理解 SOLANA 的运行

登链社区 - Tiny熊

OUTLINE

- 理解 Solana 共识（POS）
  - 如何利用 POH 快速出块
- 了解 Solana 核心概念
  - 账户模型、PDA、交易及费用、集群等

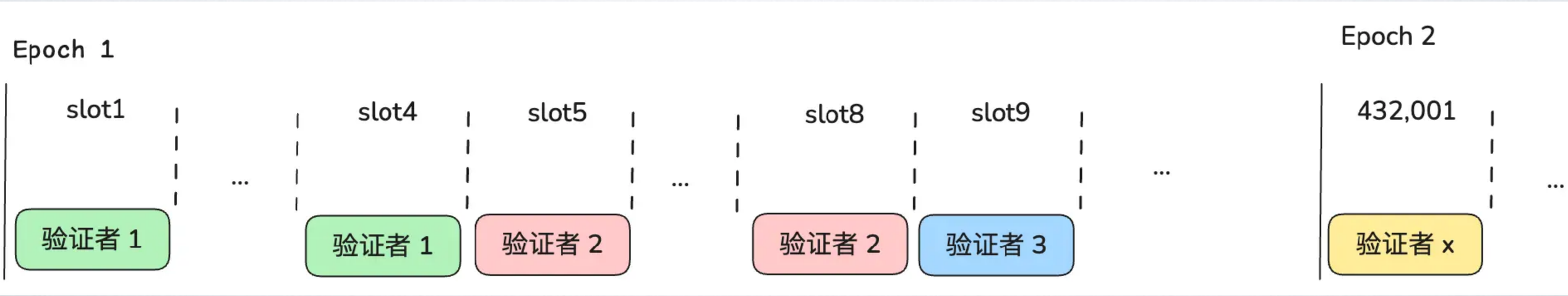
理解 SOLANA 共识

POS 共识

- Solana 是一条权益证明（PoS - Tower BFT）区块链
- POS 基本工作方式：
  - 1. 质押原生代币（Sol）， 成为一个验证者
  - 2. **被随机选择的“幸运的验证者” 出块**
  - 3. 其他的验证者对块投票见证（选择最“重”分叉）
  - 4. 累计了足够多的投票后，成为共识块(Finality)

选出验证者

- 每个 Epoch 开始时，会根据质押产生质押权重和之前的区块随机选举出一个验证者序列
- 每个验证者处理 4 个 Slot，在验证者序列确认时，验证者就会知道他需要处理哪几个 slot



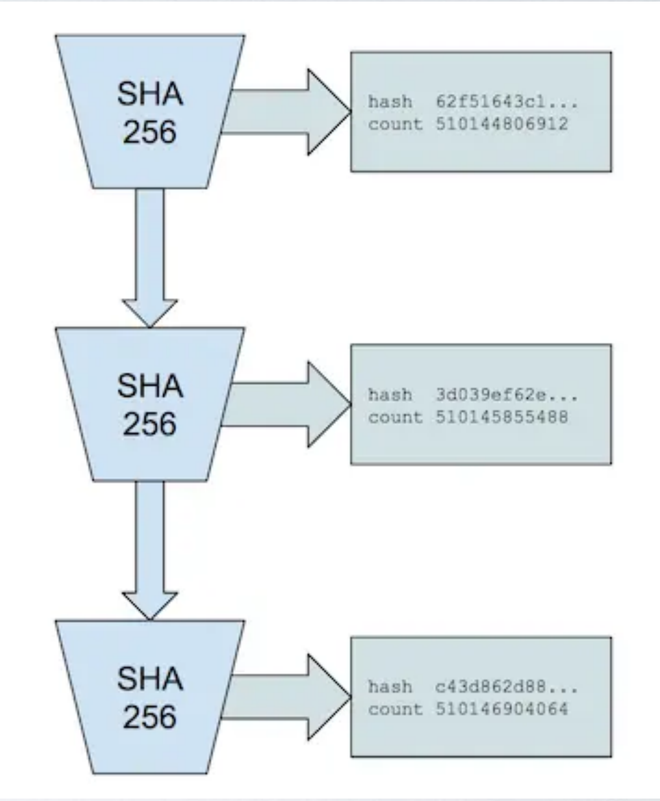
验证者什么时候出块?

- (关键问题）验证者怎么知道该轮到自己出块了?
  - 出块时间窗口只有 0.4s
  - 依靠网络通信么？如果网络延迟或上一个验证者掉线了?

验证者不能干等～～～

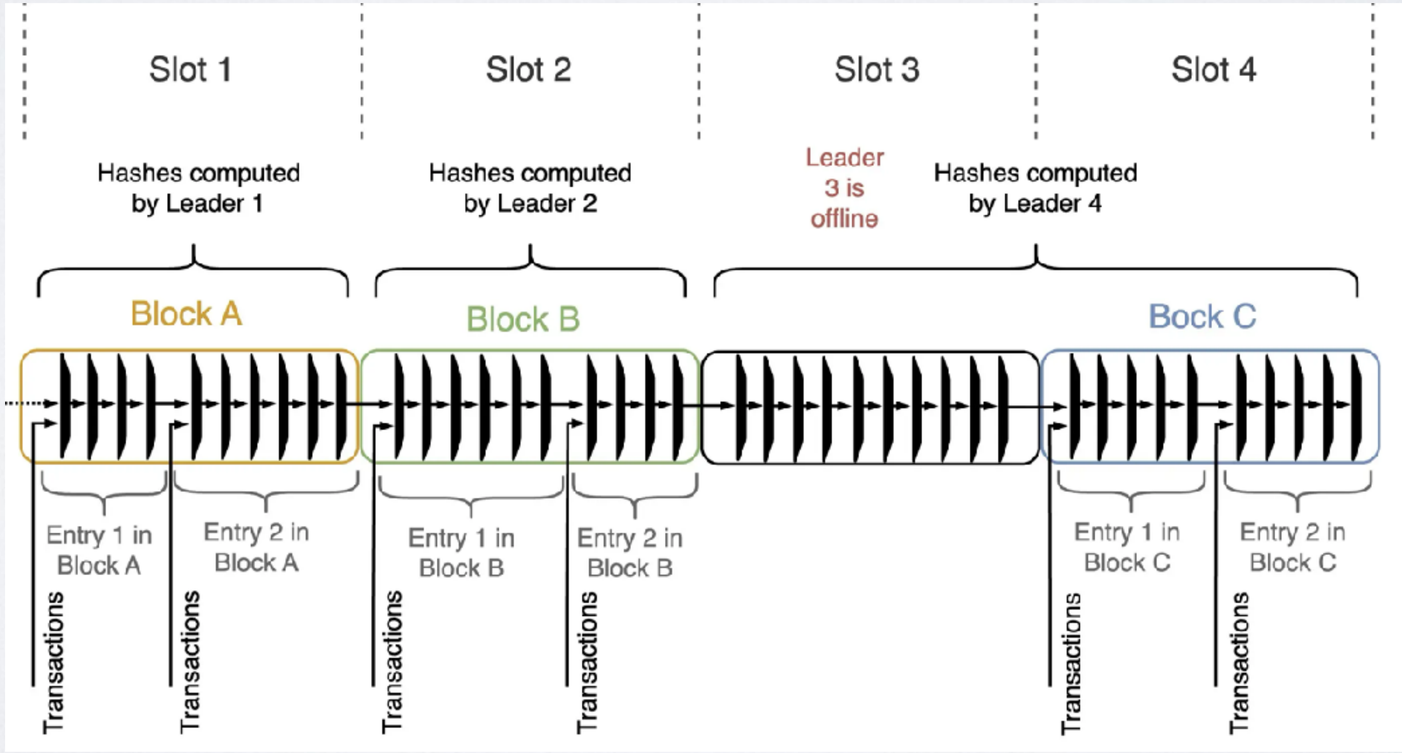
POH 出场

- POH （Proof of History)：是一个 hash 链
- 每次哈希计算（SHA256）都需要使用前一个哈希值，无法并行
- 每次哈希操作都是需要最小时间，应该 POH 可以作为时间流逝的证明



POH

- Solana 中，要求每个区块必须包含 12,500 个哈希（POH），由验证者生产。
- 每个验证者都在自己的本地创建自己的 POH，如果没有及时收到上一个验证者的 POH 序列，就基于自己的POH 序列出块。

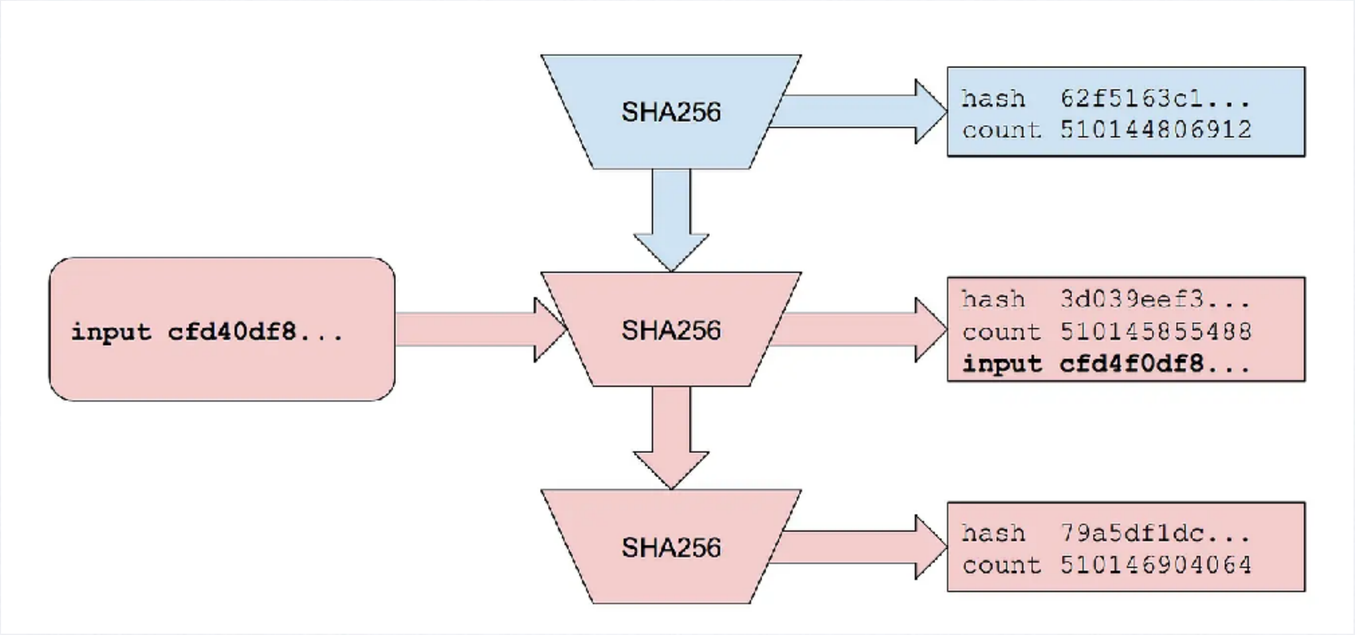


（领导）验证者出块

- **领导验证者**不停从 RPC 服务器和其他的验证者那里接收交易
- 在一个 Slot 里，打包尽可能多的交易，使用流水线并行处理交易
- 交易有效性验证、POH排序、执行、广播，相互衔接、并行重叠执行。

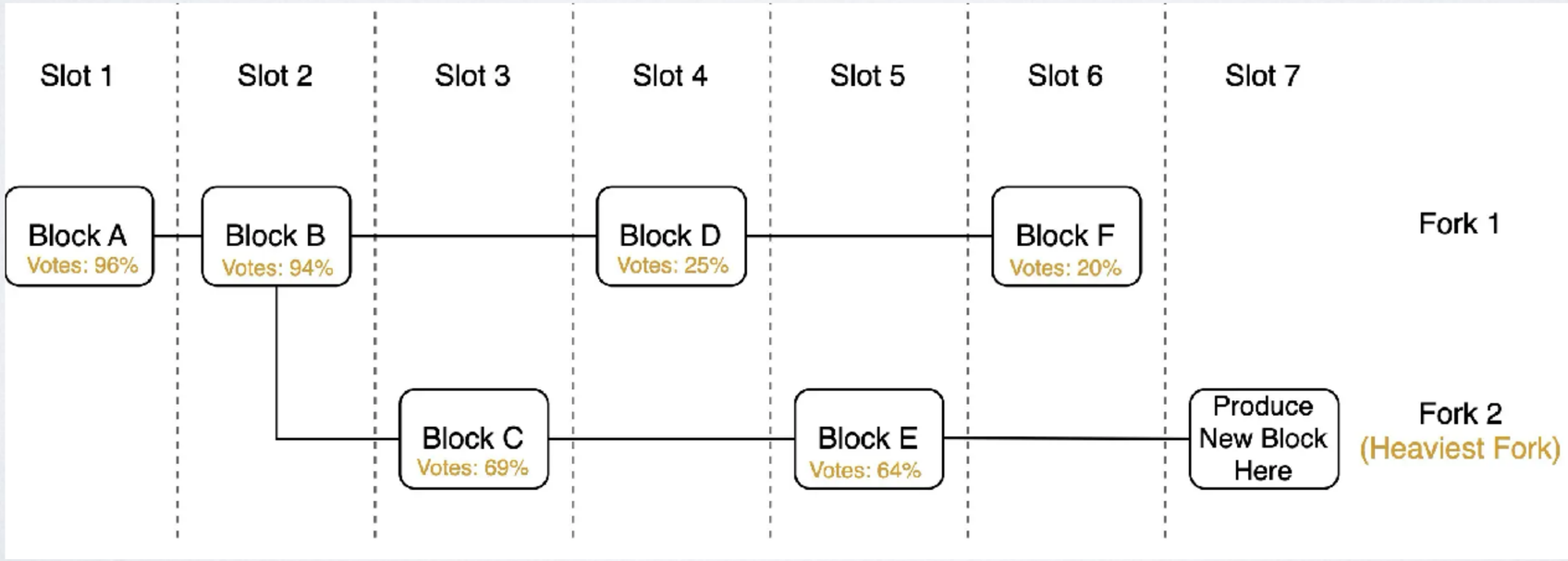
POH 交易排序

- 每次哈希计算都需要使用前一个哈希值，本身就是有序的。
- Hash 时，加入交易数据就可以确定交易次序



（其他）验证者验证区块

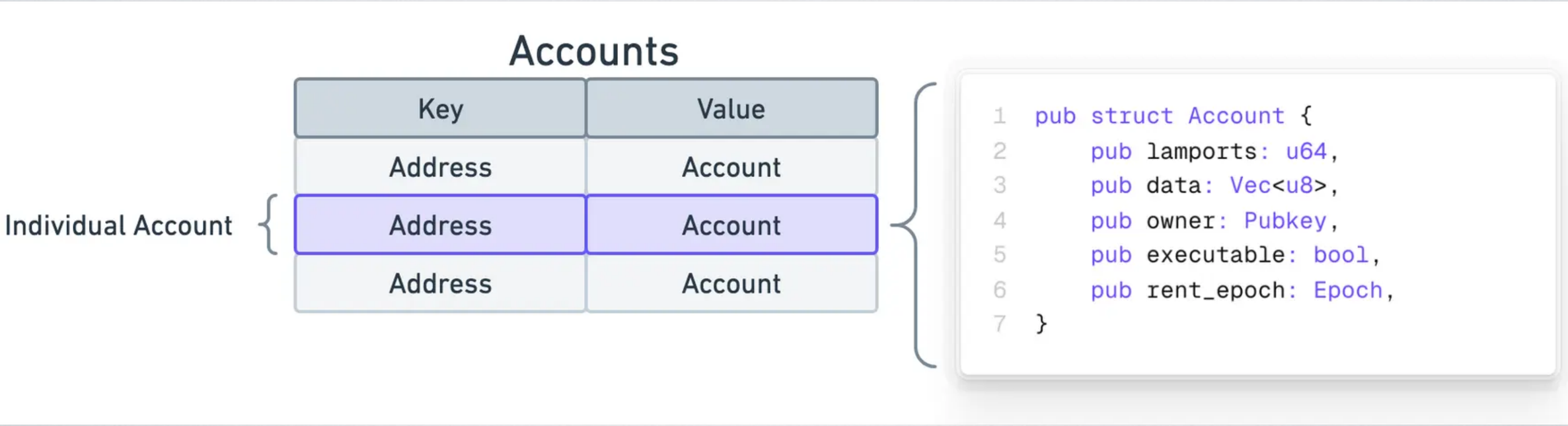
- 验证： PoH 、区块元数据（slot、leader 等）、重放交易
- 签名投票表示对区块的确认，选择**最重**的链出块和投票



SOLANA 核心概念

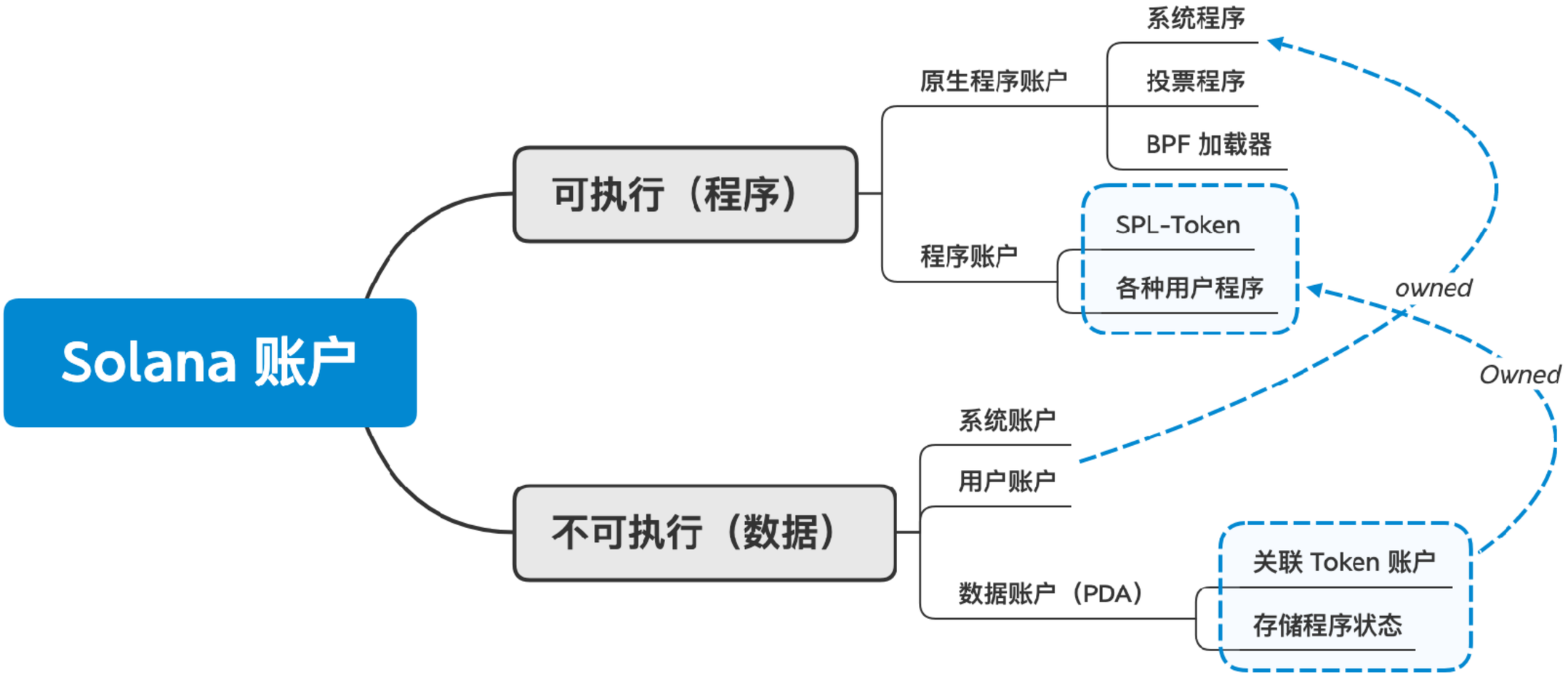
账户模型

- Solana 的账户是一个存储单元（可以存数据，也可以存程序），Linux 的文件很类似
- 用地址表示：Ed25519 公钥的 Base58 编码格式字符串表示， 如：
  `14grJpemFaf88c8tiVb77W7TYg2W3ir6pfkKz3YjhhZ5`
- 最大存 10 M 数据，空间越大，需要更多的存储费用（租金），Owner 可以才可以修改账户数据或减余额



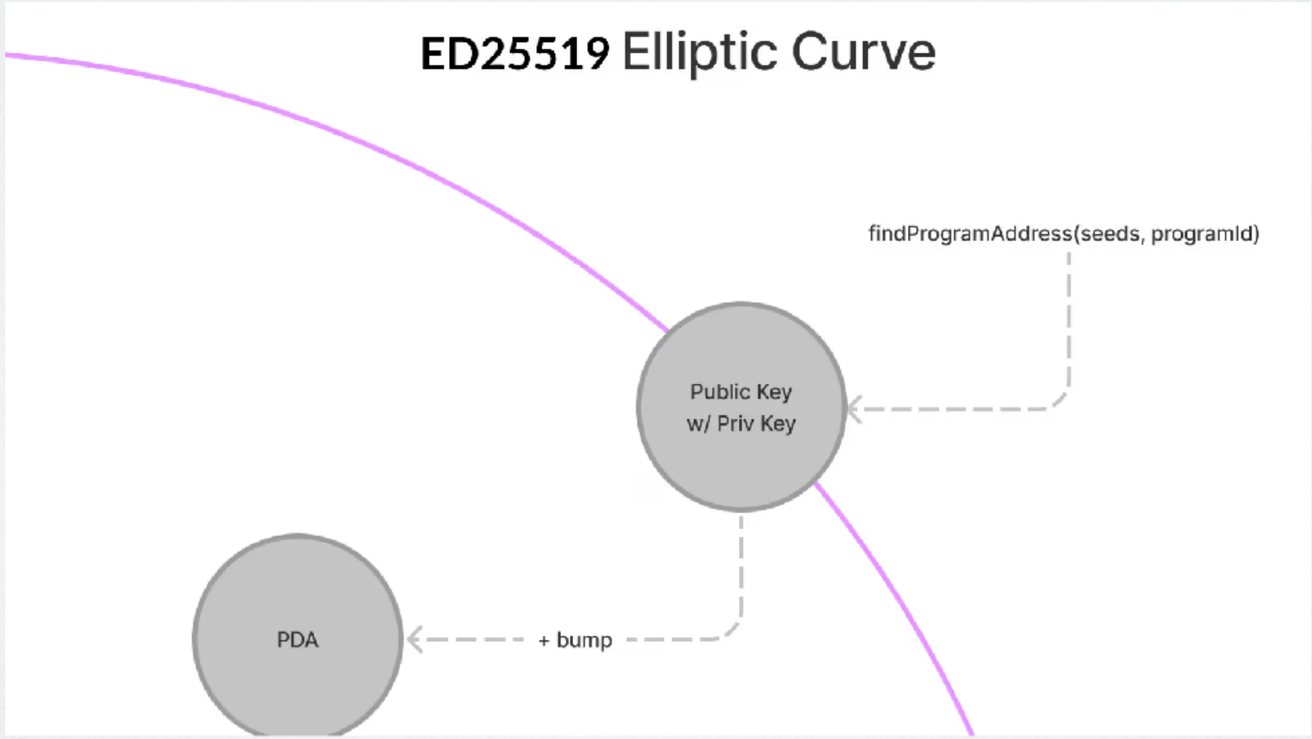
账户模型 -> 编程模型

- 账户（按能否执行）：程序账户和数据账户
- 编程模型：代码和数据的分离
- 利：程序是无状态的，可并发执行
- 弊：程序操作的数据账户，需要外部传递（通过交易）


Solana 账户
可执行（程序）
原生程序账户
系统程序
投票程序
BPF 加载器
程序账户
SPL-Token
各种用户程序
owned
Owned
不可执行（数据）
系统账户
用户账户
数据账户（PDA）
关联 Token 账户
存储程序状态

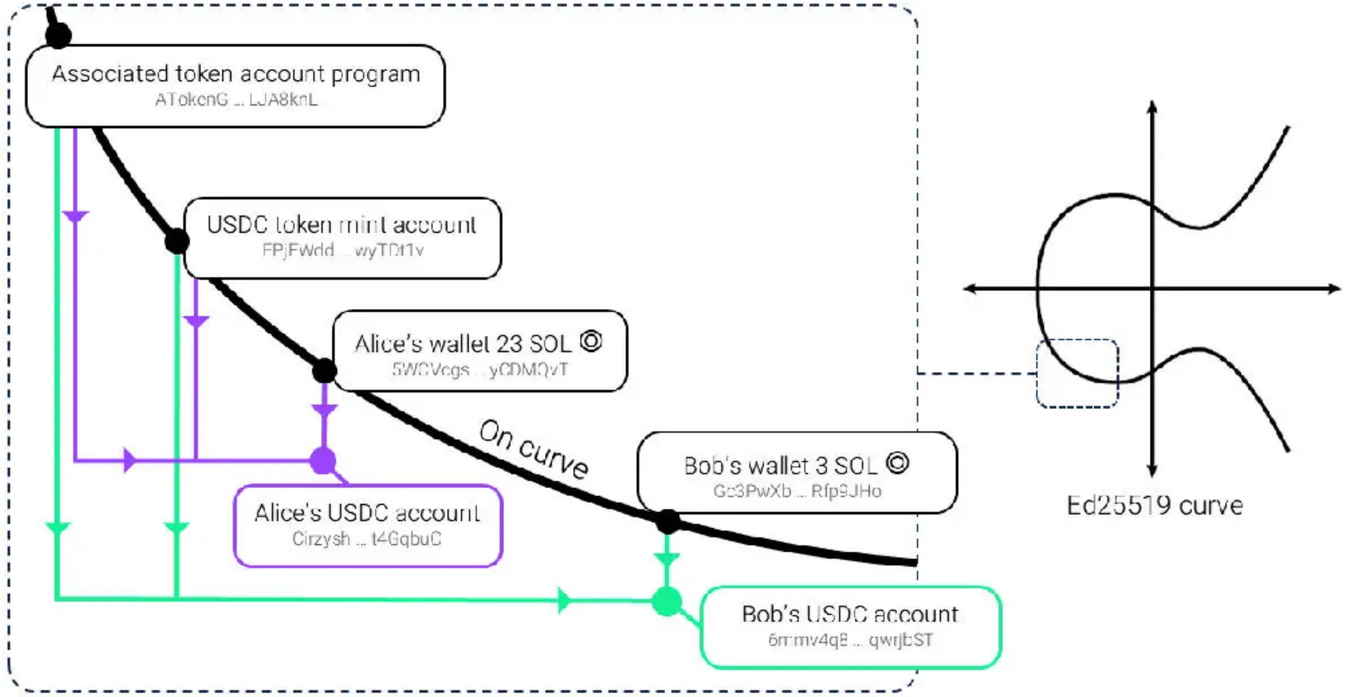
PDA

- PDA： Program Derived Addresses(程序衍生地址), 保存程序数据的账户
- 是在曲线外的生成的账户： 根据 **程序** （owner） **+ 种子 + bump** 生成



SPL-TOKEN / ATA

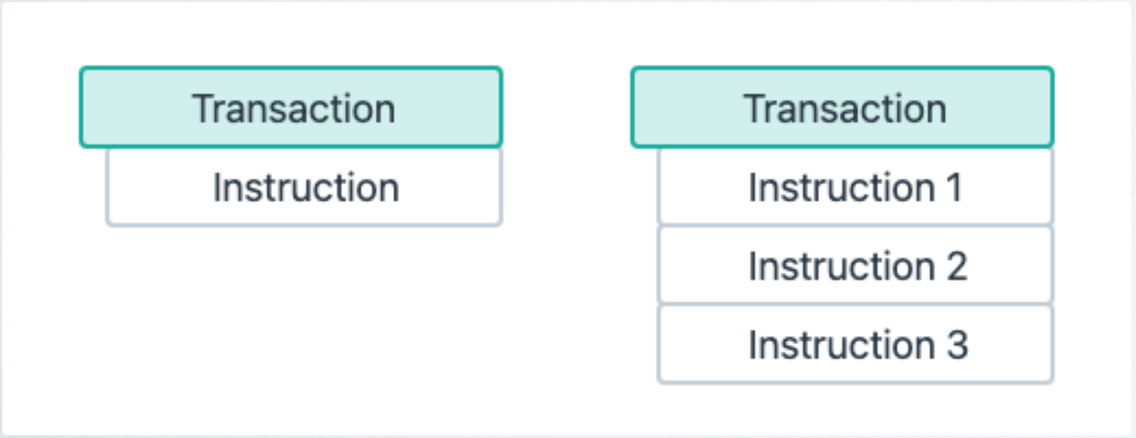
- Token 逻辑都是一样的，可以共用程序（SPL-Token、SPL-Token 2022）
- Mint Account： 指定（存储） Token 的发行量、精度、铸币者 （name、symbol）
- ATA（Associated Token Account ）： 存储着 Token 的余额

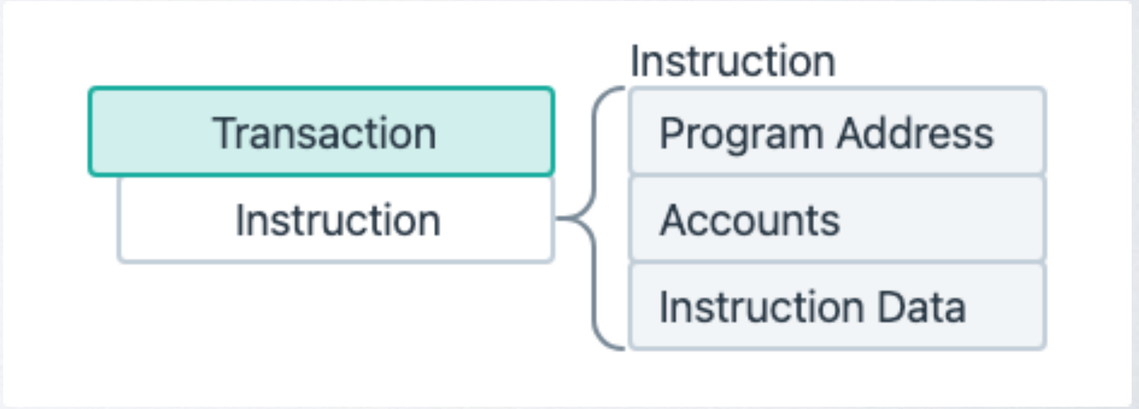



```
const [ata, bump] = PublicKey.findProgramAddressSync(
  [
    wallet.toBuffer(),
    TOKEN_PROGRAM_ID.toBuffer(),
    mint.toBuffer(),
  ],
  ASSOCIATED_TOKEN_PROGRAM_ID
);
```

交易

- 交易是指令的封装，指令包含: **执行程序、涉及的所有账户和操作数据**
  - 指令类似于以太坊智能合约上的函数调用。
  - 以太坊的 EOA 一次只能发起一个调用，Solana 可以一次发起多个指令 （原生支持 multicall）
- 交易的执行是原子性的，所有指令要么一起成功，要么一起失败。

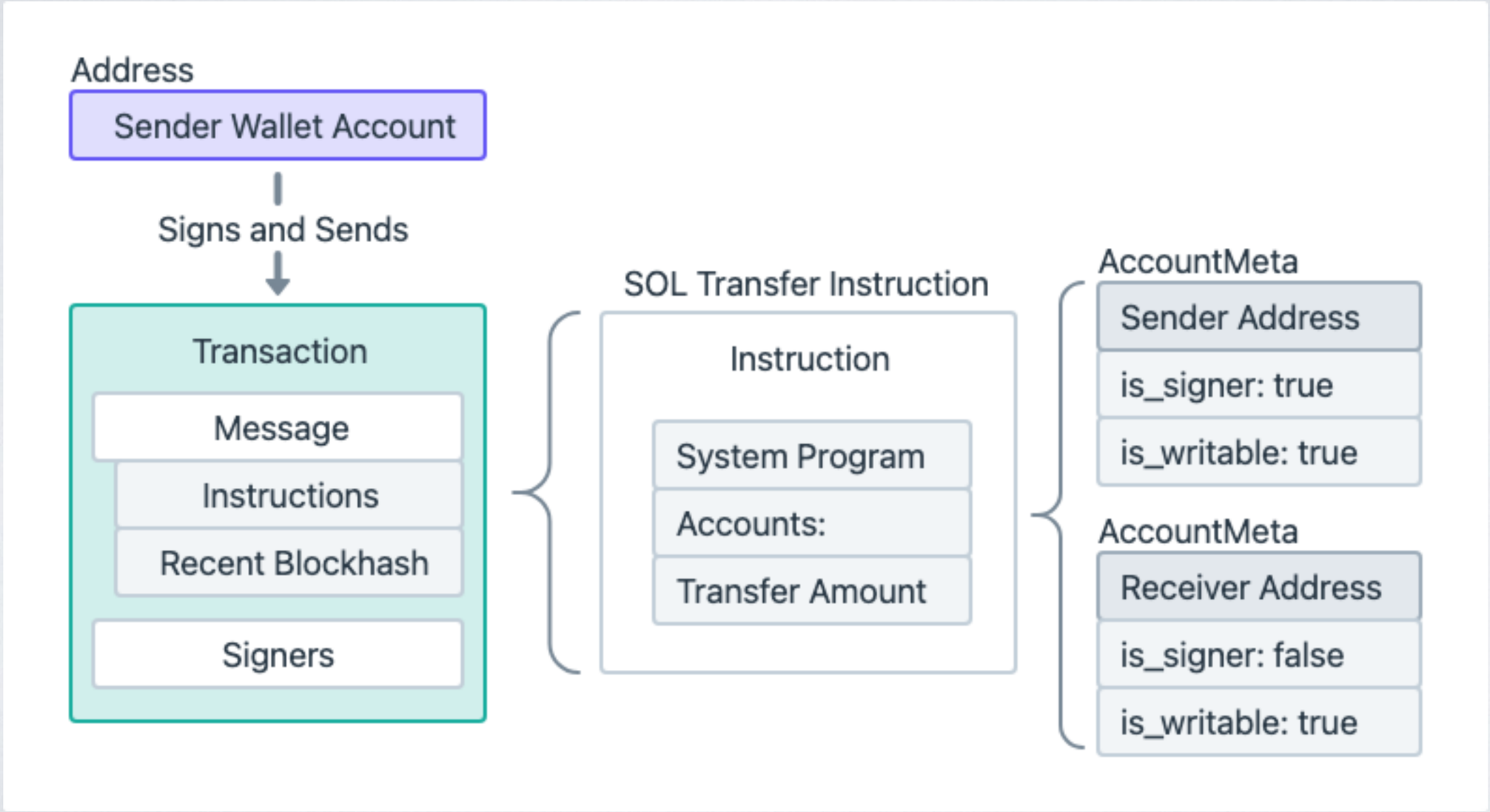




交易结构

- 交易还包含签名数据
  - 交易至少包含一个 fee payer 用户账户签名（PDA 账户的数据修改由主程序负责，不需要额外签名）
- 交易指定 recent_blockhash
  - recent_blockhash， 起到类似交易有效期的作用， 150 区块有效期（约 1 分钟）
  - 没有类似以太坊的 nonce
  - 节点会验证 (signer, recent_blockhash) 组合是否已经出现过（即是否已执行过）
- 如果要发离线或延时交易，可使用 Durable Nonces（持久 Nonce）

交易

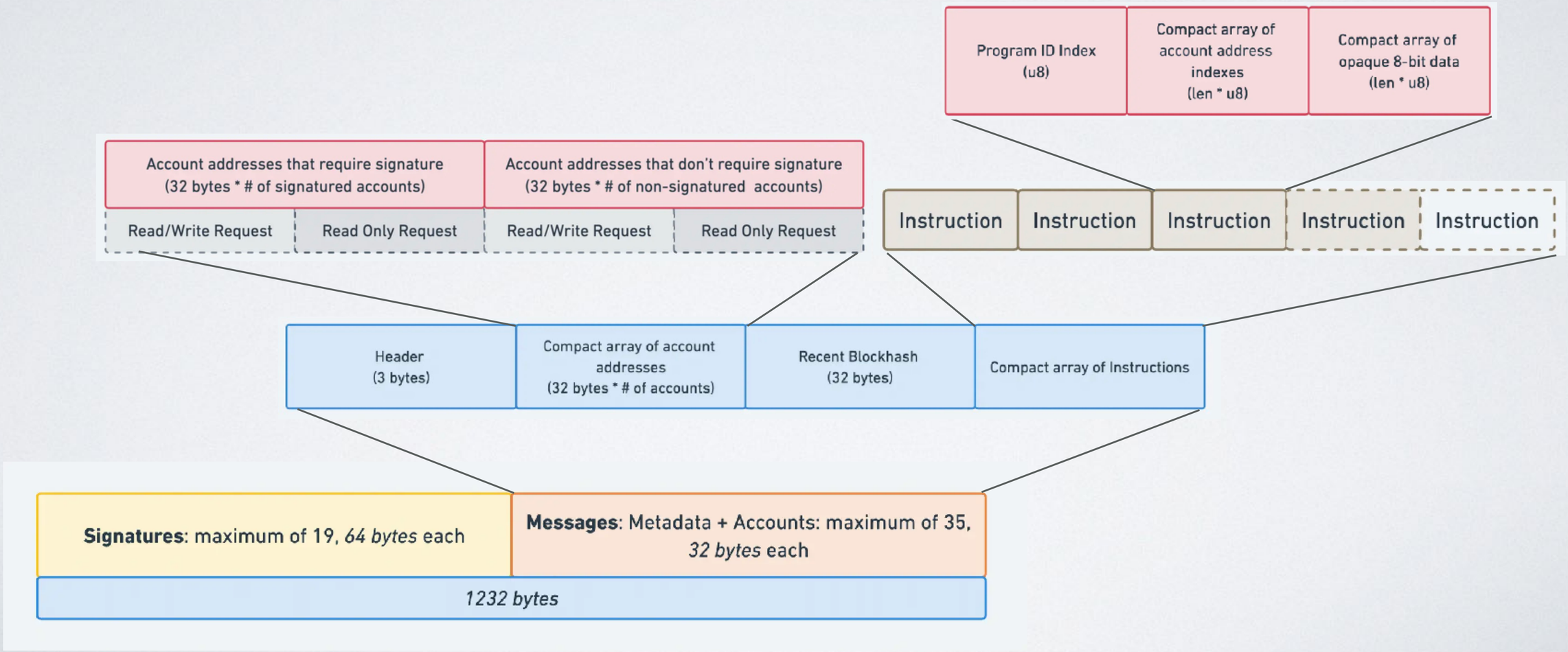

Address
Sender Wallet Account
Signs and Sends
Transaction
Message
Instructions
Recent Blockhash
Signers
SOL Transfer Instruction
Instruction
System Program
Accounts:
Transfer Amount
AccountMeta
Sender Address
is_signer: true
is_writable: true
AccountMeta
Receiver Address
is_signer: false
is_writable: true

交易费用

- 基础费用(50% 给验证者、50% 销毁)
  - 基础费用（Base fee）是按交易中包含的签名数量收取的。
  - 每个（交易）签名 5000 lamports （ 1 Sol = 10 亿 lamports ）
- 优先级费是可选的（全部给验证者）：
  - 设置每计算单元（CU）费用， 如：0.02 lamports/ CU （以 micro lamprots 来设置）
  - 每个指令默认是 200,000 CU （最大可设置为140 万）， 则 0.02 * 200,000 = 4000 lamports
- 手续费： 基础费用 + 优先级费（可选）+ 创建账户的存储费（可能有）
  - 存储费（最小免租费用）： 账户存储大小 * 每字节每年成本（SOL/ 字节）* 2

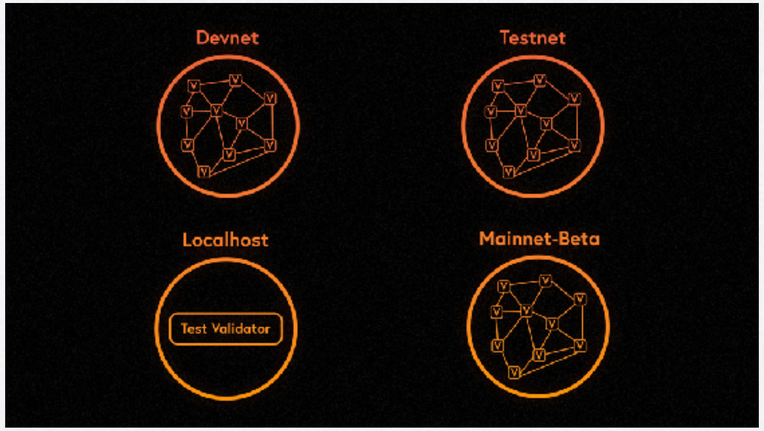
交易大小

- Solana 交易包大小限制在 1232 字节。（IPv6 MTU = 1280 bytes）



集群（CLUSTER）

- 每个集群有自己独立的验证者和账本
  - 相当于以太坊不同的网络
- 有：Mainnet-Beta、Testnet、Devnet、Localnet



钱包与浏览器

- 钱包：Phantom、Solflare、Backpack、OKX ...
- 浏览器：
  - https://explorer.solana.com/
  - https://solscan.io/
  - https://solana.fm/
  - https://orb.helius.dev/
  - https://solanabeach.io/
  - 多链支持：https://xscan.io/ https://www.oklink.com/sol https://blockchair.com/solana

作业

- 安装 Phantom 钱包、领水、发起交易
- 安装 Solana 开发环境 (anchor / @solana/web3.js / @solana/kit)

https://decert.me/quests/73599182-7447-4d78-b3b6-654221987ebf

参考文章

- 理解 Solana
- Solana 是如何工作的
- 深入 Solana 共识 - 从分叉到最终确定性
- Solana 文档 - 账户