设计指导

星云研究院

2019 年 3 月
版本号:0.0.1

目录

1 简介

分布式系统中的共识问题 (Consensus Problem) 实际上大多数是指拜占庭将军问题（Byzantine Generals Problem，BGP）[1]，即拜占庭容错问题（Byzantine Fault Tolerant，BFT）[2] 的特例。通常针对 BGP 的算法往往假设对提案者的选择（leader election）已经达成共识。

在传统分布式服务中 [3]，领导者的选取由发起请求的客户端（client）指定或者由固定算法决定 (例如 $mod\ n$)，通常在领导者发生故障时则会进行领导人重新选举。因此在正常情况下，共识并不包括领导选举过程。

区块链共识属于广义上的 BFT 问题，我们认为完整的共识流程应该包括**领袖选举（Leader Election）和交易验证**（Transaction Verification）两个步骤，在部分研究中也有将领袖选举和交易验证解耦的思想 [4, 5]。

1.1 领袖选举

为保证匿名性，区块链中所有节点均表示为基于公钥产生的地址（例如以太坊中长度为 64 位的 16 进制字符串），同时为保证去中心化以及防止 DDoS 攻击，客户端产生的 transaction 会更新到全节点共享的交易池中（transactions pool）而无法指定特定的节点来发起提案（proposal）[]。

对于联盟链以及部分采用 DPoS 机制的公链（例如 EOS），参与者需要进行身份认证并且不能随意加入离开，因此被定义为 permissioned chain。一段时间内所有参与者集合已知且不变，提案者选举通常采用简单的轮询机制（round-robin）[]，因此整个共识问题变成在提案者选择已经达成共识的情况下（此时提案者仍可以采取恶意行动），所有复制节点（backups 或 replicas）如何针对提案达成一致，后者可以基于 PBFT 之类的协议实现。

以 Bitcoin 为代表的公链对于出块节点（即提案者）没有准入限制，参与节点通过竞争方式获取提案者的选举权，此类区块链系统称之为 permissionless chain。竞争式提案者选举最为广泛采用工作量证明（Proof-of-Work，PoW），例如 Bitcoin 和现阶段的 Ethereum 所采用的谜题计算方式（puzzle solving）。需要指出的是，PoW 仅仅实现了提案者选举，而并不是完整的共识协议，这一点在最早的 Bitcoin 白皮书上已经指出 [6]，而一种普遍的错误是将 PoW 和比特币共识（通常称之为 Nakamoto Consensus）混为一谈。

由于 PoW 会消耗大量电力，权益证明（Proof-of-Stake，PoS）随后被提出，不同于 PoW，后者实现选举往往与链上资产相关而不依赖于物理算力因此都称之为 PoS，但不同基于 PoS 的共识之间通常差别较大。早期的 PoS 仍然是基于 PoW 的变种实现 [7]，即对谜题计算中的 target 基于 stake 或币龄（coin-age）进行加权。另一个备受瞩目的以太坊项目 Casper[8] 被认为是一种 PoS 协议，但实际上 Casper 是利用 stake 实现链式结构的终结性 (finality)，并不涉及提案者选举过程，在 [8] 中也指出可以通过 PoW 实现出块者确定。

近几年涌现出一些基于随机数生成（Random Number Generation，RNG）的共识机制实现了非 PoW 提案者选举方案 [9, 10, 11]。尽管在出块方式和共识实现上存在区别（例如 Ouroboros 基于弱同步时钟），上述方案均利用伪随机函数（pseudo-random function）实现提案者的随机选举。

Permissionless chain 对于参与者没有身份验证 [rethinking]，因此其出块者选取需要引入随机性，以保证安全。同时根据 [] 的研究，在网络传输带宽有限的情况下，区块产生间隔和链分叉率存在相关性，因此在采用最长链等结构的区块链中，我们认为出块者选举机制也需要控制全网出块频率。

1.2 交易验证

在分布式系统中，状态机复制可以有效解决多服务器之间的数据一致性问题 [12]。最为经典的 Paxos 协议实现了宕机容错（crash-fault tolerant）的一致性算法 [13]，其被广泛应用于数据库备份和域名服务等领域 [14, 15]。

随着区块链和数字货币的发展，基于拜占庭容错的一致性算法重新得到关注。在大规模节点网络中，网络通信成为瓶颈。以 Bitcoin 为代表的 permissionless chain 采取了最长链原则（Longest Chain Rule）[6]，[16] 提出了最重子树原则（GHOST Rule）。最长链和 GHOST 协议通常被称之为链式协议（chain-based Protocols），链式协议中节点仅依据接收区块构建链结构实现一致性而不产生额外的状态同步开销。对应地，链式协议仅能保证最终一致性（Eventual Consistency），即保证所有节点最终就共享状态达成一致，但在某个时间段内，共享状态各个节点所存的状态可能不一致。

联盟链以及部分公链通过缩小节点规模，实现了更强的一致性保证。节点针对每个提案进行投票，通常基于 PBFT[3] 或者更为宽松的投机 BFT 算法 [17]，此类机制被称之为投票协议（vote-based Protocols）。相比于链式协议，投票协议对系统要求更为严格，PBFT 和最差情况下的投机 BFT 网络通信开销为 $O(n^2)$，因此采用此类协议的网络往往规模较小。

基于 PBFT 机制，ByzCoin[5] 利用集体签名 (collective signing) 以及群签名 (CoSi) 实现了投票节点规模可扩展，因而在一些分片技术（sharding）中也被采用

[18]。

采用投票协议的区块链会针对每个出块进行一致性确认，因此每个区块状态不可能被扭转，从而实现终结性（Finality）。相比之下，基于链式结构的区块链在达到最终一致性前任何区块状态都可能发生改变（例如某个区块在某个时间后不再属于最长链），通常这种改变概率随着链式结构增长单调递减 [6]，因此链式结构共识无法保证终结性，或者仅提供概率终结性（Probabilistic Finality）。较为特殊地，Casper[8] 在检查点（checkpoints）内部采用最长链原则，而不同检查点间则通过投票保证终结性。

需要指出的是，通常认为采用链式协议的系统吞吐量（Throughput）低于基于投票协议的系统，实际上在仅满足最终一致性条件以及同等节点规模和网络环境下，链式协议实现状态同步开销更低。

2 系统模型

2.1 网络模型

异步网络设定 TBA

由于网络传输延时的存在，网络中的节点无法实现完美同步时钟（Perfectly Synchronized Clock）。因此 TBC

在后文中，若没有特殊说明，t 均表示节点的本地时间。

2.2 数据模型

在区块链中，最基本的数据结构是交易（Transaction）和账户（Account），这里给出相关定义。

定义 1. （账户）一个账户 i 拥有一个私钥（Secret Key）sk_i 和基于私钥构建的公钥（Public Key）pk_i。一个账户 i 可以表示为多元组 $\langle pk_i, sk_i, s_i \rangle$，其中 s_i 表示用户 i 状态[1]。

以 Bitcoin 为代表的交易网络主要采用了未交易输出模型（Unspent Transaction Outputs Model，UTXO Model）；而以 Ethereum 为代表的区块链系统采用了账户模型（Account Model）。通常来说，账户模型设计更接近于传统记账系统（例如银行），

[1]在交易网络中，用户状态即用户余额，随着智能合约和用户行为的多样化，这里统一称之为用户状态

并且支持图灵完备智能合约（Turing-complete Smart Contract），因此这里我们基于账户模型给出相关定义，需要指出的是，UTXO 模型理论上同样适用于我们的共识设计。

定义 2. （系统状态）系统状态定义为系统中所有账户的状态，即 $S = \{s_1, ..., s_n\}$。

定义 3. （交易）交易描述了一次系统的状态更变，一个交易包含交易发起账户 i 和接受账户 j 的状态更变，即 $tx_k = \langle s_i, s_i', s_j, s_j' \rangle$，其中 s_i，s_j 表示交易发生前账户的状态，s_i'，s_j' 表示交易发生后账户的状态[2]。交易也可能使账户状态不发生变化，即 $s_i = s_i'$ 或者 $s_j = s_j'$。

在此基础之上，我们给出区块的定义。

定义 4. （区块）区块是一个数据结构，一个区块 B 包含了一组交易 $Tx_B = \{tx_1, \ldots, tx_n\}$，一个指向上一个区块的引用 $h(B^{\prec})$，出块者的证明 φ_B 和时间戳 t_B，其中 $B^{\prec}$ 表示 B 所指向的前置区块，$h(\cdot)$ 为某种哈希函数。

3 领袖选举

3.1 基础

在传统分布式系统 [13, 3] 中负责发起状态更变请求的节点被定义为主节点 (Primary)，接受并验证请求的为副本节点（Backup）。而在区块链系统中，所有参与打包区块以及验证区块的节点，统称为矿工（Miner）[3]。为不引起混淆，后文统一将执行领袖选举协议的节点称之为提案者（Proposal），而选举具有出块权限的节点称之为领袖（Leader）。

首先，我们给出领袖选举的相关定义。

定义 5. （提案）对于提案者 i，其提案操作表示为函数 $M_i(Tx, B', \zeta_i)$，其中 Tx 为待验证交易集合，B' 为某已有区块，ζ_i 为提案参数。$M_i(Tx, \zeta_i) \in \{B, \perp\}$。

定义 6. （验证）对于提案者 i，其验证操作表示为函数 $V_i(B, t_i)$，其中 B 为待验证区块，t_i 为提案者 i 的本地时间。$V_i(B, t_i) \in \{0, 1\}$。

[2]这里假设交易发生在一对账户间，对于多对多的交易可以转换为多笔交易

[3]对于只参与同步并验证区块而不进行出块的节点，也称之为全量节点。

定义 7. （领袖选举协议 Π）对于提案者集合 N[4]，提案者 $i \in N$ 在 Tx 和 B' 不为空的情况下，执行提案操作 M_i，若输出为 B 则将其广播；同时提案者 $j \in N$ 在收到区块 B 后执行验证操作 V_j。对于提案者 $i \in N$，满足 $\exists j \in N, M_i = B \wedge V_j = 1$ 时，提案者 i 成为领袖。

通常而言，我们认为领袖选举机制需要满足以下性质：

- 随机性：提案者在执行提案操作前无法预知自己或者他人是否会成为领袖。
- 可验证：TBC

不难发现选举协议 Π 的随机性来自于提案操作 M 的随机性，这里我们首先给出 Π 随机性的数学描述。我们将时间表示为有序的离散[5]集合 T，因此提案者 i 的提案可以表示为随机过程 $\mathbb{X} = \{X(i,t), t \in T\}$。其中 $X(i,t)$ 表示为在 t 内 M_i 输出不为 $\perp$ 的概率。根据数学定义 [19]，如果一个随机过程在某个时刻的取值在这个时刻之前就可能可以知道（可测），则此随机过程称之为可预测过程（Predictable Process）。

定义 8. （提案 n 阶可预测性）在 t 时刻，如果已知 Tx, B', ζ_i，可确定 $t+1$ 至 $t+n$ 时刻 M_i 的取值，但无法确定 $t+n+1$ 时刻 M_i 的取值，则 M 具有 n 阶可预测性。

对应地，我们给出不可预测性的定义。

定义 9. （提案不可预测性）在 t 时刻，如果已知 Tx, B', ζ_i，不可确定 $t+1$ 时刻 M_i 的取值，则 M 具有不可预测性。

当选举协议可预测时，会出现若干潜在问题。例如自私挖矿（Selfish Mining）不需要承担风险因而变得更加普遍 []，并且节点在成为领袖前可能存在消极运行的问题（"last actor" problem）[11]。更进一步地，目前许多链上 DApp（例如博彩）的随机性来自于出块随机性，当出块可预测时，此类 DApp 则会有被操控风险。

此外，网络中的节点在接受到区块时需要首先验证区块的有效性，因此合法的区块需要具有有效性的特征。

特征 1. 基于区块 B 本身即可验证其有效性，即区块 B 的有效性仅与交易组 Tx_B、$B^{\prec}$ 和 φ_B 相关。

[4] N 可以为有限或者无限集合。
[5] 当 T 为连续集合时，该定义仍然可描述，这里简单采用离散集合描述。

对于 UTXO 模型和 Account 模型，交易组的有效性验证方式不同，这里不做过多展开。对于 φ_B，根据领袖选举协议的不同其有效性验证方式也有所不同，具体分析见后续章节。在保证 Tx_B 和 $B \prec$ 有效性的情况下，区块 B 的有效性依赖于其出块证明 φ_B 的有效性。

3.2 PoW 协议

我们首先给出基于 PoW 的领袖选举机制相关分析。

定义 10. (选举协议 Π_w)基于 PoW 机制的领袖选举协议,有 $\epsilon = \langle d, t, nonce, Mr(Tx), \tau \rangle$, 且:

$$M(Tx, B', \epsilon) = \begin{cases} B & \text{if } h(d, t, h(B'), nonce, Mr(Tx)) < \tau \\ \perp & \text{otherwise} \end{cases} \tag{1}$$

其中 $d \in N^+$ 表示困难程度，通常会随着时间动态调整,τ 表示目标值，$nonce$ 为一个临时随机数，t 为时间戳，$h(\cdot)$ 表示某种哈希函数，例如 Bitcoin 中采用的 SHA-256。对于提案区块 B，有 $t_B = t$，$Tx_B = Tx$，$Pre(B) = h(B')$ 以及 $V(B) = \epsilon$。

推论 1. Π_w 满足伪随机性。

证明. 根据公式1，$\Pr(M(Tx, B', \epsilon) = B)$ =$\Pr(h(d, t, h(B')|nonce, Mr(Tx)) < \tau)$，$h(\cdot)$ 具有伪随机性[6]，TBC □

推论 2. Π_w 具有可验证性。

证明. 对于提案区块 B，有 $V(B) = \epsilon$，对于其他提案者已知 $nonce$ TBC □

3.3 非 PoW 协议

目前基于非 PoW 的领袖选举协议主要是基于权益的领袖选举，在此我们给出几种具有代表性的非 PoW 选举协议。

[6]在给定 y 情况下，目前没有比穷举更好的算法来寻找 x 使其满足 $h(y|x) < c$。

定义 11. （选举协议 $\Pi_{\bar{w}}^{\alpha}$）对于提案函数 $M_i(Tx, B', \zeta_i)$，有 $\zeta_i = \langle s_i, t_i, \delta \rangle$，其中 s_i 表示提案者 i 的状态（即权益），t_i 为 i 的本地时间，δ 表示某个伪随机数，并且

$$M_i(Tx, B', \zeta_i) = \begin{cases} B & \text{if } g(s_i, \delta, t_i) = 0 \\ \perp & \text{otherwise} \end{cases} \tag{2}$$

$g(\cdot)$ 表示某种函数。对于提案区块 B，$Tx_B = Tx$，$B^{\prec} = B'$（即 $h(B^{\prec}) = h(B')$），$\varphi_B = \langle t_B, s_i, sig_i, \delta \rangle$，其中 $t_B = t_i|_{M_i=B}$，sig。

对于验证操作 V_j，具体地：

$$V_j(B, t_j) = \begin{cases} 1 & \text{if } V(B^{\prec}, t_j) = 1, t_j > t_B, g(s_i, \delta, t_B) = 0, sig_i ... \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

可以看出，执行协议 $\Pi_{\bar{w}}^{\alpha}$ 的提案者需要提供某个账户的权益证明（即 s_i），当提案者的当前时间满足条件 $g(s_i, \delta, t_B) = 0$ 时，则可以成为领袖出块。

推论 3. $\Pi_{\bar{w}}^{\alpha}$ 具有 1 阶可预测性。

证明. 对提案者 i，在 $t = n$ 时刻，已知 Tx，B' 和 $\zeta_i = \langle s_i, t, \delta \rangle$，$X(i,t)$=$\Pr(g(s_i, \delta, n) = 0) \to [0, 1]$。因为 g 为确定性函数，令 $t = n + 1$，可以求得 $g(s_i, \delta, n + 1)$，即在 t 时刻给定输入 Tx，B' 和 ζ_i，可以预测 $X(i, t + 1)$。 □

推论 4. $\Pi_{\bar{w}}^{\alpha}$ 具有 ∞ 阶可预测性。

证明. 对提案者 i，在 $t = n$ 时刻，已知 Tx，B' 和 $\zeta_i = \langle s_i, t, \delta \rangle$，$X(i,t)$=$\Pr(g(s_i, \delta, n) = 0) \to [0, 1]$。令 $t = n + m$，可以求得 $lim\Pr(g(s_i, \delta, n) = 0)$。即在 t 时刻给定输入 Tx，B' 和 ζ_i，可以预测 $X(i, t + m)_{m \to \infty}$。 □

可以发现最原始的协议不具备不可预测性，我们基于此做部分改动。

定义 12. （领袖选举协议 $\Pi_{\bar{w}}^{\beta}$）对于提案函数 $M_i(Tx, B', \zeta_i)$，有 $\zeta_i = \langle s_i, t_i, \delta \rangle$，其中 s_i 表示提案者 i 的状态（即权益），t_i 为 i 的本地时间，δ 表示某个伪随机数，并且 δ 更新周期为 Δ。

推论 5. $\Pi_{\bar{w}}^{\beta}$ 具有 Δ 阶可预测性。

4 交易验证

基于第3章，在已选举出提案者的情况下，交易验证对出块进行验证。首先给出拜占庭环境下交易验证的相关定义和特性。

定义 13. （交易验证）对于验证者集合 V[7]，包含 n 个节点，其中最多有 f 个拜占庭节点，即最少有 $n-f$ 个诚实节点。所有节点必须从提案中最终做出决策，并且满足下述条件：

- 一致性（Agreement）：所有诚实节点的决策必定相同。
- 可终止性（Termination）：所有诚实节点在有限的时间内结束决策过程。
- 有效性（Validity）：选择出的决策值必须来自某个有效的提案。

对于任何安全的交易验证协议，都需要满足上述条件[8]。如1.2小节介绍，交易确认机制根据块是否具有终结性划分为链式协议和基于投票协议。投票式交易验证满足终结性，即交易一旦完成验证则不可能发生改变。对应地，链式交易验证则仅满足概率终结性，即随着链结构的增长交易发生改变逐渐降低，但无法为 0。需要指出的是，即便是类似 PBFT 的投票协议，其数据结构仍可以采用链式区块结构。

4.1 投票协议

从安全性角度，我们不希望已经验证的交易会存在修改的可能，即出现长距离攻击（Long-Range Attack）。同时，随着数据的不断增长，一些普通节点可能无法负担庞大的数据量，因此确定性的终结性可以减少数据的存储。最后，考虑到未来的分片设计，数据分片需要终结性作为基础。因此我们采用基于投票机制的设计，保证交易的终结性。

对于投票协议，除了上述特性之外，我们认为其需要考虑如下因素：

- 投票效益：通常而言区块链共识的投票是根据参与者的持有（或抵押）的资产计算所得，在没有投票成本的情况下，验证者更倾向分散其投票效益。一种解决方法是引入投票成本，或者对于分散投票行为作出惩罚 [8]。此外，投票效益的计算必须能够抵抗女巫攻击（Sybil Attack）。

[7]类似于定义7，此处 V 可为有限或无限集合。
[8]部分研究中将可终止性和有效性描述为活性（Liveness）。

- 委员会选举：为防止贿选以及共谋现象（Colluding），投票前投票节点身份不应曝光，同时在每次投票后（例如公布对某个区块的签名公证），其将丧失委员会资格，直至再次入选委员会。
- 延迟：尽管任何共识机制都需要保证在有限时间内结束决策，但我们仍希望投票能够快速达成一致，从而减少交易的确认延时（confirmation delay）。

参考文献

[1] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 4, no. 3, pp. 382–401, 1982.

[2] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," Journal of the ACM (JACM), vol. 27, no. 2, pp. 228–234, 1980.

[3] M. Castro, B. Liskov, et al., "Practical byzantine fault tolerance," in OSDI, vol. 99, pp. 173–186, 1999.

[4] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16), pp. 45–59, 2016.

[5] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in 25th {USENIX} Security Symposium ({USENIX} Security 16), pp. 279–296, 2016.

[6] S. Nakamoto et al., "Bitcoin: A peer-to-peer electronic cash system," 2008.

[7] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," self-published paper, August, vol. 19, 2012.

[8] V. Buterin and V. Griffith, "Casper the friendly finality gadget," arXiv preprint arXiv:1710.09437, 2017.

[9] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in Proceedings of the 26th Symposium on Operating Systems Principles, pp. 51–68, ACM, 2017.

[10] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 66–98, Springer, 2018.

[11] T. Hanke, M. Movahedi, and D. Williams, "Difinity technology overview series–consensus system (rev. 1)," 2018.

[12] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," ACM Computing Surveys (CSUR), vol. 22, no. 4, pp. 299–319, 1990.

[13] L. Lamport et al., "Paxos made simple," ACM Sigact News, vol. 32, no. 4, pp. 18–25, 2001.

[14] M. Burrows, "The chubby lock service for loosely-coupled distributed systems," in Proceedings of the 7th symposium on Operating systems design and implementation, pp. 335–350, USENIX Association, 2006.

[15] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," ACM Transactions on Computer Systems (TOCS), vol. 26, no. 2, p. 4, 2008.

[16] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in International Conference on Financial Cryptography and Data Security, pp. 507–527, Springer, 2015.

[17] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: speculative byzantine fault tolerance," in ACM SIGOPS Operating Systems Review, vol. 41, pp. 45–58, ACM, 2007.

[18] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in 2018 IEEE Symposium on Security and Privacy (SP), pp. 583–598, IEEE, 2018.

[19] H. van Zanten, "An introduction to stochastic processes in continuous time," Lecture notes, 2013.