

# DAO 治理

登链社区 - Tiny熊

# 要点

- 什么是 DAO 及 DAO 治理?
- 如何实现 DAO 链上治理
- SnapShot 链下投票

# DAO

- 什么是 DAO: Decentralized Autonomous Organization 去中心化自治组织
- 以太坊 \ MakerDAO (SKY) \ CityDAO \ 众多 DEFI 项目
  - 全新的组织形态: 去中心化, 自治, 可治理, 公开透明, Token激励
- 社群组织 DAO: 围绕一个共同目标 (愿景、纲领) 而组成的团体
  - DAO 的公开透明, 更容易吸纳贡献者, 形成大规模协作
  - Developer DAO、 SeeDAO、 GCC

# DAO 案例

- 以太坊：
  - <https://ethereum-magicians.org/>
  - <https://github.com/ethereum/EIPs>
  - 讨论会议：<https://github.com/ethereum/pm>
- DEFI：Uniswap
  - <https://snapshot.box/#/s:uniswapgovernance.eth>
- GCC：捐赠基金
  - <https://snapshot.box/#/s:gccofficial.eth>
- 登链黑客松投票
  - <https://snapshot.box/#/s:openspacebuilder.eth>

# DAO 治理

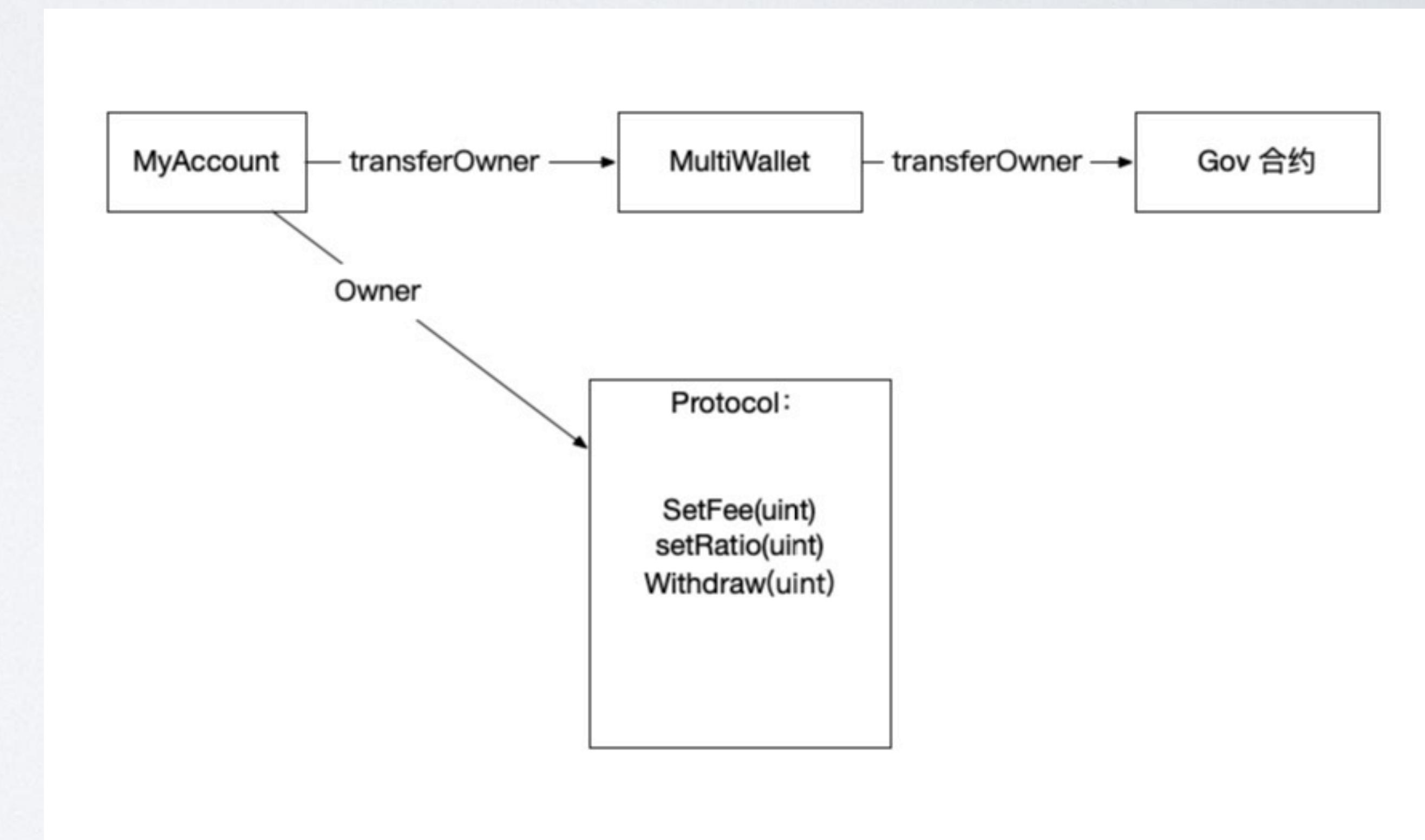
- DAO 治理：怎样的方式来管理 DAO
  - 没有权利中心，决策由集体做出
  - 信奉“Code is law”，通过合约实现管理的代码化（根据代币投票）
  - Defi 带动了 DAO 的实践

# DAO 治理

- DAO治理内容：
  - 资金分配
  - 参数调节
  - Bug 修复
  - 系统升级
  - 发展方向

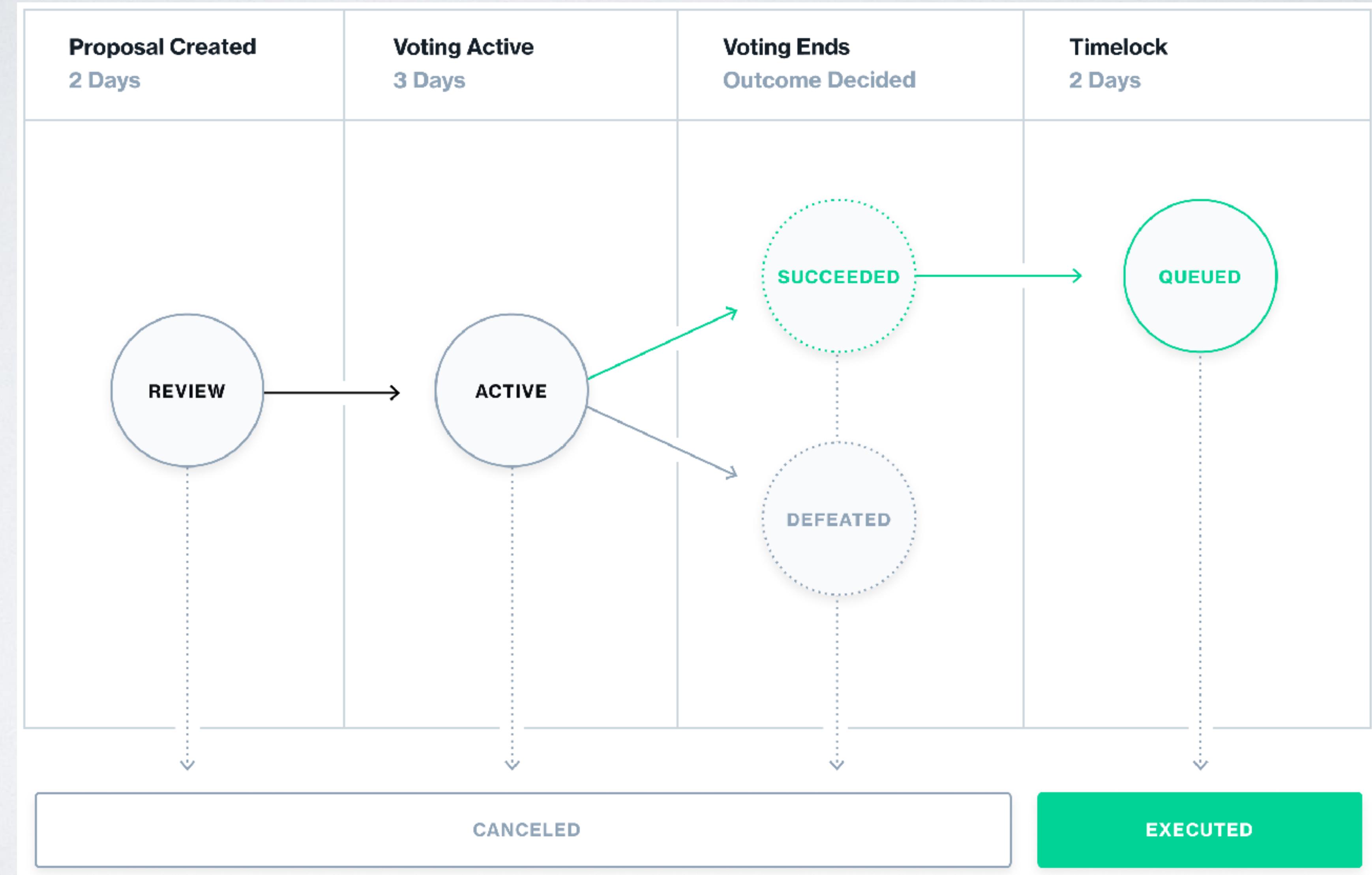
# DAO 治理

- 治理过程：
  - 提案、投票、执行（若通过，通常延迟执行）
- DAO治理方法：
  - 专制：管理员、多签方式
  - 治理代币投票 + 链上执行
  - Snapshot：链下投票、无需 GAS



# 投票治理

- 治理过程



# 链上治理 – 提案

- 提交执行的动作（调用哪个合约的哪个方法）
- 使用底层调用：可以执行所有可能的函数调用。
- 治理合约保存要调用的 calldata 、 target 等信息

如调用合约的setFee(uint256) 方法

```
bytes memory payload = abi.encodeWithSignature("setFee(uint256)", 10);
(bool success, bytes memory returnData) = address(target).call(payload);
require(success);
```

# 链上治理 - 提案

```
struct Proposal {
    uint id;
    address proposer;
    uint eta;

    address[] targets;
    uint[] values;
    string[] signatures;
    bytes[] calldatas;

    uint startBlock;
    uint endBlock;

    uint forVotes;
    uint againstVotes;

    bool canceled;
    bool executed;

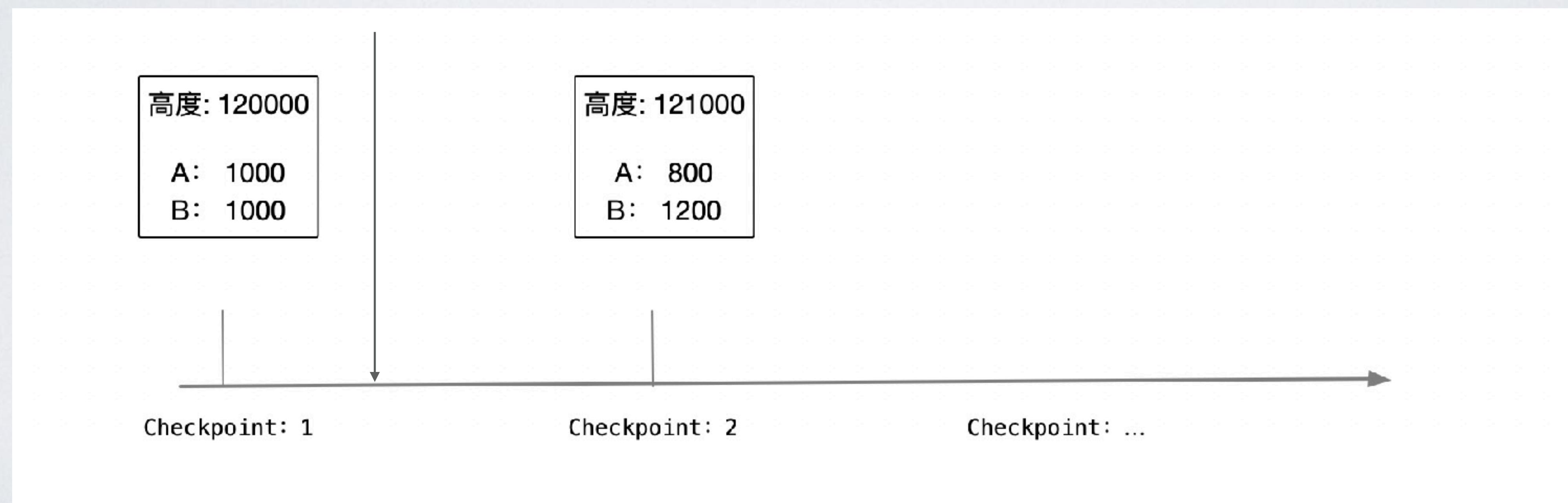
    mapping (address => Receipt) receipts;
}
```

# 链上治理 – 投票

- 问题：用户会频繁转账，如果防止转账后，重复投票？

# 链上治理 – 如何计票？

在每次转账的时候，记录一个检查点，检查点上写入区块高度，及对应的票数。



投票时，定位的提案区块高度在哪个检查点上，从对应的检查点上，获取其对应的票数。

# 链上治理 – 如何计票？

```
struct Checkpoint {  
    uint32 fromBlock;  
    uint96 votes;  
}  
  
/// 每个检查点对应的投票数  
mapping (address => mapping (uint32 => Checkpoint)) public checkpoints;  
  
///每个账号有多少个检查点  
mapping (address => uint32) public numCheckpoints;  
  
function getPriorVotes(address account, uint blockNumber) public view returns (uint96) {  
    // 采用二分法  
}
```

<https://github.com/compound-finance/compound-protocol/blob/master/contracts/Governance/Comp.sol>

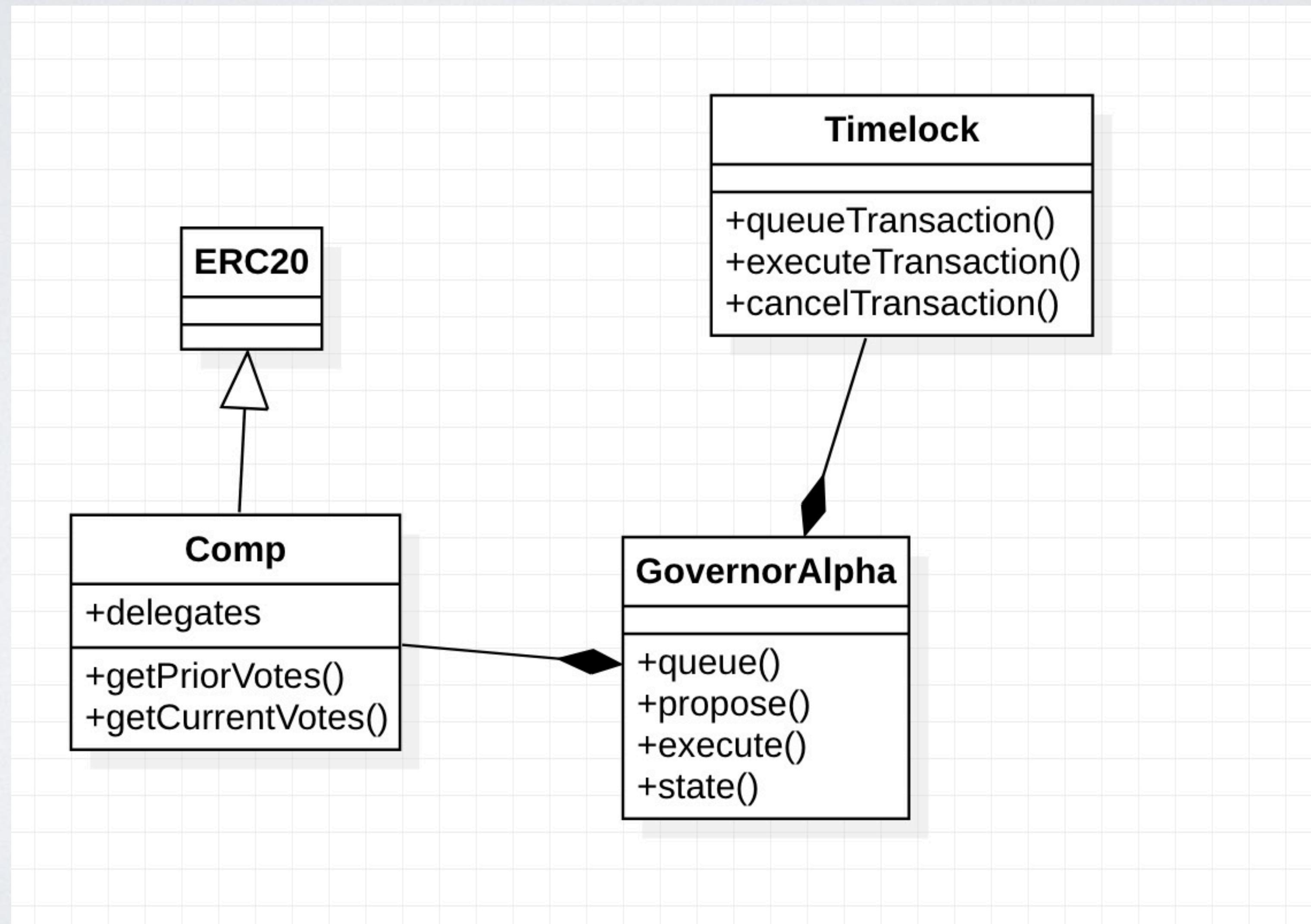
# 链上治理 - 执行

- TimeClock : 延迟执行
  - 提供额外的时间来模拟执行、审查代码，发现执行可能潜在的问题，必要时紧急采取行动。
  - 给执行提案不利的一方有时间窗口退出。

<https://github.com/compound-finance/compound-protocol/blob/master/contracts/Timelock.sol>

# 投票治理

<https://github.com/compound-finance/compound-protocol>



# 链下治理 – Snapshot 投票

- 工作原理：
  - 链下快照所有用户在每个区块高度的余额 (Token、NFT)
  - 创建提案：提案内容 + 指定投票的时间和规则
  - 用户链下签名投票，确认每个用户的投票权。
  - 链上执行（可选）：如多签团队在链上执行决策

<https://snapshot.box/#/>

<https://github.com/snapshot-labs>

# 练习题

- 实现一个可以计票的 Token
- 实现一个通过 DAO 管理资金的 Bank:
  - 管理员可以从 Bank 合约中提取资金 withdraw()
  - 治理 Gov 合约作为管理员, Gov 合约使用 Token 投票进行治理
  - 通过发起提案从 Bank 合约资金

<https://decert.me/quests/4cbe2544-6848-4881-b2f5-c4f291241621>