结了通过随机化、部分同步模型、故障检测器、条件限制与混合共识的方法规避FLP结论的优劣,指出了异步共识协议大多仍停留在理论阶段,难以真正应用;最后,简单探讨了共识中的等价性问题,期望拓展共识协议的实现途径,推动异步共识协议的创新和发展。

关键词:异步共识;FLP结论;随机化;部分同步;故障检测器;拜占庭故障
中图分类号 TP311

Research on Consensus Protocols in Asynchronous Network Model

WANG Di[1,2],LEI Hang[1] and CAO Guangping[2]
1 School of Information and Software Engineering,University of Electronic Science and Technology of China,Chengdu 610054,China
2 Southwest China Institute of Electronic Technology,Chengdu 610036,China

Abstract As the development of distributed systems progresses,the consensus problem has garnered widespread attention in the field of computer science. The inevitable asynchrony in distributed systems greatly impacts the design of consensus protocols. However,the FLP impossibility result indicates that in asynchronous systems,there is no definitive consensus protocol capable of enabling a distributed system to reach agreement. Researchers have conducted extensive studies on how to circumvent the FLP conclusion in the field of asynchronous consensus. This paper first analyzes the relevant definitions and theories of the distributed consensus problem,summarizing and concluding the definition of asynchronous consensus protocols. Then elaborates on the development trajectory,implementation methods,and performance metrics of asynchronous consensus protocols from the perspective of different strategies to circumvent the FLP conclusion. The paper analyzes and summarizes the advantages and disadvantages of avoiding the FLP conclusion through methods such as randomization,partial synchronous models,failure detectors,conditional restrictions,and hybrid consensus approaches. It points out that the design of asynchronous consensus protocols mostly remains at the theoretical stage and is difficult to put into practical use. Finally,briefly discusses the issue of equivalence in consensus,hoping to expand the implementation pathways of consensus protocols and promote the innovation and development of asynchronous consensus protocols.

Keywords Asynchronous consensus,FLP impossibility,Randomization,Partial synchrony,Failure detectors,Byzantine fault

1 引言

分布式系统的一致性问题,已有半个多世纪的研究历史。早在1959年,Eisenberg等就提出了一种主观概率的共识方法[1],主要研究在特定概率空间中,每个个体具有其独特的主观概率分布时,如何达成一个共同认可的概率分布。随着计算机网络与分布式系统的发展,共识问题受到了计算机领域的广泛关注,逐渐成为分布式存储系统、区块链、云计算等的核心技术之一。

1975年,首个被证明无解的通信问题——两军问题[2]的提出,标志着计算机领域对分布式共识问题研究的开始。随后,Pease等探讨了分布式系统中的一致性问题及其面临的挑战[3],即如何设计一种算法,让一组可能包含故障或恶意成员的分布式节点通过点对点通信达成共识。1987年,Dolev等[4]更加深入地研究了不同条件对分布式系统共识的作用,探讨了故障模型、同步条件、消息传递可靠性、进程启动时间等对达成共识的影响,并尝试确定实现共识所需的最小同步条件。

研究者们已从不同的角度对共识协议进行了梳理,文献[5]重点关注了共识协议的出块节点选举和主链共识两个

到稿日期:2024-06-21 返修日期:2024-09-21
通信作者:王迪(di_wang0227@foxmail.com)

主要步骤，并分别对两个步骤不同的实现机制进行了详细的分析与比较。文献[6]将共识协议分为竞争类、选举类、随机类与其他类，并分析了不同类型中具有代表性的共识协议及发展历程，还建立了一套共识算法的评价体系，对各类共识协议进行了初步的分析评估。文献[7]从共识协议的网络模型、腐化模型与敌手模型出发，并基于3类模型分别对典型共识协议进行分析比较。文中虽然提到了异步网络模型，但仅简单阐述了3种异步共识协议。文献[8]将共识协议分为基于某种属性证明、基于节点投票和类Paxos 3类共识协议，并通过蒙代尔不可能三角理论进行了分析对比。文献[9]根据共识协议实现原理的不同，选取了PoW(Proof of Work)，PoS(Proof of Stake)，PoW＋PoS和PoW/PoS＋BFT/PBFT(Practical Byzantine Fault Tolerance)的代表性共识协议进行了分析。文献[10]从公有链、联盟链和私有链的角度出发，通过建立量化的评价体系，对不同共识协议进行比较与总结。

虽然针对共识协议已有大量研究，但国内外在该领域的研究与实现主要是基于部分同步网络模型开展，未见针对异步网络模型下共识协议的分析与总结。然而，在一个分布式环境中，消息传递所需的时间、时钟的漂移量以及节点执行单个操作所需的时间都没有固定上限[11]，因此，异步性是分布式共识协议设计必须要考虑的问题[12]。1985年，Fish等提出并证明了以三人名字首字母命名的FLP(Fischer-Lynch-Paterson)[13]不可能结论，指出在基于异步通信模型的分布式系统中，不存在确定的共识协议能够使系统达成一致。FLP结论成为设计异步共识协议必须规避的结论与难点，节点间通信模型的差别也极大地影响了共识协议的设计。

迄今为止，研究者们通过修改系统模型、随机化或引入新的组件来寻求解决异步共识的“工程解”，并试图不断提升容忍故障节点的比例，降低通信轮次、计算与消息的复杂度，优化共识期望轮次等，以达到提升异步共识效率的目的。本文致力于梳理与研究异步共识协议的实现途径与发展脉络，以期为未来异步共识协议的设计和创新提供参考。本文的主要研究内容如下：

1)分析、总结分布式一致性的定义与分布式共识领域的重要结论，给出异步共识协议的定义，并分析其应用场景和实现难点；

2)根据规避FLP不可能结论方法的不同，包括随机化、部分同步模型、故障检测器、基于条件限制与混合共识等方式，梳理异步共识协议的实现方法与研究进展，并总结分析各种异步共识协议的特点与性能指标；

3)总结了异步共识协议的发展脉络，并对不同异步共识协议进行量化分析与比较，分析了多种规避FLP结论途径的优劣，并且简单阐述了共识中的等价性问题；

4)总结了异步共识协议的优势，并对异步共识协议的发展提出了一些思考。

2　异步共识协议

2.1　定义

“异步”是指在分布式系统中，节点间进行数据交换时的一种通信模式。在这种模式中，消息传递和节点操作的执行都没有固定或预先确定的时间限制，即消息的传输时延和处理速度未知，系统不存在全局时钟。异步产生的原因包括网络延迟、处理速度的差异及节点故障等，而不同的节点故障类型会影响共识协议的设计以及能够容忍的故障节点数。根据协议对故障的容错能力，可分为崩溃容错协议(Crash Fault Tolerant，CFT)和拜占庭容错协议(Byzantine Fault Tolerant，BFT)，其分别对应如下缺陷类型。

宕机缺陷(Fail-Stop)：故障节点可能在任何时候停止发送消息，但在首次停止发送消息之后，它们宕机并停止参与协议。

拜占庭缺陷(Byzantine)：拜占庭缺陷又称为拜占庭将军问题，是指故障节点可能会任意背离协议，发送错误、矛盾和误导性的消息[14]。拜占庭将军问题由Lamport等[15]提出，作者详细讨论了在分布式系统中，如何确保在存在恶意行为的组件时，系统仍可靠地运行。该问题在分布式系统领域得到了广泛的研究[16]，并产生了各种解决方案和共识协议，如实用拜占庭共识协议PBFT[17]。

共识协议则是在存在故障节点的分布式系统中实现对某种状态或数据一致性的方法[18]，它涉及到系统中多个节点之间相互协调，以应对节点可能的故障[19]，最终达成对特定状态的共同认可。在这个过程中，“共识”指的是节点间达成一致的过程，而“一致性”则是指最终状态的统一结果。共识问题可以归纳定义为[20-21]一致性、有效性和可终止性，它们称为共识问题的3个属性。共识协议同时满足一致性和有效性时称为满足安全性，满足可终止性时称为满足活性。

一致性(Agreement)：如果一个正确的节点提出值v，那么所有正确的节点都提出v。

有效性(Validity)：如果一个正确的节点提出值v，那么至少有一个正确节点将v作为输入。

可终止性(Termination)：每一个正确节点都必须有一个最终确定的值。

综上所述，异步共识协议就是在可能存在故障或恶意节点的分布式异步网络中，节点间通过一定的交互逻辑，实现满足共识问题3个属性的协议。

2.2　应用

数字货币及区块链技术的诞生，成为共识协议发展的分水岭，从此共识协议被分为经典共识和区块链共识两类。

经典共识主要应用于分布式数据库的数据同步与故障容错。分布式数据库一般节点数量有限，运行及网络环境相对可靠，因此经典共识协议通常不具备拜占庭容错与适应异步网络的能力。

区块链系统可根据节点管理组织和参与方式的不同，分为私有链、公有链与联盟链。私有链由单一实体控制，具有较强的封闭性，常用于存证与管理企业内的部审计信息、财务信息等有着不可篡改需求的核心数据，经典共识则能满足其数据一致的需求。公有链一般用于数字货币、数字资产的交易或流转，其运行在一个未知且缺乏信任的开放环境中，网络可靠性难以保障，节点之间相互不信任，因此共识协议必须具有拜占庭容错与在异步网络中保持活性的能力。而联盟链则是介于两者之间，由多个组织或企业参与维护和使用，他们通过共同维护一致的账本来达成某些目标或业务需求。联盟链共识协议的选择由节点受信任程度、业务

安全需求与实际网络环境决定。

私有链仅由单一实体运行维护，并不能发挥区块链系统分布式共享协同的特点，不是理想的区块链形态。而公有链、联盟链作为区块链系统的主要类型，选择使用异步拜占庭共识协议，明显有着更广阔的应用范围。

2.3 实现难点

异步共识协议难以实现的核心在于异步网络所带来的不确定性。在异步网络中，难以区分节点是宕机还是运行缓慢或消息传递延迟，异步共识协议需要处理未知的延迟，而不能假设所有消息都会在预定时间内到达；此外，异步网络中时钟不同步，这进一步增加了确定消息传递顺序和处理时间的复杂性。然而，不确定性问题在拥有时间假设的部分同步网络中是不存在的，并且同步网络作为更为理想的网络环境，消息甚至能够在已知时间内按照一定的顺序到达所有节点。

异步网络的不确定性与时钟无法同步，给共识协议的设计带来了更多的挑战。1）容错性挑战：共识协议需在面对故障节点时继续运行并达成共识，但在异步网络中，因为无法通过超时来准确判断节点是否故障，故障检测变得难以实现。2）活锁和死锁挑战：在异步网络中，共识协议可能会陷入活锁（即系统无法取得进展）或死锁（即系统完全停止），这增加了共识协议活性设计的复杂度。3）性能和效率挑战：异步网络下，共识协议可能需要引入额外的消息交换或等待周期，这极大地增加了共识协议的消息复杂度和通信复杂度，使得大多异步共识协议仅停留在理论阶段，难以应用。4）安全性挑战：异步网络中的不确定性可能被恶意节点利用，增加了安全风险。

2.4 FLP 结论

FLP[13]通过证明，得出了在存在故障节点的异步系统中，不存在确定性的协议能够实现共识的结论。Lout[22]使用基本相同的技术证明了一个类似的结果，表明在至少有 1 个不可检测故障节点的异步系统中，共识是不可能的。

FLP 结论的诞生，促使研究者们放弃了直接将同步系统中的共识解决策略应用于异步环境的尝试，转而探索新的方法来绕过 FLP 的限制，以实现异步系统中的共识。

3 异步共识协议的实现途径

3.1 随机化

通过为共识协议增加一个随机输入，修改共识问题“三属性”中“可终止性”的定义，将共识协议的终止从确定性变为概率性，从而实现异步环境下的概率性共识，这就是一种可能的规避 FLP 结论的策略。目前大部分的异步共识协议都采用随机化方法来规避 FLP 不可能结论，通过节点间若干轮次的通信交互，使系统以逐渐趋于 1 的概率达成共识。

3.1.1 Rabin 和 Ben-Or 抛硬币法

1976 年，Rabin 探讨了概率算法（Probabilistic Algorithms）的概念和应用[23]，这些算法在设计和分析中使用随机性来提高效率或解决确定性算法难以解决的问题。1983 年，该作者利用概率算法提出了一个解决异步环境下拜占庭共识问题的随机化解决方案[24]。该方案基于秘密共享与数字签名，通过投票（Polling）、抽签（Lottery）和决策（Decision）3 个主要步骤来实现。1）投票阶段：每个进程试图了解其他进程关于共同决策的意见。2）抽签阶段：各进程通过一个随机过程来确定一个共同的随机比特（秘密共享值）。3）决策阶段：使用抽签阶段得到的秘密值来决定是否接受投票阶段的多数派候选消息。

这 3 个步骤整合在一起，旨在即使存在故障进程的情况下，也能使所有正确的进程达成一致。投票阶段收集信息，抽签阶段引入随机性以抵抗故障进程的影响，决策阶段则根据前两个阶段的结果来更新每个进程的消息值。通过重复这些步骤，协议最终能够在有限的轮次内，以一定的概率达成一致。该协议实现共识的预期轮次为 4 次，容忍故障节点数为小于节点总数的 1/10，即 $f<n/10$（f 为故障节点数，n 为节点总数），通信复杂度为 $O(n^2)$；同时，作者指出通信复杂度可以通过优化降低到 $O(nf)$。

Ben-Or[25]专注于完全异步的模型，于 1983 年提出了一种解决异步网络环境下共识问题的概率性解决方案。该方案包含两种共识协议，分别针对崩溃故障和拜占庭故障。

针对崩溃故障的共识过程如图 1 所示，包括以下步骤。

1）初始化：初始化输入值 X_p，设置轮次编号 $r=1$。

2）发送初始值：进程 P 向所有其他进程发送消息$(1,r,X_p)$，其中 1 为消息类型的标识。

3）收集反馈：等待来自 $n-f$ 个进程的消息。如果超过 $n/2$ 的消息具有相同的值 v，则发送消息$(2,r,v,D)$给所有进程，其中 2 表示确认消息，D 表示决定值。如果没有超过半数相同值，则发送消息$(2,r,?)$。

4）决定值：等待来自 $n-f$ 个进程的$(2,r,*)$类型消息。如果收到一个 D 消息，则 $X_p=v$。如果收到超过 f 个 D 消息，则决定 v 为共识值。否则，$X_p=\text{random}\{0,1\}$。

5）下一轮：轮次编号 r 增加 1，并重复步骤 1）。

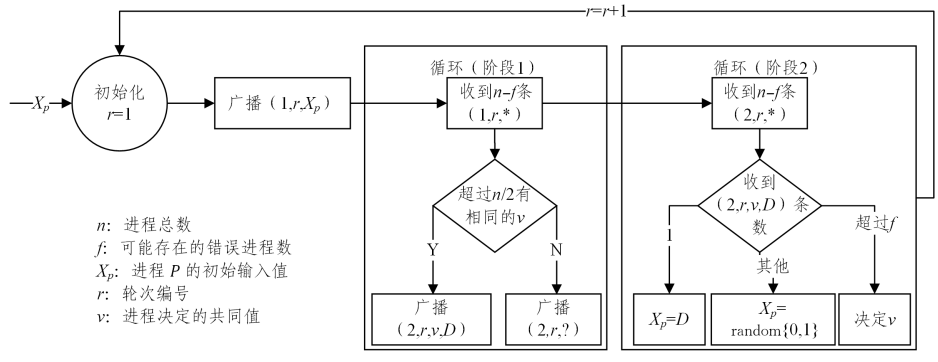


图 1 Ben-Or1983 非拜占庭共识协议

Fig. 1 Ben-Or1983 Non-Byzantine Consensus Protocol

拜占庭故障的共识步骤与崩溃故障的步骤完全一致，仅是在收集反馈和决定值阶段对于各条件判断的阈值有所不同。

Ben-Or 的方案更侧重于实际应用中的可行性和效率，其将容忍故障节点数提升至 $f<n/5$。该文指出，如果故障进程的数量 $f=O(\mathrm{sqrt}(n))$，那么文中所提及共识协议达成共识的预期轮数是常数，即不依赖于进程总数 n。同时，作者指出共识至少需要 $f+1$ 轮通信才能保证在存在拜占庭进程的情况下达成共识[26-27]。

概率性终止的提出，成为了解决异步共识问题的重要途径之一。其中，“抛硬币”作为一种随机化手段，在后来的研究中被广泛运用。根据抛硬币方法的不同，可以分为以下两类。

1）局部抛硬币：参与者各自独立地生成随机数，并通过相互交互来达成对随机结果的共识。这类方法通常需要指数轮通信，参与者需要交换他们的随机数以确保全局的一致性。

2）全局抛硬币：通常依赖于一个可信的第三方，在系统初始化阶段生成并分发随机数或随机数的份额给系统中的每个参与者。在协议执行过程中，通过密码学技术实现随机数的秘密共享，参与者使用这些预共享的随机数或份额来进行“抛硬币”，并决定共识协议的后续行为。这类协议通常可在常数轮达成共识，但初始化阶段有较高的设置成本，会另外增加通信和计算开销。

3.1.2　秘密共享与异步二元广播

Toueg[28]在 Rabin 提出的共识协议基础上进行改进，设计了一种异步广播算法 echo_broadcast。这是一种两阶段异步广播协议，包含初始（initial）阶段和回声（echo）阶段，确保了异步网络环境中，系统中的非故障进程也能就广播的消息达成一致；同时，该协议基于 Shamir[29]的秘密共享实现了 compute_secret 方案，该方案存在一个可靠的 dealer，可将随机生成的共享秘密位 S_k 计算生成 n 个共享秘密片段，并使得任意 $f+1$ 个片段可以重建 S_k。秘密共享方案的引入确保了故障进程不能独立地计算共享秘密位，同时利用数字签名保证了接收到片段的真实性与正确性，还为协议提供了必要的随机性。

如图 2 所示，Toueg 率先将 echo_broadcast 与 compute_secret 作为组件，以模块化的思路构建共识协议，经过最多 r（独立于 n 和 f 的一个小常数）轮次的 2 阶段迭代（消息广播与接收、基于证明更新与广播）后达成一致。正确进程不会陷入死锁，并且在系统具有正确的初始值时，所有正确进程将以至少 $1-(1/2)^r$ 的概率达成共识，预期实现共识的轮次为 $1/(1-(1/2)^r)$。

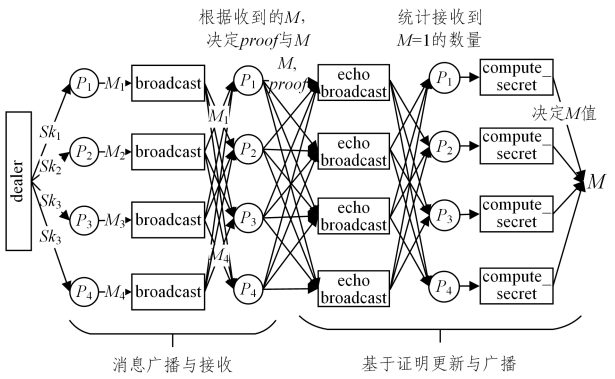


图 2　基于秘密共享的异步广播算法

Fig. 2　Asynchronous broadcast algorithm based on secret sharing

该协议将容忍故障节点数提升至 $f<n/3$，同时证明了在异步系统中，没有拜占庭共识协议能够容忍超过 $f<n/3$ 个故障进程[30]，从而确立了所提出异步共识协议的最优性。最后文献[28]提出了一种过滤器算法（Filter Algorithm），用于将多值消息简化为二元消息，再使用二元拜占庭协议来达成一致。

3.1.3　可靠广播

Bracha[31]设计了一种可靠广播（Reliable Broadcast，RBC），作者以可靠广播作为构建更复杂共识协议的关键组件，提出了一种异步拜占庭共识协议。该协议的容错能力为 $f<n/3$，达成共识的预期轮次数为 2^{n-f}，计算可得该协议的通信复杂度为 $O(n\times(n-1)\times 2^{n-f})$。虽然作者并未明确提及是否使用了密码学技术，但该协议中进程需能够识别接收到的消息来源，因此可推断该协议大概率使用了数字签名技术。此外，作者提出了该共识协议的属性，即二元性（Bivalence）、一致性（Consistency）、收敛性（Convergence），为后续研究者归纳总结共识协议的特性提供了参考。

3.1.4　门限密码与多值共识

2000 年，Cachin 等提出了一种异步二元拜占庭共识协议，并于 2005 年正式发表论文[32]。协议使用了非交互式门限签名方案和具有随机特性的门限抛币方案，率先将门限密码技术应用到异步共识中，通过门限签名、门限公钥加密与门限抛币技术增强了共识协议的鲁棒性与安全性。协议通过预处理（Pre-processing）、预投票（Pre-vote）、主投票（Main-vote）、决策检查（Check for decision）、共同抛硬币（Common coin）5 个步骤，确保即使在存在恶意行为的情况下，系统也能达成一致。共识过程如图 3 所示。

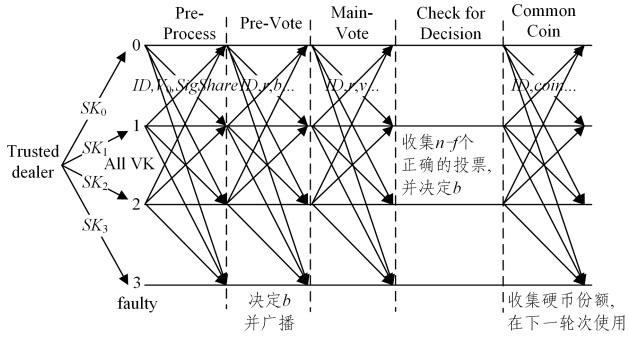


图 3　使用门限密码的异步拜占庭共识

Fig. 3　Asynchronous byzantine consensus with threshold cryptography

该协议的容错能力为 $f<n/3$，能在常数轮次内完成共识，即期望轮次复杂度为 $O(1)$，协议的消息复杂度为 $O(n^2)$，通信复杂度提升至 $O(n^{2k})$，其中 k 是 RSA 签名的长度。但 Cachin 等提出的协议使用了多种现代密码学技术，相比于不使用密码学的协议，其计算复杂度可能更高。

2001 年，Cachin 等[33]在曾经的研究成果基础上提出了 Protocol VBA（Validated Byzantine Protocol）与 Protocol VBAconst（Constant-round Validated Byzantine Protocol）两种多值拜占庭协议，使用了数字签名和门限密码方案等来解决异步环境中的多值共识问题。与单纯的二元共识协议不同，多值拜占庭协议通过循环进行二元拜占庭协议，逐步筛选

出被足够多诚实节点接受的提议值，这个值不限于二元(0或1)，可以是任意的数据。同时，相比传统的每个节点各自决定抛币值的方式，协议使用门限抛币方案，使得所有正确节点能够共同协商一个随机抛币值，大大加快了正确节点之间达成共识的速度。

两种共识协议的区别与特点如表1所列，其中 K 代表阈值签名和签名份额的长度，L 代表所有参与方提出的值和验证数据的总长度。

表1 Protocol VBA与Protocol VBAconst的对比

Table 1 Comparison between Protocol VBA and Protocol VBAconst

	Protocol VBA	Protocol VBAconst
提案阶段	每个参与方提出自己的值作为候选值	同Protocol VBA
排列选择	固定排列	通过阈值硬币抛掷随机生成排列
消息传递	发送和接收投票消息，对提案进行投票	可能省略在投票消息中包含提案的步骤
终止条件	决定提案为1时终止	结合了随机化和承诺机制，保证协议常数轮终止
承诺阶段	无	在协议循环前进行承诺交换，提交支持的提案列表
消息优化	无特定优化	提出了减少不必要消息传递的优化方案
容错能力	$f<n/3$	$f<n/3$
期望轮次	$O(n)$	$O(1)$
通信复杂度	$O(n^2(fK+L))$	$O(n^3+n^2(K+L))$

此外，Chandra等[34]提出了在崩溃故障的异步系统中共识和原子广播的等价性。在此结论之上，Cachin等提出了原子广播的定义与协议。原子广播可确保所有诚实的接收方以相同的顺序收到所有消息。

Cachin等还提出了协议的自适应腐败模型，即攻击者或故障的类型可以随着系统行为的变化而变化；最后，就如何抵御自适应腐败、通信复杂度的优化等一些未解决的问题指出了未来的研究方向。

3.1.5 无需签名的异步拜占庭二元共识

Mostefaoui等[35]提出了一种不依赖签名的异步拜占庭二元共识协议，该协议引入一种二元广播算法(Binary-Value Broadcast，BV-broadcast)，如图4所示。BV-broadcast是一种通信抽象，与之前的广播算法不同[36-37]，其专注于值本身，而不是广播它们的进程，这意味着不用关心某个特定进程是否广播了某个值，而是关注值本身是否被足够多的正确进程广播。同时，在BV-broadcast算法中，一个值只引发单一的“回声(echo)”，并且无需签名。简洁的设计不仅使该算法可在有限的预期轮次完成广播，而且降低了计算与通信的开销。

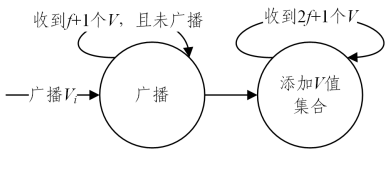


图4 二元广播算法

Fig. 4 Binary broadcast algorithm

作者以二元广播算法为组件，基于共同抛币值[24]设计了一种无需签名的异步拜占庭共识协议，协议中每个正确进程开始时提出一个初始值，然后进入重复的通信轮次。在每个轮次中，进程首先通过BV-broadcast广播自己的当前估计值，等待接收到足够多的其他进程的广播值后，再根据这些信息和公共硬币的结果更新自己的估计值。如果一个进程在一轮中确定所有其他正确进程都持有与其相同的估计值，并且公共抛币的结果与之一致，那么它将以这个值作为最终确定的值。该协议容错能力为 $f<n/3$，每个轮次使用2或者3步通信，完成共识的期望轮次为4，每轮次消息复杂度为 $O(n^2)$。该共识协议是目前为止最为高效的二元共识协议。

3.1.6 异步公共子集协议

Miller等借鉴了异步公共子集协议[38](Asynchronous Common Subset，ACS)的思想，提出了蜜獾拜占庭共识协议[39](Honey Badger BFT，HB-BFT)。这是一种完全异步的拜占庭共识协议，其关键部件ACS由可靠广播[40](Reliable Broadcast，RBC)和异步二元共识[35](Asynchronous Binary Agreement，ABA)构成，并使用门限加密方案实现弹性审查，使该协议不依赖于任何时间假设，能够以较低的通信开销实现共识。

如图5所示，在ACS中，每个节点首先利用RBC协议将本地提议通过门限加密后传播给网络中的其他节点。随后，每个节点针对每个RBC实例的成功与否，执行一个ABA实例。ABA实例的输出是一个位向量，指示对应的RBC实例是否被认为成功完成。具体来说，如果一个节点收到来自另一个节点的提议，则对应的ABA实例将输入设置为1，当一个节点收到足够多($n-f$)个节点的提议时，它将其他ABA实例的输入设置为0。ACS保证所有正确节点都得到相同的输出。

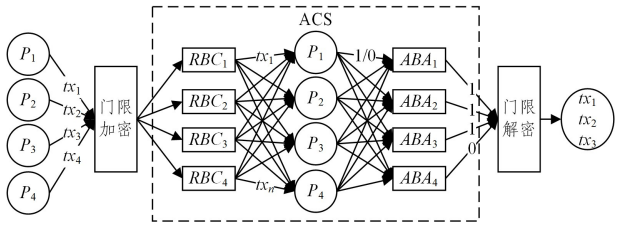


图5 蜜獾拜占庭共识协议

Fig. 5 Honey badger BFT

协议使用了门限加密、门限签名、纠删码与Merkle树等密码学技术，能够在异步网络环境中安全、高效地达成共识，即使面对拜占庭将军问题，也能保持鲁棒性。其通信复杂度并不是一个确定值，其最坏情况下的下界为 $O(\lambda n^2 \log n)$，其中 λ 是安全参数，其影响协议中密码算法的安全性，协议的预期运行轮次为 $O(\log n)$。

此外，Miller等还回顾了不同的网络模型，并构造了一个敌对的网络调度器。这个调度器违反了PBFT[17]的弱同步假设，并导致PBFT完全无法进展，但对于任何纯异步协议(如Honey Badger BFT)却能够取得良好的进展。这点表明，基于弱同步假设的协议在这些假设被违反时会失效，而纯异步协议则可以在这些条件下继续有效工作，提供更好的性能和鲁棒性。据称，Honey Badger BFT是第一种实用的异步拜占庭共识协议，但只能在节点身份已知且数量不可变的环境中运行。

Guo 等[41]基于 Honey Badger BFT，通过“先共识消息哈希，后请求缺失消息”，在消息长度远大于哈希长度的前提下，可实现消息传输效率的优化，但该优化方法并不适用于数字货币或数据量较小的分布式系统。

Guo 等[42]改进了 Honey Badger BFT 的 ACS，如图 6 所示，通过使用了一个委员会选举（Committee Election，CE）机制来随机选择常数个节点作为委员会成员，将 ABA 实例减少到一个与节点数无关的小常数 k，从而降低协议通信的复杂度。实验结果表明，与 Honey Badger BFT 相比，Dumbo1 在延迟和吞吐量方面都有明显改进。

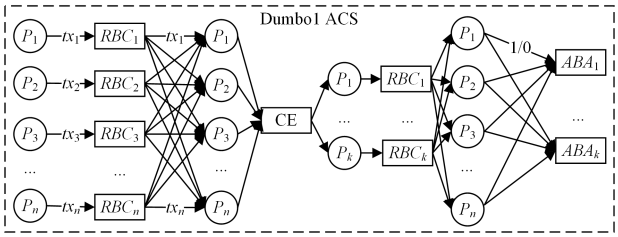


图 6　Dumbo1 异步公共子集协议

Fig. 6　Dumbo1 synchronous common subset

此外，Guo 等重新审视了多值拜占庭共识协议[32]（Multi-Value Validated Byzantine Agreement，MVBA）在 ACS 中的适用性，特别是当输入规模较小时，MVBA 的通信复杂性可能比 Honey Badger BFT 中的 ACS 更低，这与之前认为 MVBA 因通信复杂性高而不适用于 ACS 的观点不同。因此，作者基于可证明的可靠广播（Provable Reliable Broadcast，PRBC）、MVBA 与断言 Q（Predicate Q）设计了 Dumbo2，这里的 Predicate Q 其实是一种基于条件限制的异步共识协议组件。

Dumbo1 与 Dumbo2 的通信复杂度都是 $O(n^2|m|+O(\lambda n^3\log n))$，其中 $|m|$ 是消息的大小，λ 是密码学相关的安全参数。Dumbo1 预期达成共识的轮次为 $O(\log k)$，其中 k 代表运行 ABA 实例的数量，Dumbo2 理论上可以在常数轮次达成共识。与 Honey Badger BFT 类似，Dumbo1 与 Dumbo2 都使用了门限密码与数字签名技术，但 Dumbo1 不是直接的多值共识，而是对提议子集的二元决策。

Wang 等[43]在 Dumbo 的基础上提出了一种名为 PenguinBFT 的异步拜占庭共识协议，以期通过直接广播交易原文、引入节点信誉评估机制以及对网络节点进行分区等策略，解决现有异步共识算法中通信开销大和随机抽签算法缺乏信誉机制的问题。

3.1.7　基于有向无环图

Baird 等[44]提出的 HashGraph 协议使用有向无环图作为账本的数据结构，通过 Gossip 协议与虚拟投票（Virtual Voting）实现完全异步的拜占庭共识。文中提到几个关键概念：

事件（Event）：HashGraph 的基本数据结构，包含交易信息及通过哈希值与祖先事件关联，类似于区块链中的区块（Block），但与区块仅引用前一个区块不同，事件包含对两个祖先事件的引用。

可见（Seeing）：如果一个事件可通过一系列父事件连接到另一个事件，则称一个事件“可见”另一个事件。

强可见（Strongly seeing）：事件跨越大多数（超过 2/3）节点“可见”另一个事件，则称该事件强可见另一事件。

轮次（Round）：当一个事件强可见大多数先前见证者事件时，则称该事件在一个新的轮次上。

见证者（witnesses）：见证者是每一轮次中首先被创建的事件。

知名见证者（Famous witnesses）：当 R 轮的见证者事件被 $R+1$ 轮的多数见证者可见时，它就是知名见证人事件。

如图 7 所示，每个节点维护一个哈希图，并通过“Gossip about Gossip”协议随机选择节点将已知的事件进行传播，接收到事件的节点验证事件后将其同步到自己的哈希图，事件经过可见、强可见、轮次加 1、知名见证者等一系列过程逐渐“成熟”，完成事件提交，并实现共识，最终得到一个全局有序的事件列表。该协议中的虚拟投票主要是通过“强可见大多数见证者”与“大多数 $R+1$ 轮见证者可见 R 轮见证者”两个步骤来进行，前者类似于确定投票委员会成员，后者则是收集投票委员会对某个祖先的投票，如果投票数超过大多数，说明系统就该祖先事件达成共识。

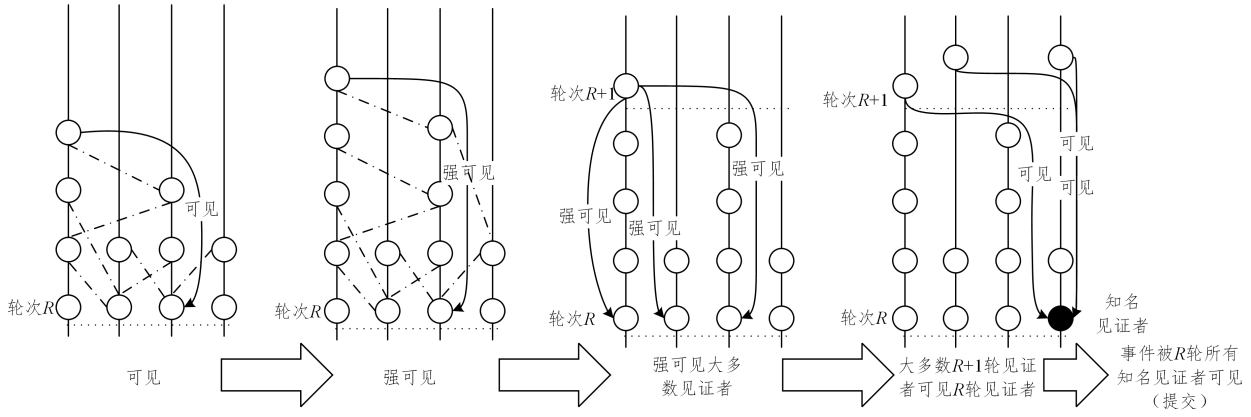


图 7　Hashgraph 共识过程

Fig. 7　Hashgraph consensus process

Hashgraph 使用了 Gossip 协议进行消息传播，因此其通信复杂度可近似为 $O(n^2)$[45]。同样，该协议使用了抛硬币来保证异步网络模型下协议的活性，通过增加一个随机轮次（Coin rounds），使用各节点数字签名的中间位进行投票，

使协议能够在常数轮达成共识。Hashgraph 完全异步，且无需领导者节点，有着较高的共识效率，但该协议仍存在空事件或交易长时间无法确认等问题。

综上可见随机化方法在解决异步拜占庭共识问题中的发展和应用。最初 Rabin 和 Ben-Or 提出基于随机化的概念，通过引入概率性决策来增强协议的鲁棒性；随着研究的深入，研究者们开始探索如何通过秘密共享、可靠广播、门限密码学、有向无环图等技术来提高协议的效率和安全性。但随机化方法导致了共识协议的终止是概率性的，在一定程度上降低了共识问题的要求[46-47]。

3.2 部分同步模型

Dolev 等[48]探讨了在分布式系统中实现共识所需的最小同步性。Dwork 等[49]率先提出了部分同步的概念，并给出了两种部分同步模型。1)未知固定延迟上限模型：分布式系统中的消息传递存在一个固定的延迟上限，但这个上限值对于系统参与者来说是未知的。2)延迟上限已知但起始时间未知模型：此模型规定了一个已知的通信延迟上限，但这个上限只在系统经过全局稳定时间点之后才开始生效。Chandra 等[34]给出了第三种模型，其是对 Dwork 所提出模型的补充。该模型指出系统最初可能表现为异步，但经过足够长的时间后，系统会达到一个稳定的状态。

部分同步模型的核心是通过时间假设的引入，使系统的同步性逐渐呈现。在部分同步模型的基础上，更多的学者开展了异步共识协议的研究，本文选择其中比较有代表性的工作进行研究与阐述。

3.2.1 领导者与视图

Brian 等[50]提出了一种名为 VR(Viewstamped Replication)的共识协议。该协议采用主-备份副本方法，其中主副本负责按顺序处理客户端请求，而备份副本则接收并执行主副本的指令，在主副本失效时，系统通过一个确定的视图更换协议来选举新的主副本。该协议常用于分布式数据库的复制。

Abraham 等[51-52]在 Cachin 等的研究基础上，提出了一种可验证异步拜占庭共识协议。该协议基于视图的概念，每个视图包含 3 个阶段。1)广播阶段(Broadcast Phase)：该阶段提出了 4-Stage-f+1-Provable-Broadcast，顾名思义，在最多 f 个拜占庭节点存在的情况下，通过 4 个阶段的通信过程，确保至少有 f+1 个诚实节点能够证明消息被正确广播的机制。2)领导者选举阶段(Leader-election Phase)：该阶段基于随机预言机的领导者选举机制，决定哪个节点将引导当前视图进行共识。领导者选举是一个创新的解决方案，它通过引入随机性来提高系统的鲁棒性和安全性，但并不解决 FLP 问题。3)视图变更阶段(View-change Phase)：该阶段受到 Yin 等[53]的启发，基于部分同步模型，在领导者广播未完成、未能达成共识或超时等情况下，进行视图变更，以保证系统的活性，并达到 O(1)的时间复杂度。

与 Cachin 等在 2001 年提出的方案相比，该协议不依赖随机二元值，基于部分同步模型，通过随机领导者选举原语来实现共识，解决了 Cachin 等提出了近 20 年的开放式问题，将通信复杂度从 $O(n^3)$ 优化到了 $O(n^2)$，同时提供了最优的容错能力与 O(1)的时间复杂度。

3.2.2 基于可信增加容错

文献[15]证明了如果一个节点不能够针对同一提案给不同的节点发送不同的提案，拜占庭缺陷将会简化为宕机缺陷，系统的最大容错能力将提升至 $f<n/2$。

Veronese 等[54]重点关注了异步拜占庭共识协议中如何增加拜占庭节点的容错比例，设计了 USIG(Unique Sequential Identifier Generator)信任服务，为每个客户端的提案分配一个唯一的、递增的序列号，并为该序列号生成一个认证证书。这个序列号确保了即使在异步网络中，操作也能按照一致的顺序执行，因此可提升该协议容错能力至 $f<n/2$。文中提出了基于哈希(USIG-Hmac)与基于公钥密码的(USIG-Sign)实现逻辑，并给出了 3 种不同的 USIG 服务架构，即非安全 USIG(NS-USIG)、基于虚拟机的 USIG(VM-USIG)、基于 TPM(Trusted Platform Module)的 USIG(TPM-USIG)，如图 8 所示。

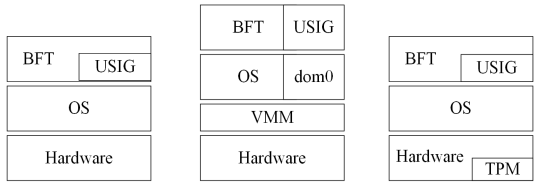


图 8 3 种不同的 USIG 架构

Fig. 8 Three different USIG architectures

Veronese 等利用 USIG，基于部分同步模型与 PBFT[17]共识协议，提出了两种异步拜占庭共识协议：MinBFT 和 MinZyzzyva。

MinBFT 消息交换模式类似于 PBFT，但减少了 PBFT 中的 Pre-prepare 阶段，并简化了进程间交互的复杂度。PBFT 与 MinBFT 协议的流程如图 9 所示。

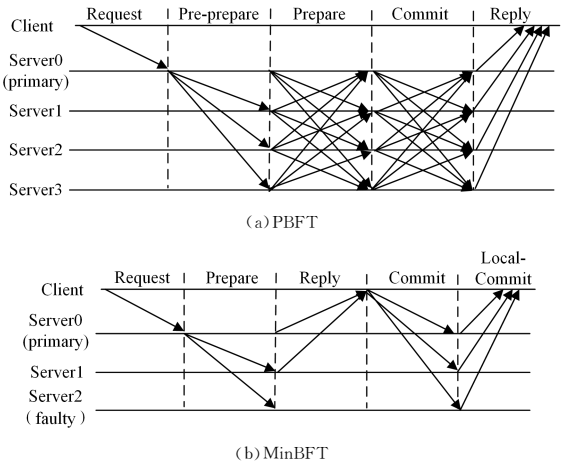
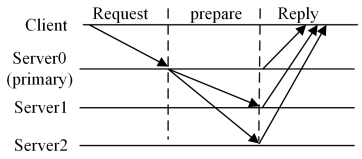


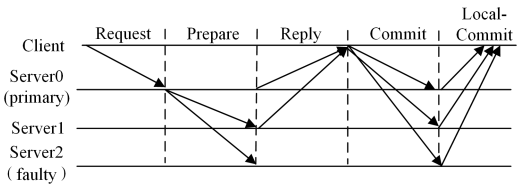
图 9 PBFT 与 MinBFT 的对比

Fig. 9 Comparison of PBFT and MinBFT

MinZyzzyva 是一种推测性的拜占庭共识协议，它允许系统在没有预先达成一致的情况下，根据主节点提出的顺序，乐观地执行操作并立即回复客户端，但在预定时间内没有收到足够多的响应时，发送 Commit 消息给所有节点，请求确认已执行的操作，确保系统的容错性和活性。MinZyzzyva 协议的乐观与非乐观执行流程如图 10 所示。



(a)乐观执行



(b)非乐观执行

图 10　MinZyzzyva 乐观与非乐观执行的对比

Fig. 10　Comparison of optimistic and pessimistic executionin MinZyzzyva

MinZyzzyva 相较于 MinBFT 协议，在乐观情况下，减少了通信步骤，在网络延迟较高的情况下能提供更好的性能和更低的延迟。但由于 MinZyzzyva 需要处理推测和执行可能的回滚，编程模型更复杂，两种协议的通信复杂度均为 $O(n)$，且都可在常数轮达成共识。此外，因依赖于 USIG，两者均可达到 $f<n/2$ 的容错能力。

基于部分同步模型的“异步”共识协议，通常依赖于特定的时间假设，但这种假设可能在实际的分布式系统中难以满足，会导致共识协议性能的下降，甚至无法进展。从一定程度上讲，基于部分同步模型的共识协议不应称为异步共识协议。另外，部分同步模型的协议设计通常比纯异步协议更复杂，需要处理更多的边界情况和异常，相较于纯异步的共识协议，鲁棒性与适应性都不佳。

3.3　故障检测器

在分布式系统中，无法判断一个节点是因为故障还是仅仅因为运行缓慢而没有及时响应。这种不确定性是导致 FLP 不可能性结论的关键因素之一。而故障检测器最初就是用来探测一个节点的状态，如果一个系统能够利用故障检测器来识别出哪些节点已经失效，即使只是暂时的，这种信息也可以用来规避 FLP 不可能性结论。因此，故障检测器作为解决异步共识的重要技术之一[55]，已经受到广泛的关注[56-57]。

根据节点的故障类型，故障检测器可分为崩溃(宕机)故障检测器和拜占庭故障检测器。崩溃故障检测器依赖于超时机制，包括基于心跳的故障检测器[58]、all-to-all[59-60]故障检测器、层次式[61-62]和使用 Gossip 协议[63]的故障检测器。拜占庭故障检测器通常使用状态机来监测节点的行为模式，使用密码学来验证消息的正确性、完整性[64]，以检测部分可识别的拜占庭故障。

故障检测器的设计方法虽然较多，但从实现的角度出发，可分为两类。

1)基于心跳的故障检测器[65]：被检测进程通过周期性地发送“心跳”消息表明自己仍然存活，如果检测进程在预定的时间内没有收到某个其他进程的心跳，它会怀疑该进程可能已经崩溃。该方法实现简单，“心跳”可承载额外的信息，但“心跳”会产生更多的周期性消息。

2)基于轮询的故障检测器：通过主动询问其他进程的状态来检测故障。与心跳方式相比，轮询由检测器进程主动发起，可以通过调整轮询频率来平衡检测的敏感性和系统负载，但轮询会引入更多的网络通信开销。

3.3.1　Paxos

1989 年，Lamport 提出 Paxos 协议[66]，通过类比古希腊 Paxos 岛上的议会制度，基于异步通信及崩溃容错，设计了一种分布式共识方法。该方法通过立法者之间发送消息来确定议会主席，一个立法者在约定时间内向其他立法者至少发送 1 次他的名字，并且如果一个立法者在约定时间内没有收到任何“权限更高”的立法者的消息，那么他就认为自己是议会主席，议会主席将负责按照议会协议的要求，及时地发起投票。Paxos 分为准备与接受两阶段，主席至少需要与 $n/2+1$ 个立法者进行通信，因此该协议的消息复杂度可表示为 $O(n)$。此外，Paxos 协议假设了信道的可靠，在实际实现中常使用数字签名来保障消息的正确性。虽然 Paxos 协议晦涩难懂，但其作为经典的共识协议，引出了大量的变种与改进研究[67]。

该文中并未明确提及 Paxos 协议使用了故障检测器，但 Paxos 就是一种基于 Ω 故障检测器的共识协议[68]。Ω 故障检测器是一种理想化的故障检测器模型，可以帮助设计者构建能够容忍进程故障的分布式协议。

在实际的分布式系统中，构建完美的 Ω 故障检测器是非常困难或不可能的，因为网络延迟、分区故障、进程崩溃等因素都会影响故障检测的准确性和及时性。然而，Ω 故障检测器作为一个基准，可以为设计者理解和评估他们的系统在面对故障时的行为和性能提供参考。

3.3.2　分布式系统中的故障检测器

1996 年，Chandra 等[34]使用故障检测器向系统内的进程提供关于其他进程是否可能已经崩溃的信息，用以解决异步系统中的分布式共识问题。但故障检测器所提供的信息可能是不准确的，因此作者提出了不可靠故障检测器(Unreliable Failure Detectors)的概念，并根据准确性和完整性对故障检测器的类型进行划分，如表 2 所列。

同时，Chandra 等证明了◇S 是解决异步系统共识问题的最弱故障检测器[69]。◇S 类故障检测器应满足：1)强完整性，即任意发生宕机的进程最终都会被正确进程加入到怀疑列表中；2)最终弱精确性，即至少存在一个正确进程，在某个时间点之后，它不会被系统中的任何其他正确进程怀疑。此外，文中还指出了◇W 与◇S 类故障检测器的能力是相当的。

表 2　故障检测器分类

Table 2　Classification of failure detectors

完整性	准确性			
	强	弱	最终强	最终弱
强	P	S	◇P	◇S
弱	Q	W	◇Q	◇W

Kihlstrom 等[70-71]定义了两类拜占庭故障检测器：最终弱拜占庭故障检测器◇W(Byz)与最终强拜占庭故障检测器◇S(Byz)。这两类故障检测器被定义可解决拜占庭缺陷。但

作者并没有解决算法和拜占庭故障检测器之间循环依赖的问题，没有讨论拜占庭故障检测器实现的正确性。有学者指出[72]，除非将算法集限制为具有一定通信规则的算法，否则即使在完全同步的模型中，拜占庭故障检测器最终也可能无法实现。

3.3.3 故障检测器在异步共识协议中的应用

1)基于 quorum 的通用共识协议

在 Chandra 等[34]研究的基础上，Mostefaoui 等[73]提出了一种可适应宕机缺陷的异步共识协议，该协议基于协调进程和 S/◇S 类故障检测器实现。系统中每个进程都配备了一个故障检测器模块，同时，每个进程根据故障检测器提供的信息动态地维护了一个议员(quorum)集合，用于在每一轮中确定何时收集到足够的信息以做出决策。协议中每一轮次分为两个通信阶段：第一阶段，轮次的协调进程将当前的估计值广播给所有其他进程；第二阶段，所有进程尝试收集足够的信息来决定一个值，或者确定需要进入下一轮。如果一个进程在一轮中决定了一个值，它会使用可靠广播机制将该决策值传播给其他所有进程。

故障检测器提供的怀疑列表可以帮助协议处理进程的崩溃情况，确保协议的活性，同时在存在故障进程的情况下，故障检测器可以减少不必要的消息传播。该协议仅针对崩溃容错，每轮的通信开销为 $O(n^2)$，如果使用 S 类故障检测器，协议保证在最多 n 轮之后达成共识。对于◇S 类故障检测器，文中并未明确说明一个固定的轮次上限。

2)零退化共识协议

Dutta 等[74]提出了一种基于 Ω 类故障检测器的异步共识协议，该协议在没有进程崩溃并且故障检测可靠的情况下，通过两个通信步骤就能完成共识；同时，该协议具有零退化(Zero Degradation)特性，即使在某些故障发生的情况下，协议的性能也不会受到影响。作者还讨论了在保持零退化特性的情况下，如何将 Ω 类故障检测器转换为◇S 类故障检测器。

3)基于◇S 类故障检测器的异步共识

Malkhi 等[75]借鉴了 Paxos 协议轮值主席的思想，使用◇S 类故障检测器与可靠广播，提出了一种异步拜占庭共识协议。该协议以轮次进行，每一轮都有一个领导者，所有进程在开始时持有一个输入值。领导者收集来自其他进程的值，一旦收集到足够多的相同值，就提议一个共识值，其他进程对提议的值进行响应，发送确认(ACK)或拒绝(NACK)消息。如果一个进程收到超过一半的 ACK 响应，它就接受提议的值，并进入下一轮。整个协议进行过程中，故障检测器扮演了关键的角色。

(1)怀疑故障进程：故障检测器允许进程怀疑其他进程可能已经发生故障。

(2)领导者选择和消息广播：故障检测器的信息会影响领导者的选择和广播消息的有效性。

(3)收集响应和确认：领导者在收集到一定数量的格式良好的消息后，会选择一个共识值并广播提议。其他进程根据故障检测器的输出来决定是发送 ACK 还是发送 NACK。

(4)决定协议终止：进程使用故障检测器来确定何时可以安全地终止协议。如果一个进程收到足够的 ACK 响应，并且故障检测器表明没有其他进程被怀疑，那么它可以决定共识值并终止协议。

(5)非格式良好的消息(non-well-formed message)检测：如果进程接收到非格式良好的消息，它会根据故障检测器的状态来决定是否将消息发送者添加到怀疑列表中。

4)静默故障检测器

Doudou 等[72]扩展了故障检测器的方法，从处理崩溃故障扩展到可处理静默故障。静默故障是一种恶意故障，进程停止发送特定算法消息，但可能继续发送其他消息。静默故障检测器◇M_A用于检测那些停止发送算法消息的进程，它通过监测进程是否继续发送“I-am-alive”或其他非算法消息来确定进程是否发生了静默故障。

作者定义了静默故障检测器的属性。

(1)最终静默完备性(Eventual Mute Completeness)：存在一个时间点，在这个时间点之后，对于算法 A，每一个对正确进程 p 静默的进程将被进程 p 永远怀疑。

(2)最终弱准确性(Eventual Weak Accuracy)：存在一个时间点，在这个时间点之后，对于算法 A，任何其他正确进程将不再怀疑正确进程 p 是静默的。

图 11 展示了静默故障检测器的实现逻辑，它依赖于超时机制，由 3 个并发任务组成。每个进程 p 初始化一个超时值 Δp，同时初始化两个空集合：一个用于存储进程 p 怀疑已经发生静默故障的进程列表 $output\_p$，另一个用于存储当前轮次中的关键进程列表 $critical\_p$。

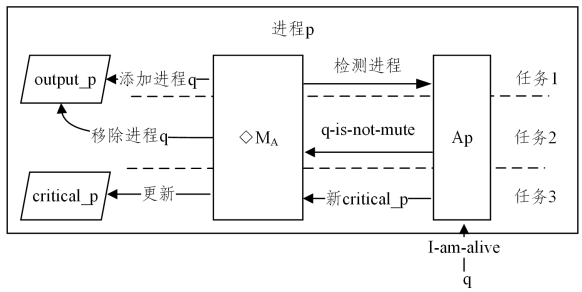


图 11 静默故障检测器

Fig. 11 Silent failure detector

任务 1 怀疑进程

在每个轮次中，进程 p 会检查 $critical\_p$ 集合中的每个进程 q。如果进程 q 不在 $output\_p$ 集合中，并且在 Δp 时间内没有收到来自进程 q 的“I-am-alive”消息，那么进程 p 会怀疑进程 q 发生了静默故障，并将 q 添加到 $output\_p$ 集合中。

任务 2 接收消息

当 A_p(算法的本地实例)收到来自某个进程 q 的“I-am-alive”消息时，它会向◇M_A发送一个“q-is-not-mute”消息，如果 q 已经在 $output\_p$ 集合中，那么进程 p 会将 q 从 $output\_p$ 集合中移除，表示进程 q 目前没有发生静默故障。

任务 3 更新关键进程集合和超时值

在每个新的轮次开始时，进程 p 会收到一个更新后的 $critical\_p$ 集合，其中包含该轮次的关键进程列表。同时，进程 p 还会根据某个函数计算一个新的超时值。

文献[72]同时结合 Early Consensus[76]，给出了一个与上述静默故障检测器结合的共识协议的例子。最后，指出并非所有的共识协议都适合使用静默故障检测器，特别是那些在完全同步的系统中也无法区分静默进程和正确进程的情况，使用静默故障检测器是没有意义的。

Friedman 等[77]依赖简单设计的原则，设计了一种基于静默故障检测器的共识协议，该协议消息复杂度为 $O(n^2)$，除了携带的轮次编号之外，任何消息的大小都是 O(1)，且不需要数字签名等技术，能确保在有利情况下执行单个通信步骤即可终止。虽然该协议可容忍的故障节点数量（$f<n/6$）不是最优的，但该协议简单且高效。

5）拜占庭故障检测器

Baldoni 等[78]结合静默故障检测器和拜占庭行为检测器，提出了一种异步拜占庭向量共识协议。其中，静默故障检测器与 Doudou 等[72]所提出的一致；拜占庭故障检测器则是基于有限状态自动机（Finite State Automata，FSA）实现，通过检查收到的消息和证书来检测拜占庭行为，如果消息类型不正确、消息发送的时机不对或证书验证失败，则有限状态自动机将标记该进程为拜占庭进程。有限状态自动机定义了如下状态：

start：初始状态。

q_0，q_1，q_2：一个共识轮次中不同阶段的状态。

Byz：拜占庭状态。

final：最终状态。

图 12 展示了各状态的转换逻辑。

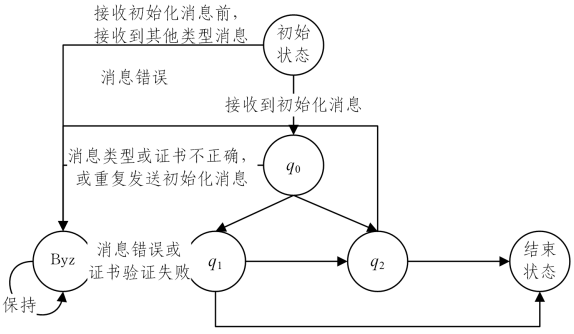


图 12　有限状态自动机

Fig. 12　Finite state automaton

文献[78]同时结合了两种故障检测器、数字签名技术及共识模块，描述了分布式系统中一个进程的结构。如图 13 所示，每个进程中 5 个模块协作以达成共识。该结构有助于提高系统的模块化设计，增强系统的灵活性。

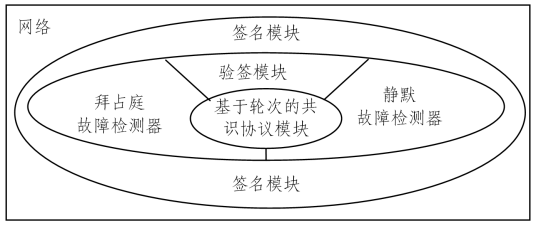


图 13　进程结构

Fig. 13　Process structure

虽然 Baldoni 等设计了一种拜占庭故障检测器，但该检测器无法检测到所有类型的拜占庭行为，特别是一些高级欺骗行为，例如模拟多个身份发送矛盾信息或者是对进程内部状态的篡改，故障检测器完全检测所有的拜占庭行为是不可能的[15,17]。

故障检测器是用于解决异步环境下共识问题的又一重要途径。通过对检测器的抽象定义，可以将焦点集中在协议的逻辑和行为上，降低了共识协议的设计难度，并使共识协议的证明更加容易。故障检测器的研究已经从解决基本的异步共识问题发展到减少必要的通信轮次数，并逐步扩展到了能够处理包括静默故障和部分拜占庭故障在内的更广泛的故障类型，这也反映了分布式系统领域内对提高系统鲁棒性和性能的持续追求。但异步环境的本质使得我们无法构建完美的故障检测器，实用的故障检测器几乎都依赖时间假设，本质上是对系统行为的时间特性进行假设和扩展。

3.4　基于条件限制

除上述规避 FLP 的方法外，还有研究者通过弱化共识定义，基于条件限制的方法来解决异步系统中的共识问题。这些条件包括输入向量的限制、层次化的条件限制以及纠错码理论的应用等，通过限制输入向量集合，减少系统中的不确定性，并在这些限制条件下寻找解决共识的可能性，从而为解决异步共识问题提供了新的解决思路。

3.4.1　限制输入向量

Mostefaoui 等[79]设计了一种基于条件限制解决异步共识的方法，阐述了“可接受条件”的概念：如果存在一个共识协议能够解决特定条件下的共识问题，那么该条件被称为可接受的。同时，提出了两个具体的条件 C1 和 C2，并且在证明它们是可接受的同时，还证明了任何尝试的扩展都会得到一个不可接受的条件。

条件 C1：给定一个输入向量 $\boldsymbol{I}$，出现次数最多的值与出现次数次多的值之间的差距大于 f。

条件 C2：输入向量中最大的值出现的次数大于 f。

文献[79]通过定义输入向量的条件集，来确定在哪些情况下可以解决共识问题，通过断言 P（Predicate P）判断当前的输入向量是否属于可接受条件，并牺牲一定的安全性来保证部分节点发生崩溃时正确进程总是终止。

在之前的研究基础上，Mostefaoui 等[80]于 2004 年又提出了一个更为深入和系统化的条件层次结构理论，为设计针对特定条件的共识协议提供了一个框架。该框架可根据输入向量满足的条件强度来调整协议的开销与成本。

3.4.2　纠错码与一致性问题

Friedman 等[81]研究了 3 种相关的一致性问题：共识（consensus）、交互一致性（interactive consistency）和 k-集合一致性（k-set agreement）；并基于条件方法（condition-based approach）分析，证明了在异步分布式环境中，即使面对进程崩溃，也能够设计出解决共识问题的协议。

Friedman 等建立了纠错码与分布式共识问题之间的桥梁，使用纠错码检测与纠正接收信息的错误以实现共识。此外，指出了输入向量的判定条件和纠错码之间的关系，即进程

崩溃对应于纠错码中的擦除错误，拜占庭故障和值域故障对应于纠错码中的腐败错误。

文献[81]还展示了可以使用限制性较小的纠错码来解决k-集合一致性问题，但尚未证明使用纠错码的必要性，这仍是一个开放性问题。在后续的研究中[82]，研究者展示了如何利用纠错码的容错能力来设计能够容忍拜占庭故障的共识协议，同时提出了一些具体的编码方案，用以解决或近似解决分布式共识问题。

采用条件限制的方法为设计异步共识协议提供了新的视角，在满足可接受条件的前提下，共识协议能够确保最终达成一致的决策。基于条件限制解决异步共识的研究虽不多，但给研究者们提供了一个新的方向。

3.5 混合共识

研究者们也尝试了使用上述多种途径相结合的方式来共同解决异步共识的问题[83]，使共识协议具有更强的鲁棒性与适应性。

Malkhi 等[75]在提出基于◇S 类故障检测器的异步共识协议基础上，还提出了一种结合了故障检测器和随机化技术的混合共识协议，允许故障检测器表现良好时在常数轮通信的情况下达成共识。但由于领导者需使用故障检测器不间断地与其他节点维持心跳，因此其消息复杂度至少为 $O(n)$。此外，在故障检测器的弱准确性不能保证的情况下，协议也能够以一定的概率成功终止，但其消息复杂度将达到 $O(n^2)$。这种方法结合了确定性（故障检测器）和概率性（随机化）的优点，提高了协议在面对不确定性和潜在的拜占庭行为时的可靠性。

Aguilera 等[84]提出了一种混合共识方法，该方法与 Malkhi 等提出的协议的思想一致，通过结合故障检测和随机化机制，增强了共识协议的鲁棒性。当故障检测机制运行正常时，协议能够确定性地终止，如果故障检测不准确，协议仍然能够以一定概率成功终止。

Mostefaoui 等[85]提出了包含 3 个阶段的混合二元共识协议，如图 14 所示。

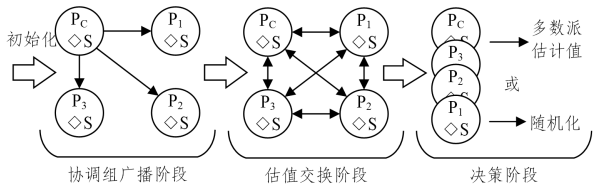


图 14 混合二元共识协议

Fig. 14 Hybrid binary consensus protocol

1）协调者广播阶段：轮次的协调者根据故障检测器的信息广播其当前的估计值给所有进程。在这一阶段，故障检测器提供了一个进程怀疑列表，其中包含它认为可能已经崩溃的进程。协调者在广播前会检查自己是否被怀疑，如果没有被怀疑，它会广播自己的估计值；如果被怀疑，其他未被怀疑的进程可能会成为新的协调者。

2）估计值交换阶段：每个进程将它们当前的估计值发送给其他所有进程，进程根据收到的估计值信息，确定是否存在一个多数派意见。如果某个估计值被超过一半的进程持有，它将成为这一阶段结束时的多数派估计值。

3）决策阶段：决策阶段结合了随机化和故障检测器来推动共识的达成。如果故障检测器指示所有进程都是活跃的，并且没有新的崩溃发生，那么进程可以依赖于收到的多数派信息来做出决策。如果进程在第二阶段结束时没有收到明显的多数派信息，协议将使用随机化来使系统达成共识。

Mostefaoui 等强调了共识协议的模块化实现方法，通过删除协议中的特定部分，可以将其转换为纯随机化共识协议或基于故障检测器的共识协议。例如，删除第 1 阶段可以得到 Ben-Or 的随机化协议[25]；删除第 2 阶段和第 3 阶段的随机化部分可以得到基于 Chandra 等不可靠故障检测器的共识协议[73]，同时可通过修改第 1 阶段，设计出新的混合二元共识协议。最后，作者进一步使用马尔可夫链模型来模拟协议的行为，并计算了在不同条件下达成共识所需的平均轮数，还讨论了共享长随机比特序列与早期决策两种优化策略。

混合共识为解决异步共识问题提供了一种平衡，其既能在故障检测器可靠时快速达成共识，也能在故障检测器不准确时通过随机化方法保证最终能够达成共识，结合了确定性和概率性方法的优点，提高了系统在面对不确定性时达成共识的能力。

4 总结

4.1 异步共识算法比较

异步共识协议的发展脉络如图 15 所示。从 1985 年 FLP 结论诞生以来，研究者们不断以随机化、部分同步模型、故障检测器、基于条件限制以及混合共识的方法提出“宽容性异步共识协议”[86]，以规避 FLP 不可能结论，寻求异步环境中分布式共识的“工程解”，并基于前人的成果，在性能优化、共识协议模块化、与密码算法结合，以及使用新型的数据结构等方面进行尝试，共同推动异步共识协议的发展。

表 3 根据避免 FLP 策略的不同，分别列出了上述异步共识算法的特点与性能，从容错率、通信复杂度、预期轮次、密码算法与共识类型等方面进行了量化分析与对比。

在采用随机化的异步共识协议中，有着较理想的通信复杂度与预期轮次的多为二元共识，这限制了其在分布式系统不同场景的适应性。但具有多值共识能力的协议，通信复杂度都较高，在节点数量增加后，易产生广播风暴，极大地了降低共识的效率。采用有向无环图技术的 HashGraph 共识协议则表现出了较高的共识效率，通过 Gossip 传播消息与“虚拟投票”的思想为异步共识的设计提供了又一思路。随机化[87-89]方法通过引入概率性决策增强了算法的鲁棒性，不依赖时间假设，能够在完全异步环境中工作，同时可在常数轮完成共识。然而，这种方法牺牲了确定性，共识的终止变得具有概率性，这在某些应用场景中是不可接受的。

基于部分同步模型的异步共识协议通常采用选取领导节点的方式实现。领导节点可以快速收集和汇总其他节点的信息，然后广播到网络中，这样可以减少节点间需要交换的消息数量，使系统可以快速决策。该类协议有着理想的通信复杂

度以及常数级的共识预期轮次，但在领导节点故障或频繁进行视图切换时系统通信开销会增加。此外，部分同步模型[44]是基于系统最终会变得同步的假设提出的一种折中的共识协议设计思路，基于部分同步模型设计的共识协议在同步网络下可完全满足共识问题的定义，但如果网络不稳定，协议的活性将无法保证。这并不是真正意义上的异步共识协议。

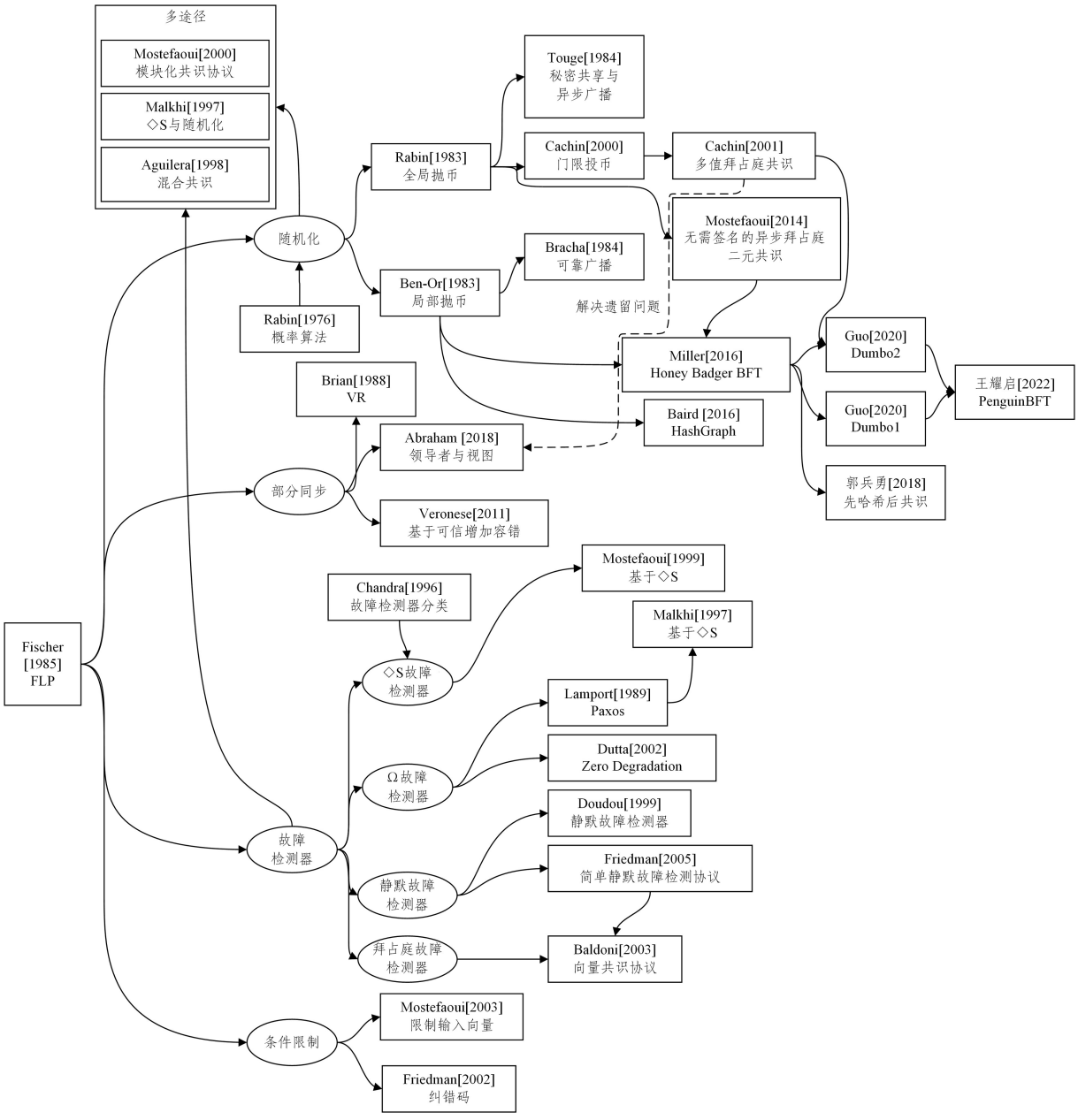


图 15　异步共识协议的发展脉络

Fig. 15　Evolutionary trajectory of asynchronous consensus protocols

故障检测器[90-91]是另一种解决异步共识问题的方法，它通过探测节点的状态来辅助共识过程。故障检测器的优势是算法高效，但拜占庭故障检测器无法检测所有拜占庭错误，且准确的故障检测在异步环境中是不可能的。此外，故障检测器和系统增加时间假设是等价的，难以适应完全异步的网络环境。

基于条件限制的方法[79-80]通过弱化共识定义或限制输入向量来解决共识问题，在满足特定条件时，可以设计出更高效的共识协议，具有更强的容错能力。但基于条件限制的方法在某些情况下，为了提高效率，可能需要牺牲一定的安全性，例如在条件不满足时允许决定默认值。同时，特定条件共识协议限制了协议的适用范围，需要事先知道或估计可能的输入条件，难以应用到更广泛的故障模型或不同的分布式系统应用场景。目前，基于条件限制的异步共识协议仍然处于理论阶段且鲜有研究。

混合共识[84-85]通常结合故障检测器与随机化，既能在故障检测器可靠时快速达成共识，也能在故障检测器不准确时通过随机化手段保障最终能够达成共识，使系统具有较高的性能，同时可确保活性。但混合共识协议一方面比单一的协议更为复杂，需要在两种策略间切换，增加了系统的计算与通信复杂度；另一方面，在故障检测器不可靠时，协议通过随机化完成共识，这同样带来了随机化异步共识所面临的问题。

表 3 异步共识算法的比较

Table 3 Comparison of asynchronous consensus protocols

避免 FLP	优点	缺点	算法	拜占庭容错	容错率	通信复杂度	预期轮次	密码算法	共识类型
随机化	设计简单，鲁棒性较强，不依赖时间假设	牺牲确定性，共识性能波动，难以预测	Rabin[1983] 全局抛硬币	是	$f<n/10$	$O(n^2)$	$O(1)$	秘密共享 数字签名	二元共识
			Ben-Or[1983] 局部抛硬币	是	$f<n/5$	$O(n^2)$	$O(1)$	数字签名	二元共识
			Touge[1984] echo_broadcast	是	$f<n/3$	$O(n^2)$	$1/(1-(1/2)^r)$	秘密共享 数字签名	二元共识
			Bracha[1984] 可靠广播	是	$f<n/3$	$O(n\times(n-1)\times 2^{n-f})$	2^{n-f}	数字签名（可能）	二元共识
			Cachin[2000] 门限抛币	是	$f<n/3$	$O(n^2)$	$O(1)$	数字签名 门限密码	二元共识
			Cachin[2001] VBA	是	$f<n/3$	$O(n^2(fK+L))$	$O(n)$	数字签名	多值共识
			Cachin[2001] VBAconst	是	$f<n/3$	$O(n^3+n^2(K+L))$	$O(1)$	数字签名 门限密码	多值共识
			Mostefaoui[2014] 无需签名的共识	是	$f<n/3$	$O(n^2)$	$O(1)$	无需密码算法	二元共识
			Miller[2016] Honey Badger BFT	是	$f<n/3$	$O(\Omega\lambda n^2\log n)$	$O(\log n)$	数字签名 门限密码 纠删码 Merkle 树	多值共识
			Guo[2020] Dumbo1	是	$f<n/3$	$O(n^2\lvert m\rvert+O(\lambda n^3\log n))$	$O(\log k)$	数字签名 门限密码	多值共识
			Guo[2020] Dumbo2	是	$f<n/3$	$O(n^2\lvert m\rvert+O(\lambda n^3\log n))$	$O(1)$	数字签名 门限密码	多值共识
			Baird[2016] Hashgraph	是	$f<n/3$	$O(n^2)$	$O(1)$	数字签名 哈希函数	多值共识
部分同步模型	与真实网络环境相似，完全满足共识问题的属性	不适用完全异步的网络环境，协议可能无法进展或终止，增加了协议的不确定性与安全挑战	Brian[1988] VR	否	$f<n/2$	$O(n)$	$O(1)$	未提及	多值共识
			Abraham[2018] 可验证异步拜占庭共识	是	$f<n/3$	$O(n^2)$	$O(1)$	门限密码	多值共识
			Veronese[2011] MinBFT	是	$f<n/2$	$O(n)$	$O(1)$	数字签名	多值共识
			Veronese[2011] MinZyzzyva	是	$f<n/2$	$O(n)$	$O(1)$	数字签名	多值共识
故障检测器	可提供确定的终止保障，使系统增强容错能力，并做出更好的决策	存在漏报和误报，无法检测所有拜占庭故障，同时会增加系统的通信开销	Lamport[1989] Paxos	否	$f<n/2$	$O(n)$	$O(1)$	数字签名	多值共识
			Mostefaoui[1999] 基于 quorum 的通用共识	否	$f<n/2$	$O(n^2)$	$O(1)$	数字签名	多值共识
			Dutta[2002] 零退化共识	否	$f<n/2$	$O(n)$	$O(1)$	数字签名	多值共识
			Malkhi[1997] 基于◇S 类故障检测器的共识	否	$f<n/2$	$O(n)$	$O(1)$	数字签名	多值共识
			Friedman[2005] 基于静默故障检测器的共识	否	$f<n/6$	$O(n^2)$	$O(1)$	无需密码算法	多值共识
			Baldoni[2003] 基于拜占庭故障检测器的共识	不能完全容错	$f<n/3$	$O(n^2)$	$O(1)$	数字签名	多值共识
基于条件限制	可通过定义条件，提高协议效率，增强容错性	特定条件共识协议限制了协议的适用范围，难以应用到更广泛的故障模型或不同的分布式系统架构	Mostefaoui[2003] 限制向量输入	否	未明确	$O(n)$	$O(1)$	未提及	多值共识
混合共识	通常可采用模块化设计，有较高的系统鲁棒性与性能，具备零退化特点	增加协议的计算、通信复杂度，综合了各途径的缺点	Malkhi[1997] ◇S 与随机化	是	未明确	$O(n)$ 或 $O(n^2)$	$O(1)$	数字签名	二元共识
			Aguilera[1998] 混合共识	否	$f<n/2$	$O(n^2)$	$O(1)$	未提及	二元共识
			Mostefaoui[2000] 混合二元共识	否	$f<n/2$	$O(n^2)$	$O(1)$	未提及	二元共识

大部分的共识协议都采用数字签名技术实现可信的通信信道，保障交互数据的可验证性，特别是针对拜占庭容错的共识协议，如果拜占庭节点能够伪造其他节点消息，那么拜占庭问题将变得难以解决。此外，也有部分共识协议利用秘密共享或门限密码技术来提升共识协议的鲁棒性、安全性与效率。共识协议的设计一方面依赖于系统的正确性假设与安全性证明，另一方面，通过密码学技术可极大地简化共识协议的设计，并实现更易证明的安全保障，但随之也带来了更多的计算开销。

4.2　共识问题中的等价性

在共识问题的研究中，研究者也逐渐发现并证明了一些等价性问题。共识问题中的等价性是指在特定模型下的分布式系统中，如果存在算法能够将解决问题 A 的任何算法转换为解决问题 B 的算法，那么问题 A 和问题 B 是等价的[92]。等价性问题的研究，有助于拓展共识协议的适用范围，还可以拓宽分布式问题的定义与研究思路。

Correia 等[93]探讨了共识问题与其他分布式系统问题的等价性，指出了现有研究已证明的相关等价性问题：1）在崩溃故障模型中，多值共识问题被证明与原子广播问题等价[34]；2）在某些条件下，原子广播问题与拜占庭共识问题等价[94]；3）二元共识到多值共识，以及从多值共识到向量共识之间存在转换关系[28-32]，但这些转换在某些情况下需要数字签名来保证安全性；4）向量共识可能与组成员身份问题存在转换关系。

然而，在拜占庭故障模型中，由于进程可能存在任意的故障或恶意行为，因此并非所有在崩溃故障模型中建立的等价性都能直接扩展到拜占庭故障模型。

结束语　加密货币[95]和区块链技术[96]的成功为共识协议带来了更广阔的应用场景，并展示了其在广域网上达成共识的可能性。在开放的互联网环境中，节点之间的网络延迟可能是变化的，然而同步或部分同步共识协议只能适应节点连接良好的网络，如果时间假设不成立，这些协议将停滞不前。因此，不考虑同步假设的共识协议受到越来越多的关注。异步共识协议受到关注的另一个原因是其高“响应性”，大多数同步或部分同步共识协议的效率取决于所假设的网络延迟，而高“响应性”则要求性能仅与实际网络延迟有关，因此对于不依赖于时间假设的异步共识协议，一旦消息被送达，协议就能取得进展。此外，在构建分布式系统时，因为无需处理超时机制及相应异常，异步共识协议可大大降低系统的复杂性。

FLP 结论的诞生，成为异步共识协议的阿喀琉斯之踵。半个世纪以来，研究者们一直在探索如何规避 FLP 结论，以实现在异步网络中的共识。早期的异步共识协议多为理论可能，难以真正实用；并且无论哪种方法，都无法实现完美的异步共识，每种实现途径都有其特定的优势和局限性，在选择使用何种共识协议时，需要根据具体的应用场景和系统特性来权衡各种算法的优劣。

异步共识协议的研究将持续发展。未来，在探索规避 FLP 方法的同时，可以将多种技术途径相结合，使得异步共识协议在保证系统安全性和可靠性的同时，实现更高的效率和更广泛的适用范围。

参 考 文 献

[1] EISENBERG E，GALE D. Consensus ofSubjective Probabilities：the Pari-Mutuel Method[J]. The Annals of Mathematical Statistics，1959，30(1)：165-168.

[2] AKKOYUNLU E A，EKANADHAM K，HUBER R V. SomeConstraints and Tradeoffs in the Design of Network Communications[C]// Proceedings of the 5th ACM Symposium on Operating Systems Principles. Austin，Texas，USA：ACM Press，1975：67-74.

[3] PEASE M，SHOSTAK R，LAMPORT L. Reaching Agreement in the Presence of Faults[J]. Journal of the ACM，1980，27(2)：228-234.

[4] DOLEV D，DWORK C，STOCKMEYER L. On theMinimal Synchronism Needed for Distributed Consensus[J]. Journal of the ACM，1987，34(1)：77-97.

[5] XIA Q，DOU W S，GUO K W，et al. A Survey on Blockchain Consensus Protocols[J]. Journal of Software，2021，32(2)：277-299.

[6] JIN S X，ZHANG X D，GE J G，et al. A Research Review on Blockchain Consensus Algorithms[J]. Journal of Information Security，2021，6(2)：85-100.

[7] LIU Y Z，LIU J W，ZHANG Z Y，et al. A Research Review on Blockchain Consensus Mechanisms[J]. Cryptography Journal，2019，6(4)：395-432.

[8] DENG X H，WANG Z Q，LI J，et al. Comparative Research on Mainstream Blockchain Consensus Algorithms[J]. Application Research of Computers，2022，39(1)：1-8.

[9] WANG Q，LI F J，NI X L，et al. Survey on Blockchain Consensus Algorithms and Application[J]. Journal of Frontiers of Computer Science and Technology，2022，16(6)：1214-1239.

[10] TAN P L，WANG R S，ZENG W H，et al. Overview of Blockchain Consensus Algorithms[J/OL]. https://www.jsjkx.com/CN/article/openArticlePDF.jsp? id=21672.

[11] BALDONI R，HELARY J M. Strengthening Distributed Protocols to Handle Tougher Failures[R]. France：Univ. de Rennes 1，2002.

[12] VORAPANYA A. Large-Scale Distributed Services[D]. Florida：University of Florida，2000.

[13] FISCHER M J，LYNCH N A，PATERSON M S. Impossibility ofDistributed Consensus with one Faulty Process[J]. Journal of the ACM，1985，32(2)：374-382.

[14] WATTENHOFER R. The Science of the Blockchain[M]. USA：CreateSpace Independent Publishing Platform，2016.

[15] LAMPORT L，SHOSTAK R，PEASE M. The Byzantine generals problem[J]. ACM Transactions onProgramming Languages and Systems，1982，4(3)：382-401.

[16] DOLEV D，LYNCH N A，PINTER S S，et al. ReachingApproximate Agreement in the Presence of Faults[J]. Journal of the ACM，1986，33(3)：499-516.

[17] CASTRO M，LISKOV B. Practical ByzantineFault Tolerance and Proactive Recovery[J]. ACM Transactions on Computer Systems，1999，17(4)：173-186.

[18] BARBORAK M，MALEK M，DAHBURA A. The Consensus Problem in Fault-Tolerance Computing[J]. ACM Computing Surveys，1998，6(25)：171-220.

[19] FISHER M J. TheConsensus Problem in Unreliable Distributed Systems(A Brief Survey)[M]. Berlin：Springer-Verlag Press，1983：127-140.

[20] LAMPORT L. How toMake a Multiprocessor Computer That Correctly Executes Multiprocess Programs[J]. IEEE Transactions on Computers，1979，28(9)：690-691.

[21] KRISHNA S，EMMI M，ENEA C，et al. Verifying Visibility-

Based Weak Consistency[C]// Proceedings of the 29th European Symposium on Programming. Dublin: Springer Press, 2020: 280-307.

[22] LOUT M C, ABU-AMARA H H. MemoryRequirements for Agreement Among Unreliable Asynchronous Processes[J]. Advances in Computing Research, 1987, 4: 163-183.

[23] RABIN M O. Probabilistic algorithms[M]. New York: Academic Press, 1976: 21-39.

[24] RABIN M O. Randomized Byzantine Generals[C]// Proceedings of the 24th Annual Symposium on Foundations of Computer Science. Tucson, AZ, USA: IEEE Press, 1983: 403-409.

[25] BEN-OR M. Another advantage of free choice (Extended Abstract): Completely asynchronous agreement protocols[C]// Proceedings of the ACM Symposium on Principles of Distributed Computing. New York: ACM Press, 1983: 27-30.

[26] DOLEV D, STRONG R. Polynomial Algorithms for Byzantine Agreement[C]// Proceedings of the 14th ACM Symposium on Theory of Computing. New York: ACM Press, 1982: 401-407.

[27] FISCHER M, LYNCH N. A Lower Bound for the Time to Assure InteractiveConsistency[J]. Information Processing Letters, 1982, 14(4): 182-186.

[28] TOUEG S. Randomized Byzantine Agreements [C]// Proceedings of the third ACM Symposium on Principles of Distributed Computing. Vancouver, British, Columbia, Canada: ACM Press, 1984: 183-178.

[29] SHAMIR A. How to share a secret[J]. Communications of the ACM, 1979, 22(11): 612-613.

[30] BRACHA G, TOUEG S. Asynchronous consensus and broadcast protocols[J]. Journal of the ACM, 1985, 32(4): 824-840.

[31] BRACHA G. An Asynchronous $(n-1)/3$-Resilient Consensus Protocol[C]// Proceedings of the third ACM Symposium on Principles of Distributed Computing. Vancouver, British, Columbia, Canada: ACM Press, 1984: 154-162.

[32] CACHIN C, KURSAWE K, SHOUP V. RandomOracles in Constantinople: Practical Asynchronous Byzantine Agreement Using Cryptography[J]. Journal of Cryptology, 2005, 18(3): 219-246.

[33] CACHIN C, KURSAWE K, PETZOLD F, et al. Secure and Efficient Asynchronous Broadcast Protocols[C]// Annual International Cryptology Conference. Berlin, Heidelberg: Springer Press, 2001: 524-541.

[34] CHANDRA T D, TOUEG S. Unreliable Failure Detectors for Reliable Distributed Systems[J]. Journal of the ACM, 1996, 43(2): 225-267.

[35] MOUSTAFA H, MOSTEFAOUI A, RAYNAL M. Signature-free asynchronous Byzantine consensus with $t<n/3$ and $O(n^2)$ messages[C]// Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing. New York: ACM, 2014: 2-9.

[36] BRACHA G. Asynchronous Byzantine agreement protocols[J]. Information & Computation, 1987, 75(2): 130-143.

[37] SRIKANTH T. K, TOUEG S. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms[J]. Distributed Computing, 1987, 2: 80-94.

[38] BEN-OR M, KELMER B, RABIN T. Asynchronous secure computations with optimal resilience[C]// Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing. New York: ACM Press, 1994: 183-192.

[39] MILLER A, XIA Y, CROMAN K, et al. The honey badger of BFT protocols[C]// Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2016: 31-42.

[40] CACHIN C, TESSARO S. Asynchronous verifiable information dispersal[C]// Proceedings of the International Conference on Distributed Computing. Orlando, FL, USA: IEEE Press, 2005: 191-201.

[41] GUO B Y, LI X Y. A High-Efficiency Multi-Value Byzantine Consensus Scheme [J]. Cryptography Journal, 2018, 5(5): 516-528.

[42] GUO B, LU Z, TANG Q, et al. Dumbo: Faster Asynchronous BFT Protocols[C]// CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2020: 803-818.

[43] WANG Y Q, LIU Y, LI X Y, et al. Efficient asynchronous Byzantine fault tolerance algorithm for blockchain[J]. Application Research of Computers, 2023, 40(9): 2590-2599.

[44] BAIRD L. The Swirlds hashgraph consensus algorithm: Fair, fast, Byzantine fault tolerance: Technical Report: SWIRLDS-TR-2016[R]. 2016.

[45] DEMERS A, GREENE D, HAUSER C, et al. Epidemic algorithms for replicated database maintenance[J]. ACM SIGOPS Operating Systems Review, 1988, 22(1): 8-32.

[46] EZHILCHELVAN P, MOSTEFAOUI A, RAYNAL M. Randomized multivalued consensus[C]// Proceedings of the 4th International Symposium on Object-oriented Real-time Distributed Computing. Newport Beach, CA, USA: IEEE Press, 2001: 195-201.

[47] HURFIN M, MOSTEFAOUI A, RAYNAL M. A versatile family of consensus protocols based on Chandra-Toueg's unreliable failure detectors[J]. IEEE Transactions on Computers, 2002, 51(4): 395-408.

[48] DOLEV D, DWORK C, STOCKMEYER L. On the minimal synchronism needed for distributed consensus[J]. Journal of the ACM, 1987, 34(1): 77-97.

[49] DWORK C, LYNCH N, STOCKMEYER L. Consensus in the presence of partial synchrony[J]. Journal of the ACM, 1988, 35(2): 288-323.

[50] OKI B M, LISKOV B H. Viewstamped replication: a new primary copy method to support highly-available distributed systems [C]// Proceedings of the 7th Annual ACM Symposium on Principles of Distributed Computing. Toronto, Ontario, Canada: ACM Press, 1988: 8-17.

[51] ABRAHAM I, MALKHI D, SPIEGELMAN A. Validated Asynchronous Byzantine Agreement with Optimal Resilience and Asymptotically Optimal Time and Word Communication [EB/OL]. (2018-11-04) [2024-04-28]. https://arxiv.org/abs/1811.01332.

[52] ABRAHAM I, MALKHI D, SPIEGELMAN A. Asymptotically optimal validated asynchronous byzantine agreement[C]// Pro-

ceedings of the 2019 ACM Symposium on Principles of Distributed Computing. Toronto, ON, Canada: ACM Press, 2019: 337-346.

[53] YIN M F, MALKHI D, REITER M K, et al. HotStuff: BFT consensus in thelens of blockchain[J]. arXiv. 1803. 05069, 2018.

[54] VERONESE G S, CORREIA M, BESSANI A N, et al. Efficient Byzantine fault-tolerance[J]. IEEE Transactions on Computers, 2013, 62(1): 16-30.

[55] BIRAN O, MORAN S, ZAKS S. A combinatorial characterization of the distributed tasks that are solvable in the presence of one faulty processor[C]// Proceedings of the Seventh ACM Symposium on Principles of Distributed Computing. Toronto, Ontario: ACM Press, 1988: 263-275.

[56] TEL G. Introduction to Distributed Algorithms [M]. Cambridge: Cambridge University Press, 2003: 505-509.

[57] AGUILERA M K, TOUEG S, DEIANOV B. Revisiting the weakest failure detector for uniform reliable broadcast[C]// Proceedings of the 13th International Symposium on Distributed Computing. Heidelberg: Distributed Computing, 1999: 19-33.

[58] LIU J X, WU Z B, DONG J, et al. Summarization on failure detector in distributed system[J]. Intelligent Computer and Applications, 2019, 9(2): 215-220.

[59] FAGG G E, DONGARRA J J. Building and using a Fault Tolerant MPI implementation[J]. International Journal of High Performance Applications and Supercomputing, 2004, 18(3): 353-361.

[60] BATCHU R, DANDASS Y, SKJELLUM A, et al. A Model-Based Approach to Low-Overhead Fault Tolerant Message-Passing Middleware[J]. Cluster Computing, 2004, 7(4): 303-315.

[61] FOSTER I, KESSELMAN C, TUECKE S. The anatomy of the grid[J]. International Journal of High Performance Computing Applications, 2001, 15(3): 200-222.

[62] STELLING P, FOSTER I, KESSELMAN C, et al. A fault detection service for wide area distributed computations[C]// Proceedings of the 7th IEEE Symposium on High Performance Distributed Computing. Chicago: IEEE Press, 1998: 268-278.

[63] BURNS M W, GEORGE A D, WALLACE B A. Simulative Performance Analysis of Gossip Failure Detection for Scalable Distributed Systems[J]. Cluster Computing, 1999, 2(3): 207-217.

[64] WANG G. Research on Consensus Protocols of Asynchronous Systems Based on Fault Detectors [D]. Changsha: Hunan University, 2006.

[65] AGUILERA M K, CHEN W, TOUEG S. On the Weakest Failure Detector for Quiescent Reliable Communication[R]. New York: Cornell University, 1997: 2-15.

[66] LAMPORT L. The part-time parliament[J]. ACM Transactions on Computer Systems, 1998, 16(2): 133-169.

[67] MARTIN J P, ALVIZI L. Fast Byzantine consensus [J]. IEEE Transactions on Dependable and Secure Computing, 2006, 3(3): 202-215.

[68] PRISCO R D, LAMPSON B, LYNCH N. Revisiting the PAXOS algorithm[J]. Theoretical Computer Science, 2000, 243(1/2): 35-91.

[69] CHANDRA T D, HADZILACOS V, TOUEG S. The weakest failure detector for solving consensus[J]. Journal of the ACM, 1996, 43(4): 685-722.

[70] KIHLSTROM K P, MOSER L E, MELLIAR-SMITH P M. Byzantine fault detectors for solving consensus[J]. The Computer Journal, 2003, 46(1): 16-35.

[71] KIHLSTROM K P, MOSER L E, MELLIAR-SMITH P M. Solving Consensus in a Byzantine Environment Using an Unreliable Fault Detector[C]// International Conference on Principles of Distributed Systems. 1997: 61-75.

[72] DOUDOU A, GARBINATO B, GUERRAOUI R, et al. Muteness failure detectors: Specification and implementation[C]// European Dependable Computing Conference. Berlin: Springer Press, 1999: 71-87.

[73] MOSTEFAOUI A, RAYNAL M. Solving consensus using Chandra-Toueg's unreliable failure detectors: a general quorum-based approach [C] // Proceedings of the 13th International Symposium on Distributed Computing. Bratislava: Springer Press, 1999: 49-63.

[74] DUTTA P, GUERRAOUI R. Fast indulgent consensus with zero degradation[C]// Proceedings of the 4th European Dependable Computing Conference. Berlin: Springer Press, 2002: 191-208.

[75] MALKHI D, REITER M. Unreliable intrusion detection in distributed computations[C]// Proceedings of the 10th Computer Security Foundations Workshop. Los Alamitos: IEEE Press, 1997: 116-124.

[76] SCHIPER A. Early consensus in an asynchronous system with a weak failure detector[J]. Distributed Computing, 1997, 10(3): 149-157.

[77] FRIEDMAN R, MOSTEFAOUI A, RAYNAL M. Simple and efficient oracle-based consensus protocols for asynchronous Byzantine systems[J]. IEEE Transactions on Dependable and Secure Computing, 2005, 2(1): 46-56.

[78] BALDONI R, HÉLARY J M, RAYNAL M, et al. Consensus in Byzantine asynchronous systems[J]. Journal of Discrete Algorithms, 2003, 1(2): 185-210.

[79] MOSTEFAOUI A, RAJSBAUM S, RAYNAL M. Conditions on input vectors for consensus solvability in asynchronous distributed systems[J]. Journal of the ACM, 2003, 50(6): 922-954.

[80] MOSTÉFAOUI A, RAJSBAUM S, Raynal M, et al. Condition-based consensus solvability: a hierarchy of conditions and efficient protocols[J]. Distributed Computing, 2004, 17(1): 1-20.

[81] FRIEDMAN R, MOSTÉFAOUI A, RAJSBAUM S, et al. Distributed agreement and its relation with error-correcting codes [C]// Proceedings of the International Symposium on Distributed Computing. Berlin: Springer Press, 2002: 63-87.

[82] FRIEDMAN R, MOSTÉFAOUI A, RAJSBAUM S, et al. Distributed agreement problems and their connection with error-correcting codes[J]. IEEE Transactions on Computers, 2007, 56(7): 865-875.

[83] FETZER C, CRISTIAN F. On the possibility of consensus in a-

synchronous systems[C]// Proceedings of the Pacific Rim International Symposium on Fault-Tolerant Systems. California: IEEE Press, 1995: 86-91.

[84] AGUILERA M K, TOUEG S. Failure detection and randomization: a hybrid approach to solve consensus[J]. SIAM Journal of Computing, 1998, 28(3): 890-903.

[85] MOSTEFAOUI A, RAYNAL M, TRONEL F. The best of both worlds: a hybrid approach to solve consensus[C]// Proceedings of the International Conference on Dependable Systems and Networks. New York: IEEE Press, 2000: 513-522.

[86] GUERRAOUI R, RAYNAL M. The Information Structure of Indulgent Consensus[J]. IEEE Transactions on Computers, 2004, 53(4): 453-466.

[87] CANETTI R, RABIN T. Fast asynchronous Byzantine agreement with optimalresilience[C]// Proceedings of the 25th Annual ACM Symposium on Theory of Computing. San Diego, California: ACM Press 1993: 42-51.

[88] SCHMID U, FETZER C. Randomized Asynchronous Consensus with Imperfect Communications[C]// 22nd Symposium on Reliable Distributed Systems. Florence, Italy: IEEE Press, 2003: 362-370.

[89] CHOR B, DWORK C. Randomization in Byzantine agreement [J]. Advances in Computer Research. Greenwich, 1989, 5: 443-497.

[90] FRIEDMAN R, MOSTÉFAOUI A, RAYNAL M. Pmute-based consensus for asynchronous Byzantinesystems[J]. Parallel Processing Letters, 2005, 15(1/2): 162-182.

[91] AGUILERA M K, CHEN W, TOUEG S. Failure detection and consensus in the crash-recovery model[J]. Distributed Computing, 2000, 13(2): 99-125.

[92] HADZILACOS V, TOUEG S. Fault-tolerant broadcasts and related problems[C]// Distributed Systems (2nd Ed.). 1993: 97-145.

[93] CORREIA M, VERONESE G S, NEVES N F, et al. Byzantine consensus in asynchronous message-passing systems: a survey [J]. International Journal of Critical Computer-Based Systems, 2011, 2(2): 141-161.

[94] CORREIA M, NEVES N F, VERISSIMO P. From consensus to atomic broadcast: time-free Byzantine-resistant protocols without signatures[J]. The Computer Journal, 2006, 49(1): 82-96.

[95] NAKAMOTO S. Bitcoin: a peer-to-peer electronic cash system [EB/OL]. (2008-06-10)[2024-05-13]. https://bitcoin.org/bitcoin.pdf.

[96] MA Z F. Blockchain Technology Development Guide [M]. Beijing: Tsinghua University Press, 2021.

WANG Di, born in 1988, doctoral student, engineer. His main research interests include blockchain and consensus algorithm.

(责任编辑:何杨)