公链速度之王-Solana

目录

CONTENTS

北京大学
PEKING UNIVERSITY

PART 01

区块链结构及其设计原理

1.1 银行总账和区块链总账

银行的做法有几个重要逻辑：

1. 有一笔总账，记录着每一个人的账本，以供查询并对账；
2. 这笔总账只有银行有权利记账（银行是有公信力的机构）；
3. 每一个人的账本都详细记录着每一笔交易、相应的时间戳、余额等信息；
4. 只有当一笔交易被“确认”后，才会作为收入/支出记录在账本中。

1.1 银行总账和区块链总账

参考上述银行的做法便可得知：若要解决问题一“去中心化”，核心是有一个“总账本”并满足以下条件：

1. 公开透明的（中心化的情况下，因为银行保证了你账本的真实性，所以无需公开透明；但没有银行时，只有允许账本公开透明，才能使得卖家知道买家账上是否有足够的钱来支付款项；）

2. 可供所有人查询并对账的（中心化的情况下，银行保证了你账本是真实的；但没有银行时，只有允许其他人都可以查询且对账，才能确保买家账本的真实性，否则买家可能虚报自己的账本）

3. 记账权力是公平且安全的（记账权利是指谁可以有权利修改账本，中心化的情况下，银行只有在你存款了后才会在你的账本上记账，只有银行才有记账权利，但在没有银行时，谁有权利记账呢？必须保证修改账本的权利是公平且安全的，否则任何人都可以随意篡改自己的账本，例如自己给自己账本上+1亿元）

4. 记录了所有人已确认的交易记录（已确认交易是为了避免“双重交易”，如果某笔交易没有被确认却被记账了，会导致账本余额数量对不上。例如A账本有1.5万元，需要给B支付1万元，指令已发出但后面交易取消，如果此时依然记账，A账本就只有0.5万元，是错误的。中心化的情况下，取消交易后银行会调整账本，但没有银行时，谁有权利调整呢？）

1.2 在设计的过程中遇到的问题

1. 谁有权利记账？
2. 如果有记账人，如何保证该记账人不会为了自己的利益修改记录（即如何保证记账人有银行一样的公信力）？
3. 如何保证账本不被篡改，例如修改以前的历史记录（即如何保证账本的安全性）？
4. 谁有权力对交易进行确认和验证（即证明各笔交易是否真实有效）？

1.2 在设计的过程中遇到的问题

解决办法：所有人都可以记账，任意一笔记账都需要绝大多数人进行确认和验证，且只有在绝大多数人同意的前提下才能被认为是真实有效的（即达成一致consensus）。

我们可暂且将技术原理放在一边，如果这是能做到的，那么问题1，2，和4也就解决了。
一：所有人都有记账权利，则解决了1。
二：因为记账需要大多数人同意，所以不会存在记账人为了自己利益修改记录的行为，因为这样的行为会被大家否认，则解决了2。
三：每一笔交易都确保了真实有效，因为每一笔交易都需要大家的确认和验证，则解决了4。

1.2 在设计的过程中遇到的问题

在这个前提下，我们又该如何解决问题3，即如何保证历史记录不被篡改呢？

1.2 在设计的过程中遇到的问题

保证历史记录不被篡改，有几个容易想到的思路：

第一种思路是：我们可以给历史记录加上一把复杂的”密码锁“，任何人要修改账本的历史记录需要解开这把锁。这个思路的前提是，这把锁不会妨碍人们对历史记录的查询和验证，且不会妨碍人们添加新的交易记录（即新的记账）。**可是，这把锁的钥匙、密码归谁保管呢？**

第二种思路是：**但凡有人对历史记录的进行了修改，哪怕只是一丁点儿修改，都会引起一个较大的变化，以至于被人们发现且拒绝。**

第三种思路是：对历史记录的修改的需要花费的成本巨大，且随着时间的推移越来越大，大到普通的人无法承受。**这个思路的缺陷在于，你永远都不知道富人究竟有多富。**

1.3 哈希函数和公开总账本的安全性

假设某一笔交易的历史记录我们设为自变量key1，哈希函数的上述性质告诉我们，很难找到一个不等于key1的另一笔交易记录key2，使得h(key1)=h(key2)。这也意味着：

（1）对于key1的任何变动，哪怕是一点点，都将使得哈希值/输出值与原输出值h(key1)完全不相等或者差异巨大；这就是说交易记录的一点点改变，哈希值的变化都会足以引起人们的注意。

（2）如果后一个哈希函数的输入是基于前一个哈希函数的输出，那么，如果要修改任何一个时点的历史记录，就要把前面所有时点的历史记录及其对应的哈希值都进行修改，并找到相应的哈希碰撞，而哈希函数的性质告诉我们，这在计算上是相当难实现的。换句话说，如果哈希函数的自变量不是交易记录key1，而是交易记录的哈希函数h(key1)，即自变量x1 = h(key1), 因变量y = h(x) = h[h(key1)]。此时，如果某人尝试把交易记录key1改变成key2，但又不想让别人发现，那么y = h[h(key1)] 就需要不变。也就是要使得y = h[h(key1)] = h[h(key2)]。而根据前面哈希函数的性质4“碰撞约束”：若想找一个与x1不一样的另一个自变量x2，使得h(x1)=h(x2)，在计算上不可行。

1.3 哈希函数和公开总账本的安全性

基于以上，我们很容易想到，这个性质如果能应用到我们前面提到的思路，那将是非常完美的。如果能把某一时点的交易记录进行哈希，且该哈希的输出是以后时点的交易记录哈希的输入，即h(x2) = h(h(x1))，那么不同时间点的交易记录之间将通过哈希进行紧密联系，任何一处的变动不仅将改变其自身的哈希，还会改变以他为参数之一的后面交易记录的所有哈希，而且根据哈希函数的性质，如果要找到另一个数使得后面的哈希不变，这在计算上是不可行的。也就是说，对历史记录的任何一点修改将随着时间的推移越来越难、成本越来越高，且在计算上无法篡改。

1.4 推理区块链结构

因此，我们可以推导出区块链结构的初步框架：

1. 新加入了交易记录后，新账本记作Bn，则Bn需要至少包含以下内容：

（1）Bn对应发生的时间，即“时间戳”，记为tn

（2）包含的所有交易记录，记为TXn

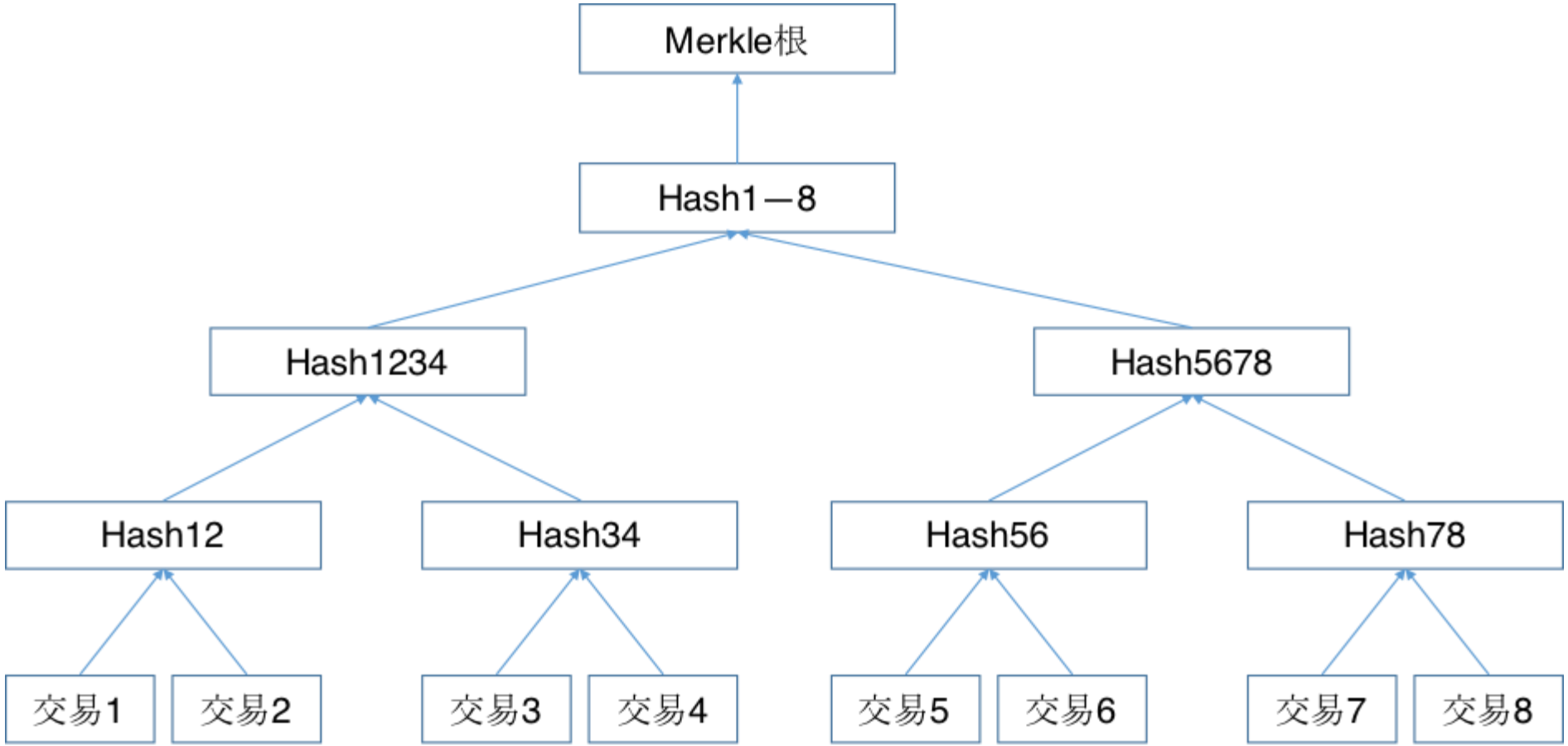
（3）上一个账本的哈希值，记为B(n-1)。

所以：Bn=H(tn, TXn, B(n-1))， 其中H(x)是哈希函数，是对自变量x进行哈希。所以Bn是对三个自变量tn, TXn, 以及B(n-1)一起进行哈希的结果。

2. 由于一个账本中可以包含很多的交易记录，为了让这么多交易记录变成一个自变量，从而简化交易记录在哈希函数中的输入，我们可以对交易记录进行哈希，也就是把很多的交易记录TX1，TX2， TX3.....进行哈希，变成一个输入值。

1.4 推理区块链结构

默克尔树（Merkle Tree）



1.4 推理区块链结构

由此可见，通过上述方法，我们把一个账本中包含的众多交易，通过不断的哈希，最后变成了一个变量，即Merkle根。我们只需要用Merkle根作为账本的一个变量，即可包含所有交易记录。

即：Bn=H(tn, Merkle-n, B(n-1))， 其中，Merlke-n=H(TXn)，所以Bn是对三个自变量tn, Merlke-n, 以及B(n-1)一起进行哈希的结果。

1.4 推理区块链结构

因此，第3个问题的解决方法为：利用哈希函数的性质，包含了新的交易记录的账本（区块）Bn=H(tn, Merkle-n, B(n-1))，新账本Bn和旧账本B(n-1)通过哈希函数紧密联系，对历史记录进行哪怕一丁点的修改，都会引起新账本Bn的值发生巨大变化，且随着时间的推移，要对账本进行篡改需要花费的成本越来越高，因此计算上可以认为账本不太可能被篡改。

不幸的是，在设计的过程中又出现了新的问题。即使是所有人都有权利记账，我们依然要解决每一笔账具体由谁来记的问题，否则如果大家都只是有权利、而没有义务去记账，最后的结果就是谁都不记账，陷入“三个和尚没水喝”的困境。

1.4 推理区块链结构

这时候就需要通过**激励**来引诱大家争相去竞争记账的权利，这就是**共识机制**的由来。

比特币区块链的做法是引入了一个“竞争机制”。通过在前述的函数Bn=H(tn, Merkle-n, B(n-1))中，引入一个随机数nounce，即**Bn=H(tn, Merkle-n, B(n-1), nounce)**，让所有人“公平”地去寻找这个随机数，使得Bn小于“目标哈希值”。谁先找到这个随机数，谁就获得了记账的权利。由于找到这个随机数在数学上的唯一方法只有通过穷举，因此谁的计算速度快、谁就先能找到这个随机数，获得记账的权利（因此，从某种意义来讲，这种记账权利并不是完全“公平”的，所以才会有诸如Proof of Stake等其他方法）。计算速度考验的是计算机的算力，而算力是耗费成本的，为了补偿使用算力造成的成本，这个账本会凭空产生出一个“代币”——虚拟货币“比特币”（Bitcoin）——发放给赢得记账权利的人，称为“旷工”，旷工通过消耗算力、找到随机数、赢得记账权利、并获得比特币被称之为“挖矿”，通过算力来进行挖矿的过程，被称为**工作量证明（PoW，Proof of Work）**。

基于此，我们进一步修改账本的哈希函数：

Bn=H(tn, Merkle-n, B(n-1), nounce)，

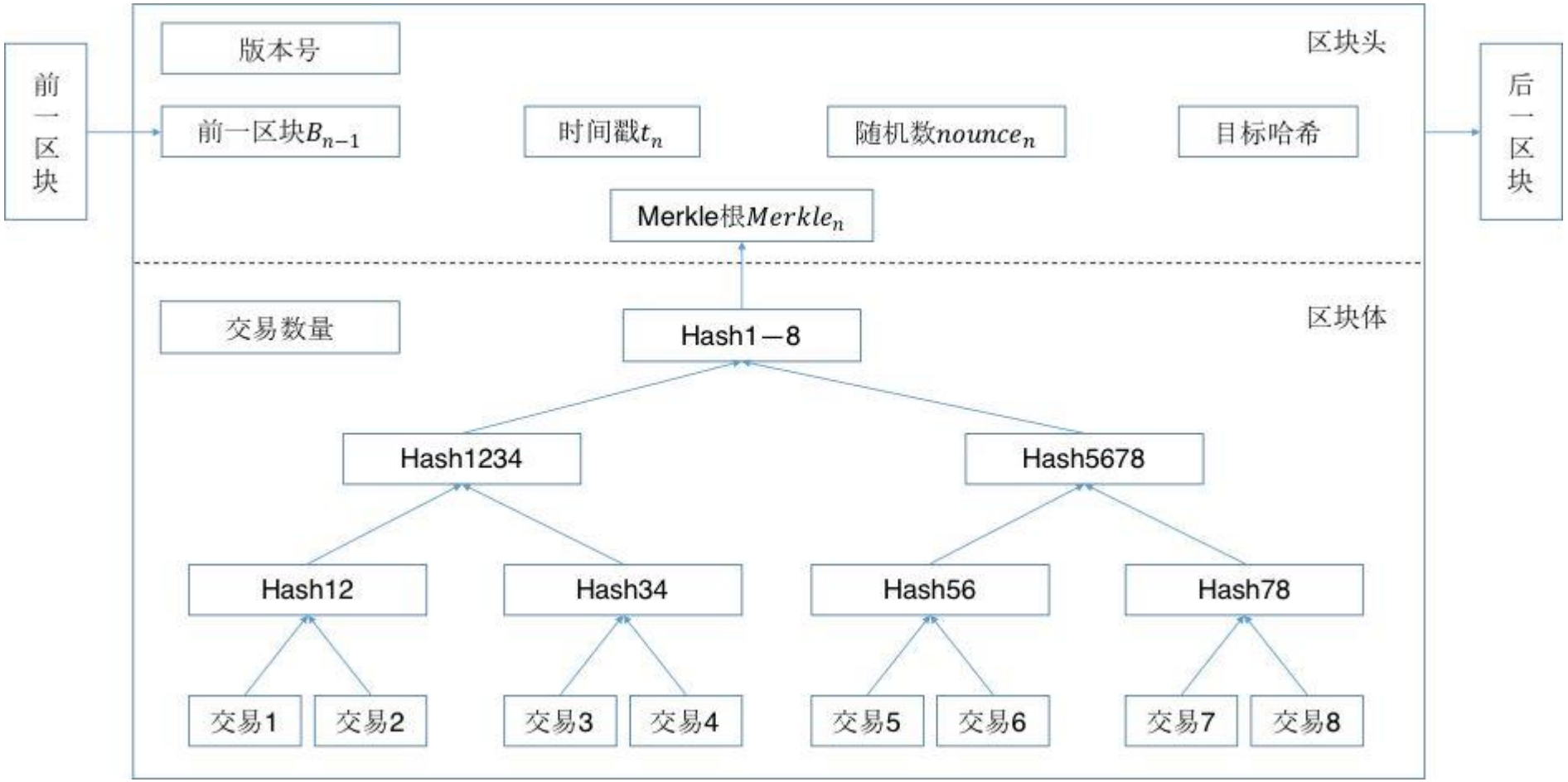
其中，Merlke-n=H(TXn)， nounce是随机数。旷工需要找到nounce，使得Bn<目标哈希。

1.5 区块链结构

基于前面的函数，我们知道一个新的区块/账本中，包含至少五个参数：**时间戳tn，前一个区块的哈希值B(n-1)，随机数nounce，目标哈希，以及Merkle根**，这五个参数将包含在“**区块头**”中（当然还包括一些其他参数，例如版本号，相关性较小我们此处暂时忽略）；

此外，我们知道，Merkle根其实是包含所有交易记录的一个哈希值，因此每一个区块中还有一个“**区块体**”，记录着所有的交易记录。清楚的区块链结构如下：

1.5 区块链结构


前一区块
版本号
前一区块B_{n-1}
时间戳t_n
随机数$nounce_n$
目标哈希
区块头
后一区块
Merkle根$Merkle_n$
交易数量
Hash1—8
区块体
Hash1234
Hash5678
Hash12
Hash34
Hash56
Hash78
交易1
交易2
交易3
交易4
交易5
交易6
交易7
交易8

1.5 区块链结构

北京大学 PEKING UNIVERSITY

Block #512352

Summary	
Number Of Transactions	1669
Output Total	9,018.00643299 BTC
Estimated Transaction Volume	980.87834681 BTC
Transaction Fees	0.25548139 BTC
Height	512352 (Main Chain)
Timestamp	2018-03-07 02:52:27
Received Time	2018-03-07 02:52:27
Relayed By	BTC.com
Difficulty	3,290,605,988,755
Bits	391481763
Size	1099.549 kB
Weight	3992.509 kWU
Version	0x20000000
Nonce	1152756558
Block Reward	12.5 BTC

Hashes	
Hash	0000000000000000000c0dc66a8437f7e2feae18366f1d34b79bce7d72bb33b2
Previous Block	0000000000000000000121461c750e76b0d7022d1aa4ae1c4aa485aba9d219b3
Next Block(s)	0000000000000000011d9deb41bfd546610de982fcbedfa251b52e2eec8719c
Merkle Root	9cfab47f8b1ae76a7ab4297bd552de368fa2c546cb5dcb42d0d8ebf7fb7a37c3

北京大学
PEKING UNIVERSITY

PART 02 ▸

Solana扩容机制分析

2.1 比特币、以太坊等区块链的弊端

北京大学 PEKING UNIVERSITY

	SOLANA		ethereum	ETHEREUM v2.0	BINANCE SMART CHAIN	CARDANO	Polkadot.	COSMOS	Avalanche	fantom
Current TPS	65,000	7	30	100,000	100	1,000	1,500	1,400	4,500	10,000
Block Time	0.4 seconds	10 minutes	15 seconds	12 seconds	3 seconds	20 seconds	2-3 seconds	1 second	1-5 seconds	1-2 seconds
Transaction Fee	$0.00001	$26.89	$12.76	--	$0.01	$0.21	--	$0.03	$0.03	--
Validators	600	--	--	--	21	--	297	160	932	57
GitHub Stars	1,700	52,700	29,800	1,800	357	3,600	3,200	2,400	662	97
Founding Year	2018	2009	2015	2021	2019	2015	2016	2014	2019	2018
Type	Layer 1	Layer 1	Layer 1	Sharding	Layer 1	Layer 1	Sharding	Layer 1	Layer 1	Layer 1
Working Product	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes
Market Cap ($BN)	$11.32	$989.82	$313.67	--	$89.36	$41.41	$31.33	$4.56	$3.52	$1.50

Data as of 4/29/21

Created by @rareliquid

2.2 以太坊和Solana最本质的区别

时间与状态是否被分割

2.3 Solana系统架构、共识机制、区块链传输流程

公链的效率主要指其处理交易的能力，也就是吞吐量TPS（每秒处理的交易笔数），这个指标受到出块速度和区块容量的影响，同时也影响着交易手续费和用户活跃度。从2018年甚嚣尘上的EOS，到近期发币的Optimism，所有扩容方案几乎都绕不开“加速出块”这个最关键的要素。

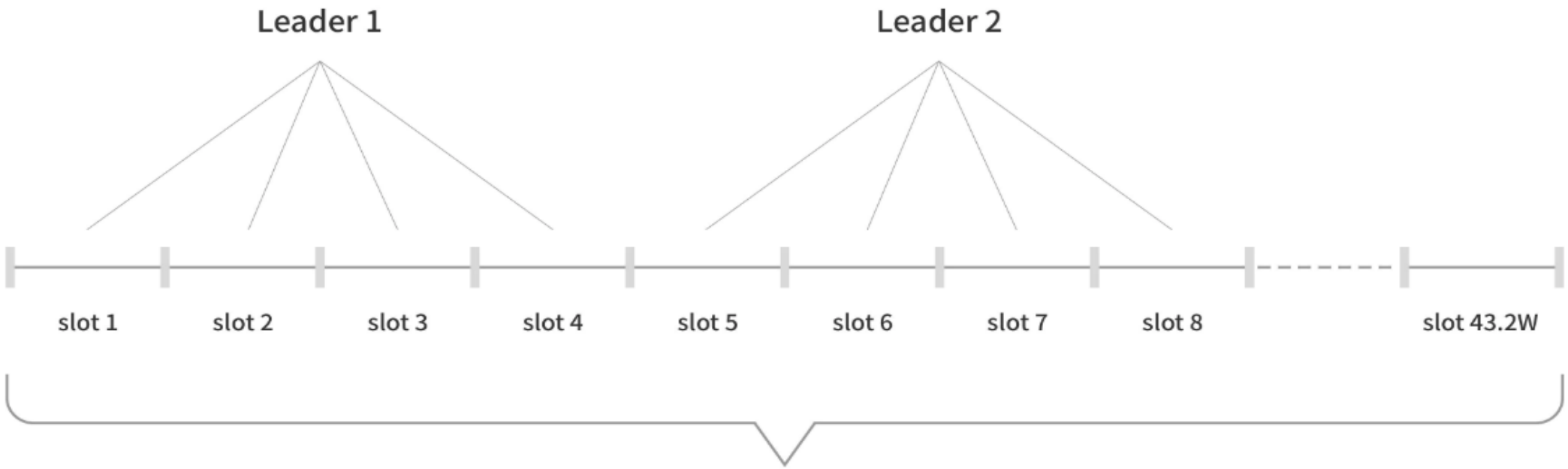
要提升出块速度，往往要在出块流程上“做手脚”，Solana也不出其右。其扩容方式主要立足于**高效利用网络带宽、减少节点间通讯的次数、提高节点处理事务的速度**三大方面，分别对应**出块率、冗余性、并行计算**三个指标。这些措施直接缩短了出块和共识通讯的时间。Solana的创始人Anatoly Yakovenko及其队友对每一个细节都进行了精心雕琢，以系统的可用性（安全）为代价，尽可能在效率上作出提升，基本达到了无分片公链的实际TPS极限，最终作出了“有代价”的创新。

相比于其他采用POS共识算法的公链，Solana最大的创新点在于其**独特的共识协议**和**网络节点通信方式**，该共识协议基于**POS**和**PBFT**（实用拜占庭容错），引入独创的**POH**（Proof of History，历史证明）作为推进区块链账本的机制，独树一帜的创建了自己的共识体系。

单从表现形式的角度看，Solana的共识协议与Cardano最早的Ouroboros（衔尾蛇）算法类似，都包含Epoch（纪元）**和**Slot（间隔）两大时间单位。每个Slot约为0.4~0.8秒，相当于一个区块的时间间隔。而每个Epoch周期包含43.2万个Slot（区块），长达2~4天。

在Solana的系统架构中，最重要的角色分为两类：**Leader（出块者）和 Validator（验证者）**。两者实际上都是质押了SOL代币的全节点，每经过四个slot做一次Leader的轮换，而没有当选Leader的全节点会成为Validator。

2.3 Solana系统架构、共识机制、区块链传输流程

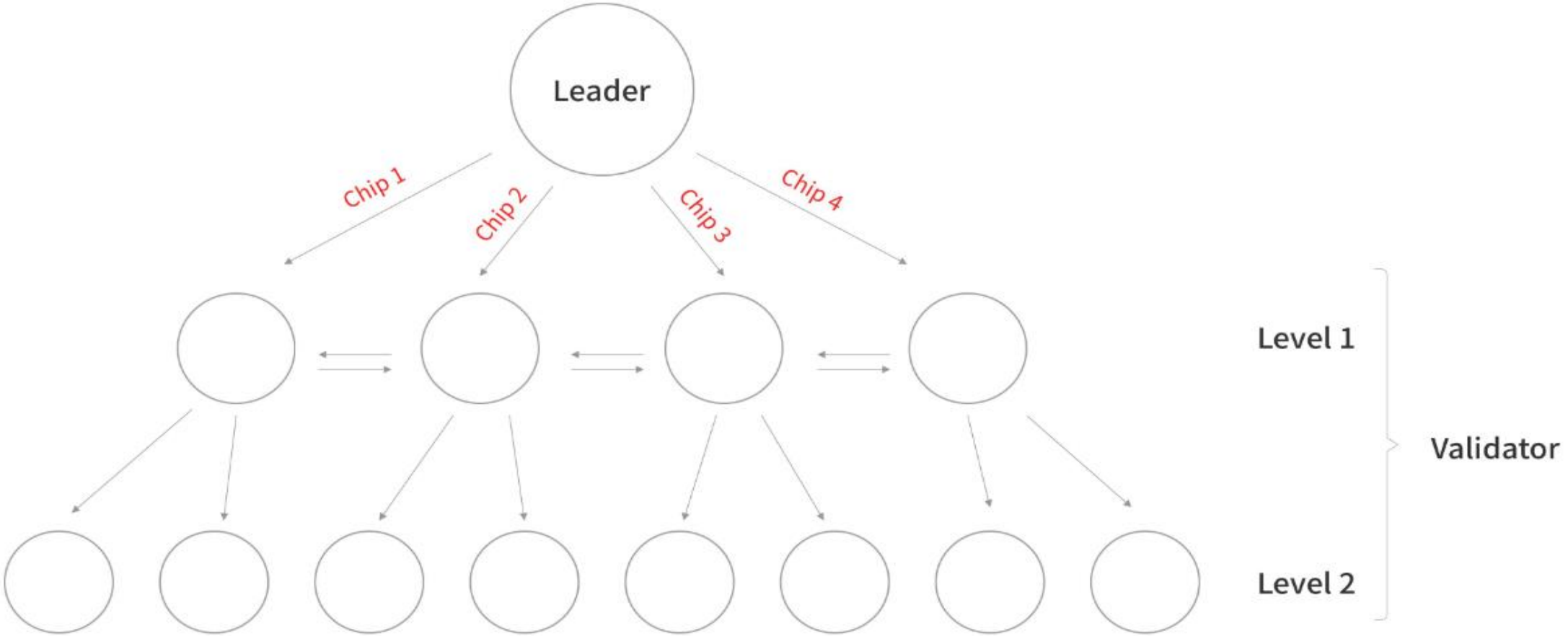


Epoch ≈ 2 — 4 Days

Slot ≈ 0.4 — 0.8 s

1 Epoch = 43.2 W Slots

2.3 Solana系统架构、共识机制、区块链传输流程



2.3 Solana系统架构、共识机制、区块链传输流程

在每个新的Epoch周期开始时，Solana网络会按照各节点的质押权重进行抽选，组成一个**出块者Leader轮换名单，“钦定”了未来不同时刻的出块者**。在整个Epoch（2~4天）内，出块者会按照名单指定的次序进行轮换，每过4个Slot（出块周期），Leader节点就会进行一次变更。在Solana的系统中，Leader的轮换顺序是公开的，这样有利于提高节点向Leader节点转发交易的速率以及降低了在Leader节点崩溃时系统所受到的损失。

2.3 Solana系统架构、共识机制、区块链传输流程

All Validators　Node Versions　**Leader Schedule**

Total **432,000** records

Slot	Leader
#135647980	Ninja1spj6n9t5hVYgF3PdnYz2PLnkt7rvaw3firmjs
#135647981	Ninja1spj6n9t5hVYgF3PdnYz2PLnkt7rvaw3firmjs
#135647982	Ninja1spj6n9t5hVYgF3PdnYz2PLnkt7rvaw3firmjs
#135647983	Ninja1spj6n9t5hVYgF3PdnYz2PLnkt7rvaw3firmjs
#135647984	7LyD6dUSrjCmZYeor4uiTPNZgSTA8zoLFmSKbb6Q2Pps
#135647985	7LyD6dUSrjCmZYeor4uiTPNZgSTA8zoLFmSKbb6Q2Pps
#135647986	7LyD6dUSrjCmZYeor4uiTPNZgSTA8zoLFmSKbb6Q2Pps
#135647987	7LyD6dUSrjCmZYeor4uiTPNZgSTA8zoLFmSKbb6Q2Pps
#135647988	CKUox1FamkkPhzahNjaBytS5FCB7cJggCvcNLDLZCaDY

（Solana第313个Epoch出块节点轮换名单的一部分）

2.3.1 设计选择总结

Solana 的高性能秘诀三个关键指标共同决定了区块链的最大吞吐量：**出块率、并行计算和冗余性**。

1. 冗余度决定了总共需要多少数据和计算量，也就是说，总计算量 = 有效计算 + 冗余度；
2. 并行计算允许节点计算的速度更快；
3. 出块率决定了一定时期内区块链数据库中可保存的数据量。

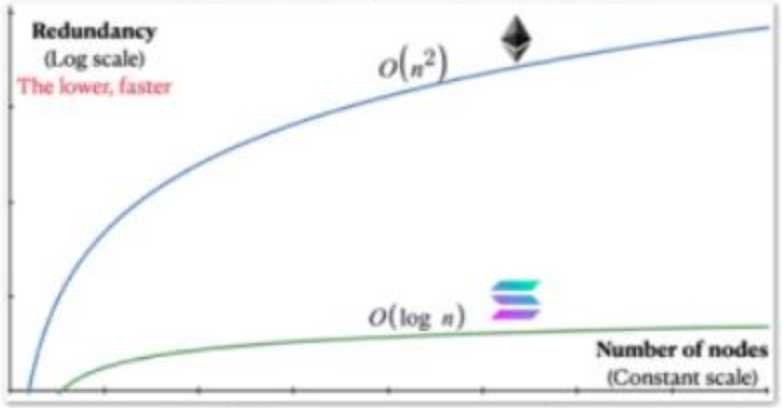
Solana的高性能秘诀
高性能区块链设计的3个要素

Solana's high-performance recipe

3 elements of high-performance blockchain design

冗余度
1. Redundancy

How much overhead to handle?
多少冗余数据量需要处理?



以太坊：O(n^2) 高冗余度
Solana：O(log n) 极低的冗余度

Ethereum: O(n^2) high redundancy
Solana: O(log n) extremely low redundancy

并行计算
2. Parallelism

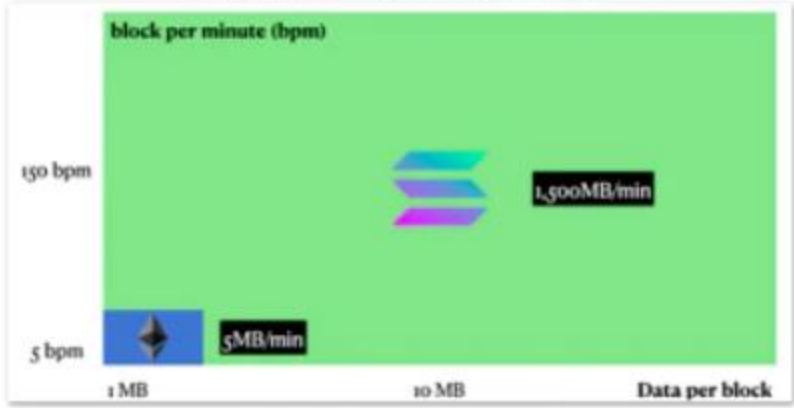
How quickly can nodes compute the data?
节点处理数据的速度有多快?

以太坊：单核CPU
Solana：4096核Nvidia GPU

Ethereum: 1 CPU core
Solana: 4096 Nvidia GPU cores

出块率
3. Block rate

How much data capacity per minute?
每分钟的数据量有多少?



以太坊：5 MB/min
Solana：1500 MB/min

Ethereum: 5 MB/min
Solana: 1500 MB/min

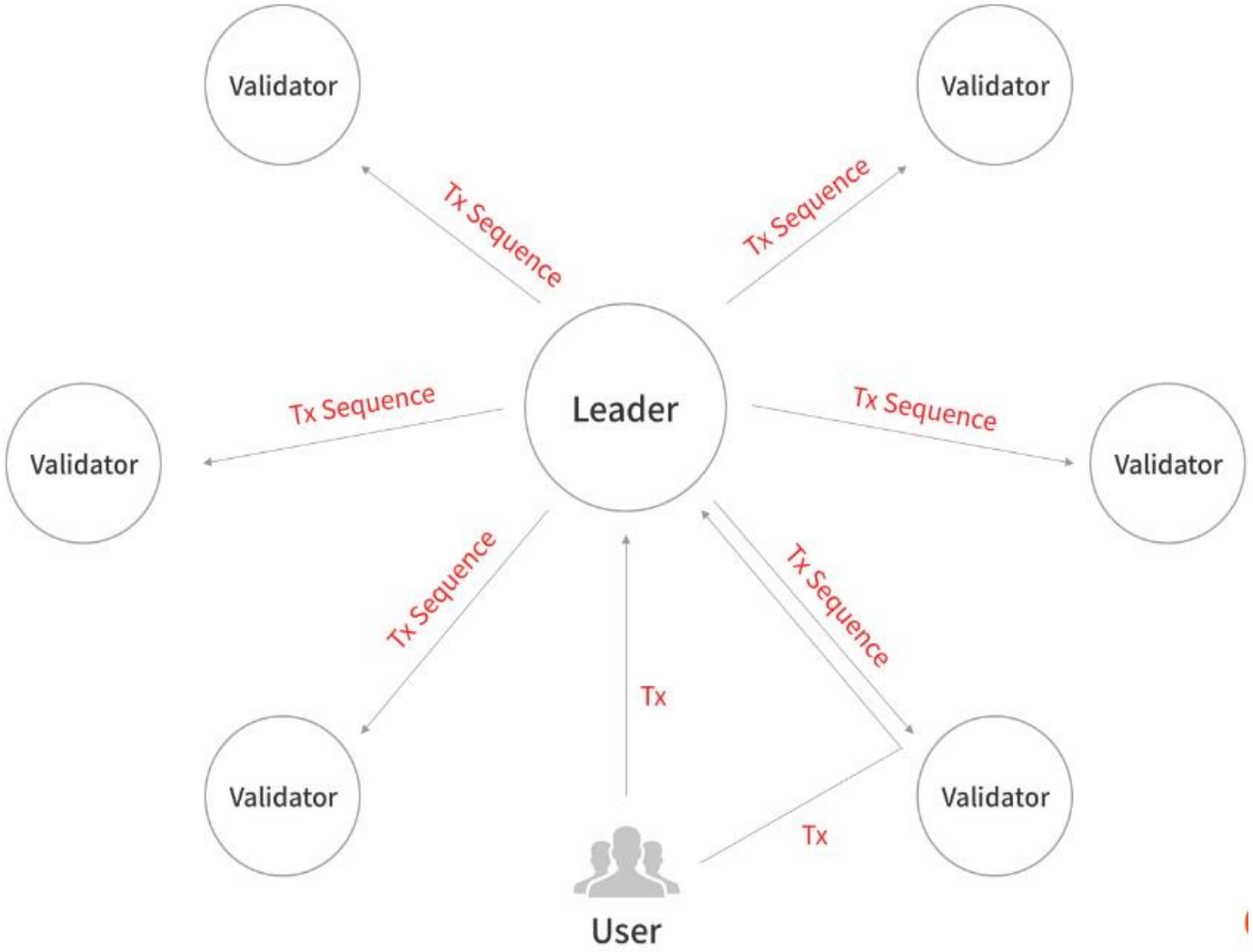
2.3.2 Solana出块流程简述

为了更清晰的理解Solana的扩容机制，我们不妨从出块逻辑开始，对Solana的大致结构进行解析：

1. 用户发起交易后，会被客户端直接转发给Leader节点，或者先被普通节点接收，再立刻转发给Leader；

2. 出块者Leader接收网络内全部的待处理交易，一边执行，一边给交易指令排序，制成交易序列（类似区块）。每隔一段时间，**Leader会把排好的交易序列发送给Validator验证节点；**

2.3.2 Solana出块流程简述

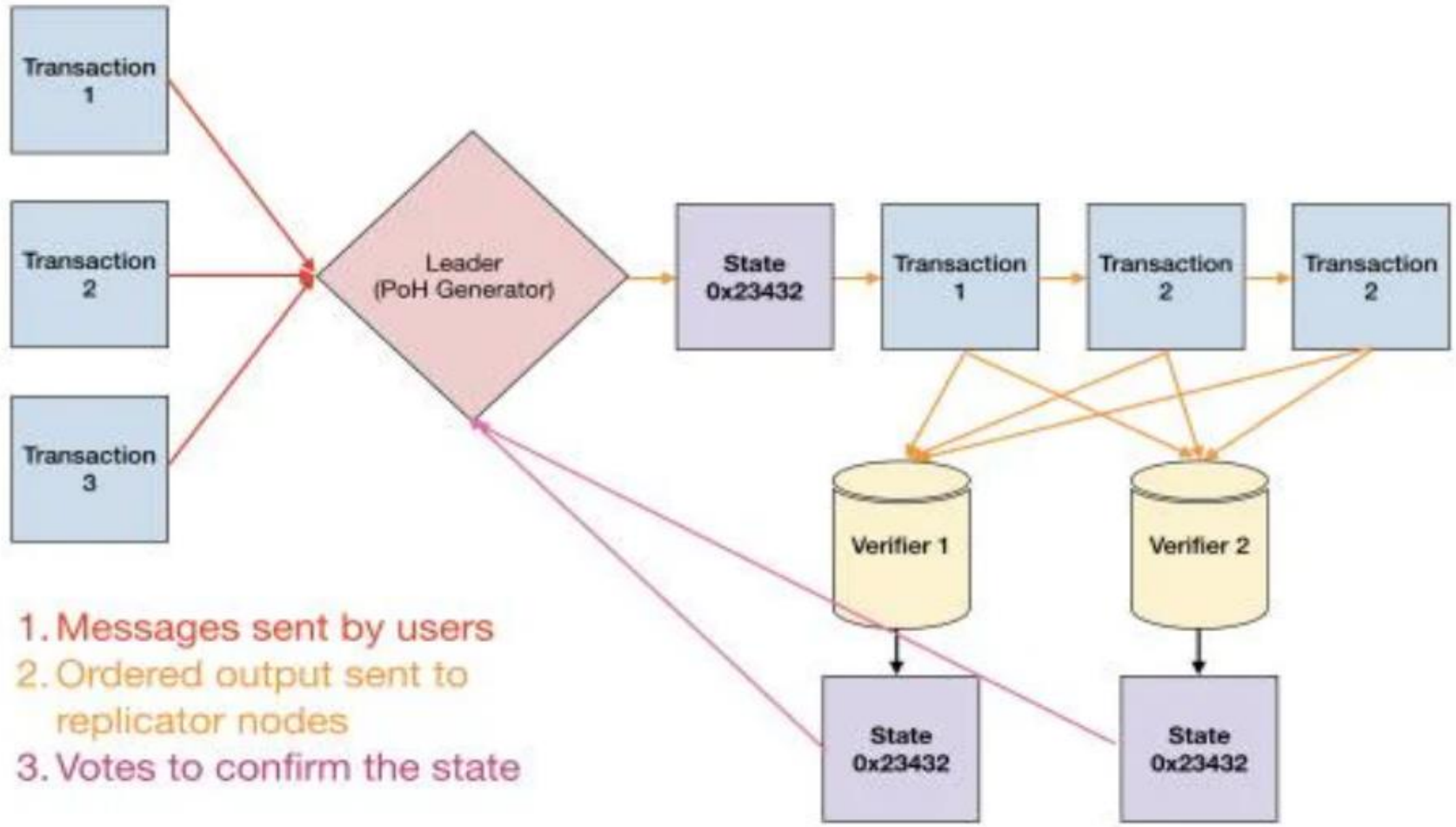


2.3.2 Solana出块流程简述

3. Validator按照交易序列（区块）给定的顺序执行交易，产生相应的状态信息State（执行交易会改变节点的状态，比如改变某些账户的余额）；

4. 每发送N个交易序列，Leader会定期公开本地的状态State，Validator会将其与自己的State作对比，给出 肯定/否定 的投票。这一步就类似于以太坊2.0或其他POS公链里的“检查点”。

2.3.2 Solana出块流程简述



2.3.2 Solana出块流程简述

5. 如果在规定时间内，Leader收集到占全网 **2/3质押权重** 的节点们给出的**肯定票**，则此前发布的交易序列和状态State就被敲定，“检查点”通过，相当于区块完成**最终确认Finality**；

6. 一般而言，给出肯定票的Validator节点与出块者Leader所执行的交易、执行后的状态都是相同的，数据会同步。

7. 每过4个Slot周期，Leader会进行一次切换，这意味着Leader每次大概有1.6秒~3.2秒时间掌握网络的“最高话语权”。

2.3.3 Solana扩容机制细解

表面看来，Solana的出块逻辑与其他采用POS机制的公链大体一致，都有一个发布区块、对区块投票的过程。但如果我们对每一个步骤都展开观察，不难发现**Solana与其他公链之间有着天壤之别，而这正是其高TPS、低可用性的根源所在：**

2.3.3 Solana扩容机制细解

一、通过确定的领导节点轮换减少冗余度

二、Gulf Stream与网络宕机

三、类似BT种子的Turbine传输协议

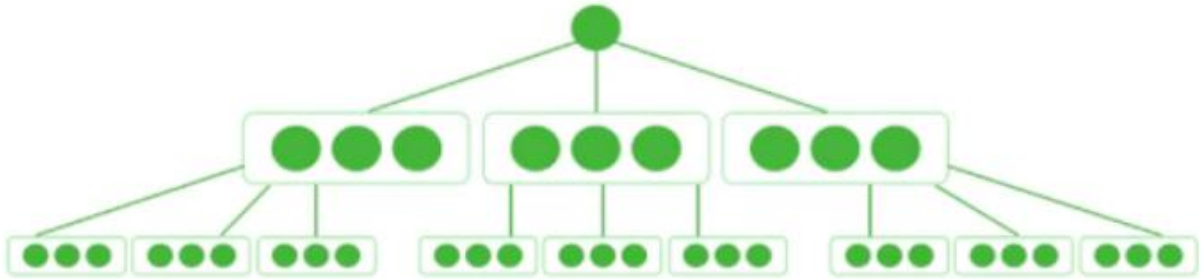
四、PoH（历史证明）

五、全网一致的时间推进

六、针对节点本身的纵向扩容

2.3.3.1通过确定的领导节点轮换减少冗余度

Ethereum:
Every node checks every other node
Redundancy is quadric to n
O(n^2)

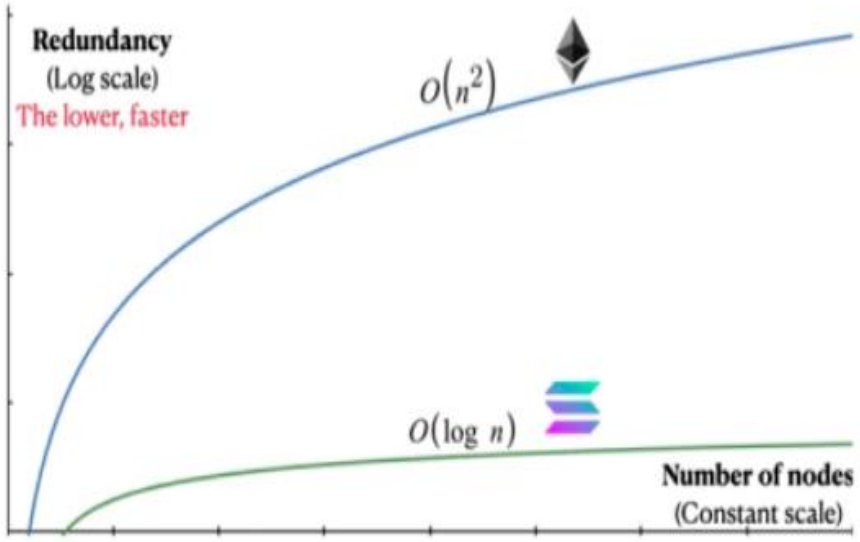


Solana:
A leader-based division-of-work protocol
Redundancy is logarithmic to n
O(log n)

2.3.3.1通过确定的领导节点轮换减少冗余度

TPS (Illustration Only)	100 Nodes	1000 Nodes	10,000 Nodes	100,000 Nodes
O(n^2)	100k	1k	10	0.1
O(log n)	100k	30k	10k	3k

Simple illustration: how redundancy Big-O affects TPS



A more scientific graphing of redundancy classes

2.3.3.1通过确定的领导节点轮换减少冗余度

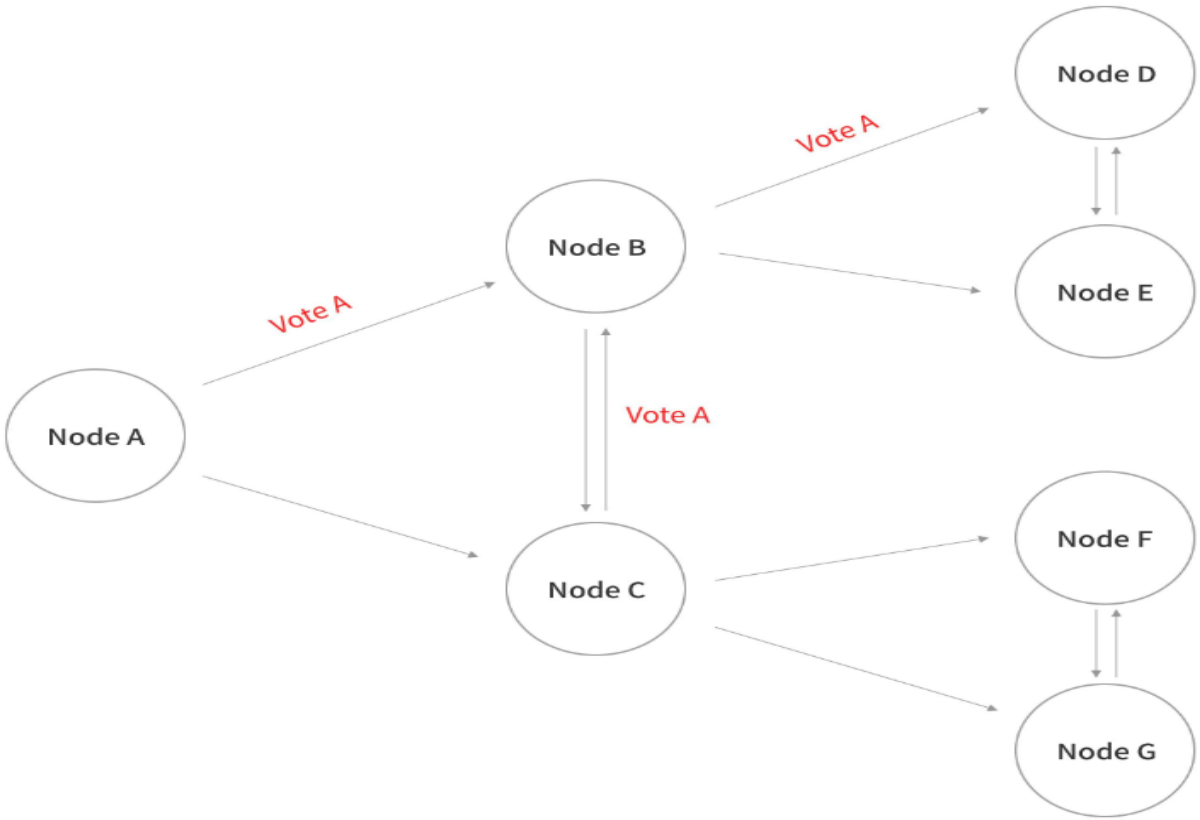
在其他POS公链中，由于缺乏单一的、可信的出块节点，网络的共识通讯效率极低，产生的时间复杂度往往比Solana高出几个数量级，这成为了多数公链在TPS上的瓶颈。

以主流的POS共识协议或PBFT算法为例，这些算法大多采用了和Solana相同的时间单位与角色划分，也有类似于**Epoch纪元、Slot区块周期、Leader出块者、Validator验证者、Vote投票** 的设定，只是参数设置和叫法不同而已。最大的不同在于，此类算法大多以安全性（可用性）为前提，不会提前公开Leader名单。

由于没有公开的出块者，节点间会“互不信任”并“**各自为政**”。此时，若某个节点自称为合法出块者，大家并不敢信任他，必须要其出示相关的Proof证明才行。但此类Proof证明的生成、传播、验证会浪费带宽资源，并产生额外的工作量（甚至会和ZK零知识证明扯上关系）。**Solana公开每个时段的Leader，可以避免此类麻烦。**

更为重要的是，在绝大多数POS共识协议或PBFT类算法中，针对新区块的投票Vote（一个区块要得到网络内2/3节点的肯定票才能敲定），往往由各个节点通过“流言协议”，以类似1对1交流的方式发送或收集，有点类似于病毒式随机扩散，实质等价于 每两个节点间都要通讯一次，其复杂度和耗时远高于Solana的共识协议。

2.3.3.1通过确定的领导节点轮换减少冗余度

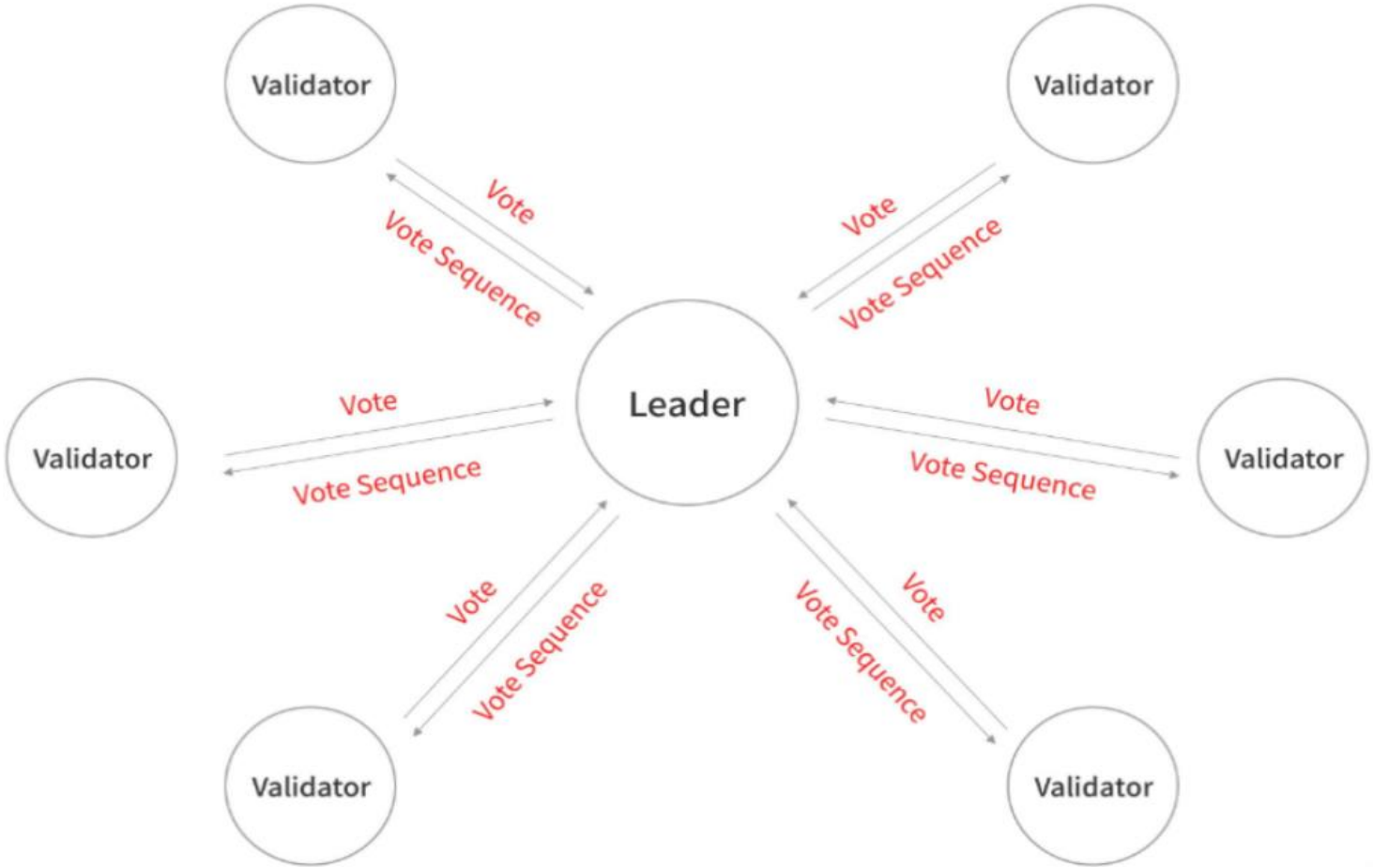


CATCHERVC

（普通公链里单个节点投票的传播方式，类似过程每个节点都会做1次，每次出块要进行N次类似的传播）

2.3.3.1通过确定的领导节点轮换减少冗余度

对此，Solana以不同的方式改良了**节点收集投票的通讯过程**，降低了时间复杂度。通俗的讲，Leader集中汇总所有Validator发出的投票，**再把这些投票打包在一起（写进交易序列里），一次性推送到网络中**。



2.3.3.1通过确定的领导节点轮换减少冗余度

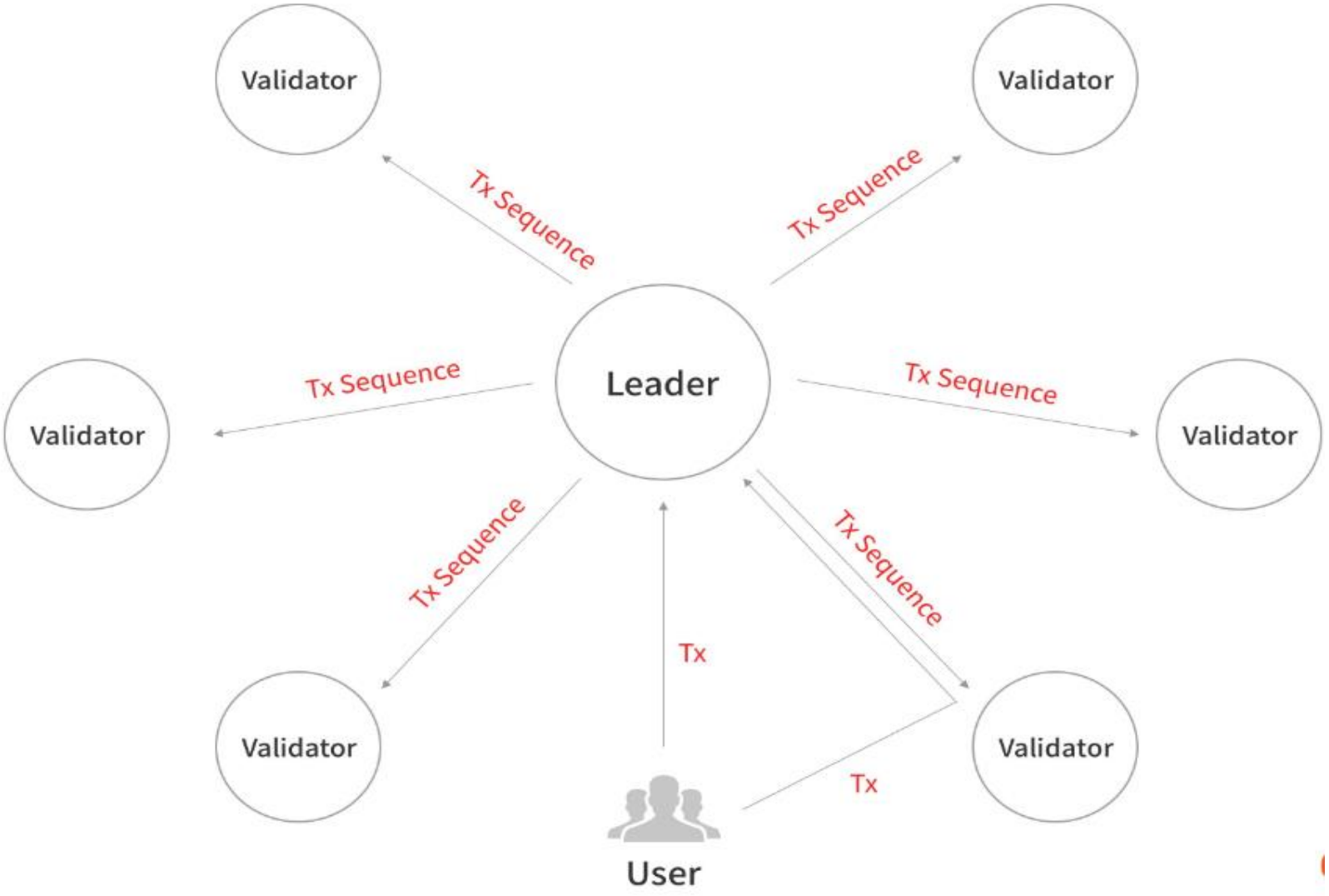
格外值得注意的是，由于每个Epoch内（2~4天）**的出块者名单是提前公开的，Solana的共识协议与原始的Tendermint算法并无本质区别，实际都**没有**赋予出块者以**不可预测性，所有人都能预知未来某个时间点由谁来出块，这就会在 **可用性/安全性 上产生诸多隐患**。 （Leader易遭遇有预谋的DDOS攻击，提高了故障率，若连续几个Leader出现故障，则网络容易宕机；且用户可提前贿赂Leader等）

2.3.3.2 Gulf Stream与网络宕机

Solana公开出块者Leader名单还有一个更重要的目的：配合其独创的Gulf Stream（海湾流）机制，提高网络处理交易的速度。

用户发起交易后，往往被客户端程序直接转发给指定的Leader，或者先被某个普通节点接收，再被该节点快速发送给Leader。这种方式可以让Leader尽快接收交易请求，提高响应速度。（称为Gulf Stream机制，是Solana宕机的主因之一）

2.3.3.2 Gulf Stream与网络宕机



2.3.3.2 Gulf Stream与网络宕机

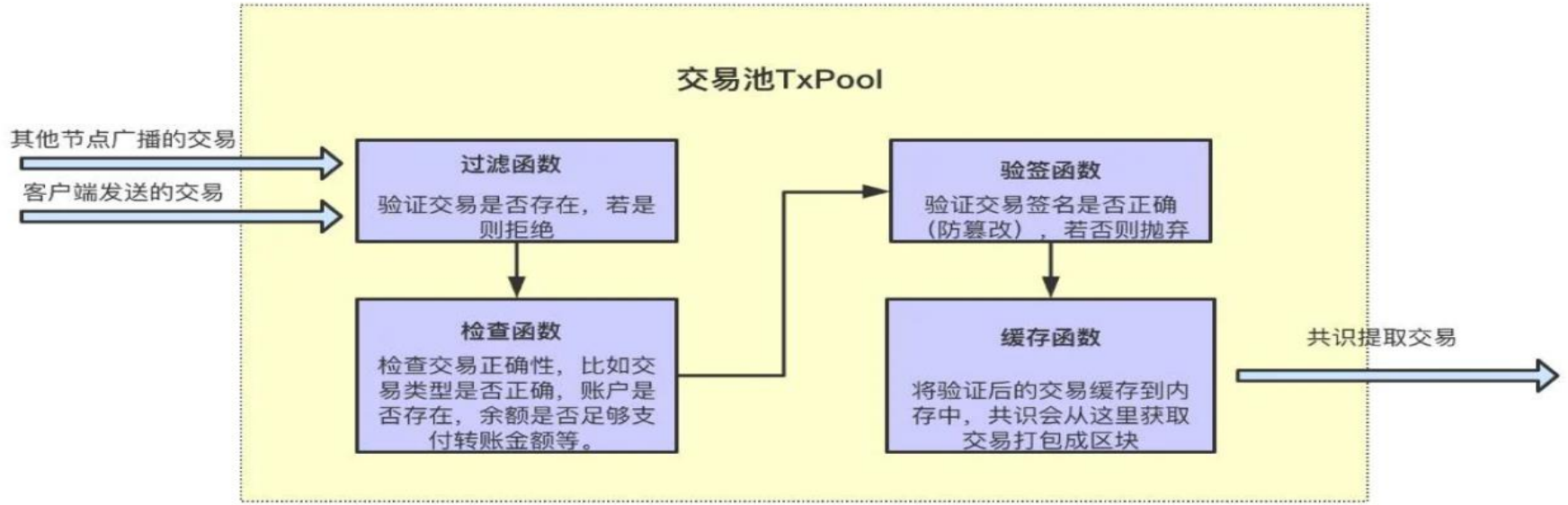
Solana的这种设定，是与其他公链截然不同的交易提交方式。Gulf Stream取缔了比特币和以太坊的“全局交易池”设定，普通节点不运行大容量的交易池。一个节点收到用户的待处理交易后，只需交给Leader，不必再发给其他节点，这种做法大幅提高了效率，但由于取缔了交易池，普通节点无法高效拦截垃圾交易，容易导致Leader节点宕机。

2.3.3.2 Gulf Stream与网络宕机

为了深刻理解这一点，我们可以对比ETH：

1. **以太坊的每个节点都有名为交易池（内存池）的存储区域，用于存放未上链、待处理的交易指令。**

2. 当**节点接收到新的交易请求后，会先进行过滤**，判断交易指令是否合规（是否为重复/垃圾交易），之后将其存入交易池，再转发给其他节点（病毒式扩散）。



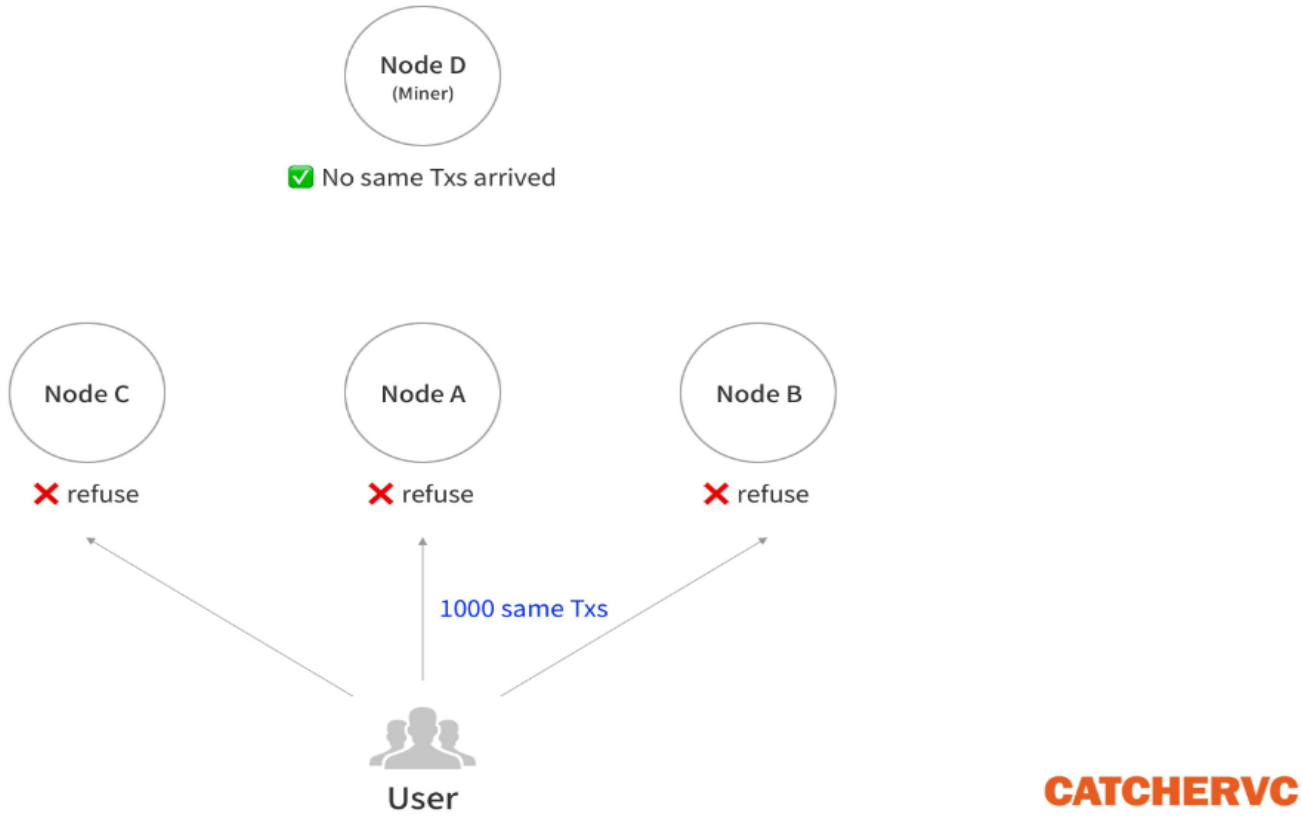
3. 最终，一笔合法的待处理交易会传遍网络，放入所有节点的交易池里，这就让**不同节点都获取到相同的数据，表现出“一致性”。**

2.3.3.2 Gulf Stream与网络宕机

Solana取缔了以太坊的那种交易池，待处理的交易无需在网络内随机扩散，而被快速提交给指定的Leader，再打包成交易序列一次性分发出去（类似于前文的分发投票的方式）。**最终，一笔交易只需要夹在交易序列里，在网络内传播1圈（以太坊实际是2圈）。在交易数量很大的情况下，这个细微差别可以大幅提高传播效率。**

但根据交易池TxPool的相关技术说明，交易池/内存池实质发挥了数据缓冲区和过滤器的作用，能提升公链可用性。所有节点都运行交易池，收录网络内全部待处理交易，不同节点就可以独立过滤垃圾请求，第一时间拦截重复交易，分担流量压力。虽然采用交易池会放慢出块速度，但如果有用户发起重复交易（交易池里有记录的请求），或其他类型的垃圾请求，接收到交易的节点可在本地将其过滤，不会再转发出去，这就让过滤工作被全网节点分担开。

2.3.3.2 Gulf Stream与网络宕机



（在以太坊网络，恶意用户发起的重复交易更容易被各个节点直接拦截）

2.3.3.2 Gulf Stream与网络宕机

Solana采取了背道而驰的做法。在Gulf Stream机制下，普通节点们不运营全网一致的交易池，无法高效拦截 重复/垃圾交易。普通节点真正能做的，只是检查交易数据包是否符合正确格式，无力辨别恶意重复请求。同时，由于普通节点“一股脑”的把交易指令推给Leader，相当于把过滤交易的“重担”甩给了Leader自己，在流量极大、重复交易数量极多的情况下，Leader节点会因压力过大而无法顺利出块，共识投票将无法顺利传播，网络容易崩溃。

对此，Solana创始人Anatoly Yakovenko于今年1月27日表示，**某些热门项目的公售时段，每秒最多有近200万笔交易请求到达同一个Leader节点，其中90%以上是完全相同的重复交易，最终导致Solana宕机。**

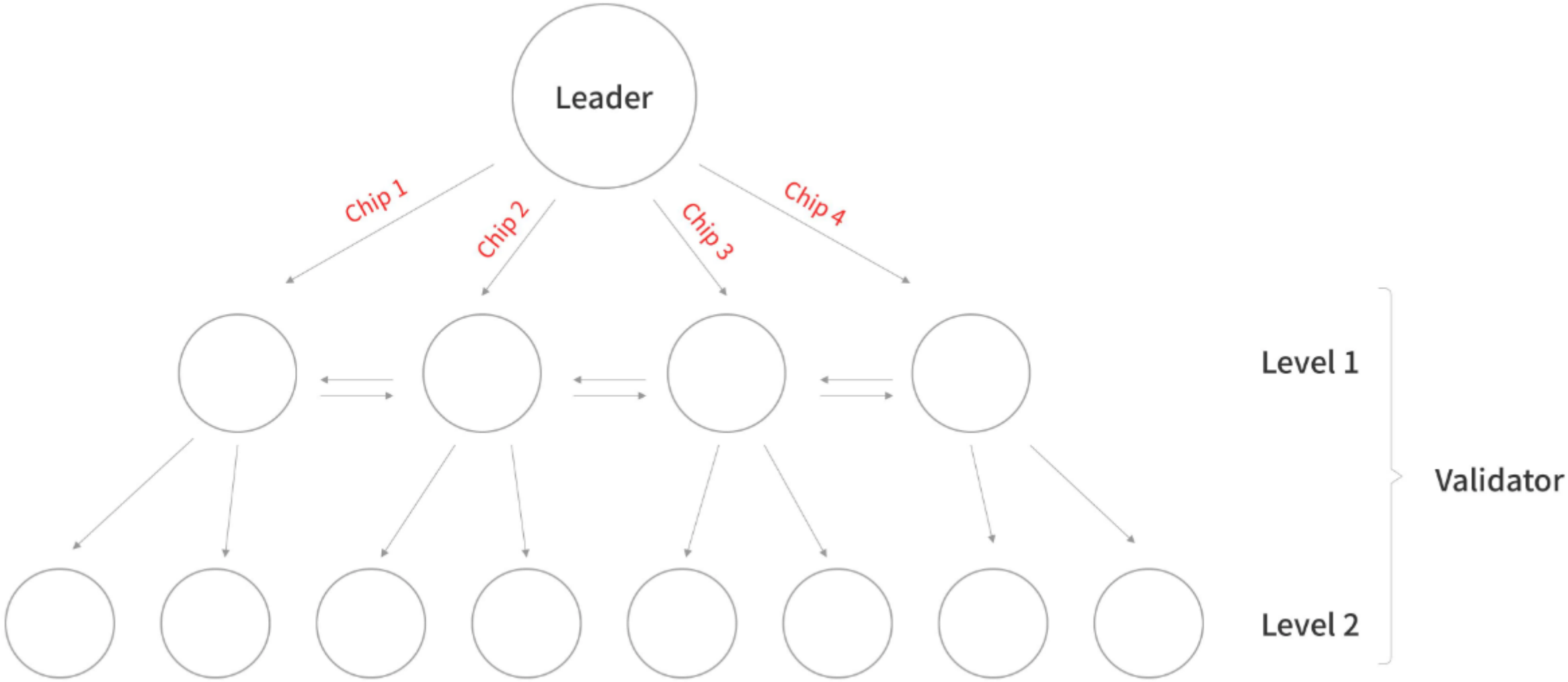
综上所述，以太坊实质是牺牲效率换安全，Solana则是牺牲安全换效率，它所面临的问题可归纳为： 由于Leader轮换顺序是给定的，必须按照这个轮换链条不断的走下去。但由于流量分担机制不完善，Leader节点故障几率较高，如果某段时间内用户流量过大（如某些火热NFT开启公售），可能使多个Leader先后出现故障（比如未来40个Slot的Leader都不能顺利出块），这样一来，**共识过程受阻，网络会分叉、Leader轮换链条会彻底断裂，最终会彻底崩溃。**

2.3.3.3 类似BT种子的Turbine传输协议

在上文的Gulf Stream机制配合下，Leader迅速接收一段时间内的全部交易请求，检查其合法性，之后会执行交易。同时，Leader采用称为POH（Proof of History）的机制，为每笔交易都盖上一个序列号，为交易事件排序。

Leader把交易事件排好序后，会把交易序列切成X个不同的碎片，分别发送给质押资产最多的X个Validator，再由他们传播给其他Validator。Validator群体会自行交换收到的碎片，在本地拼凑完整的交易序列（区块）。

2.3.3.3 类似BT种子的Turbine传输协议



2.3.3.3 类似BT种子的Turbine传输协议

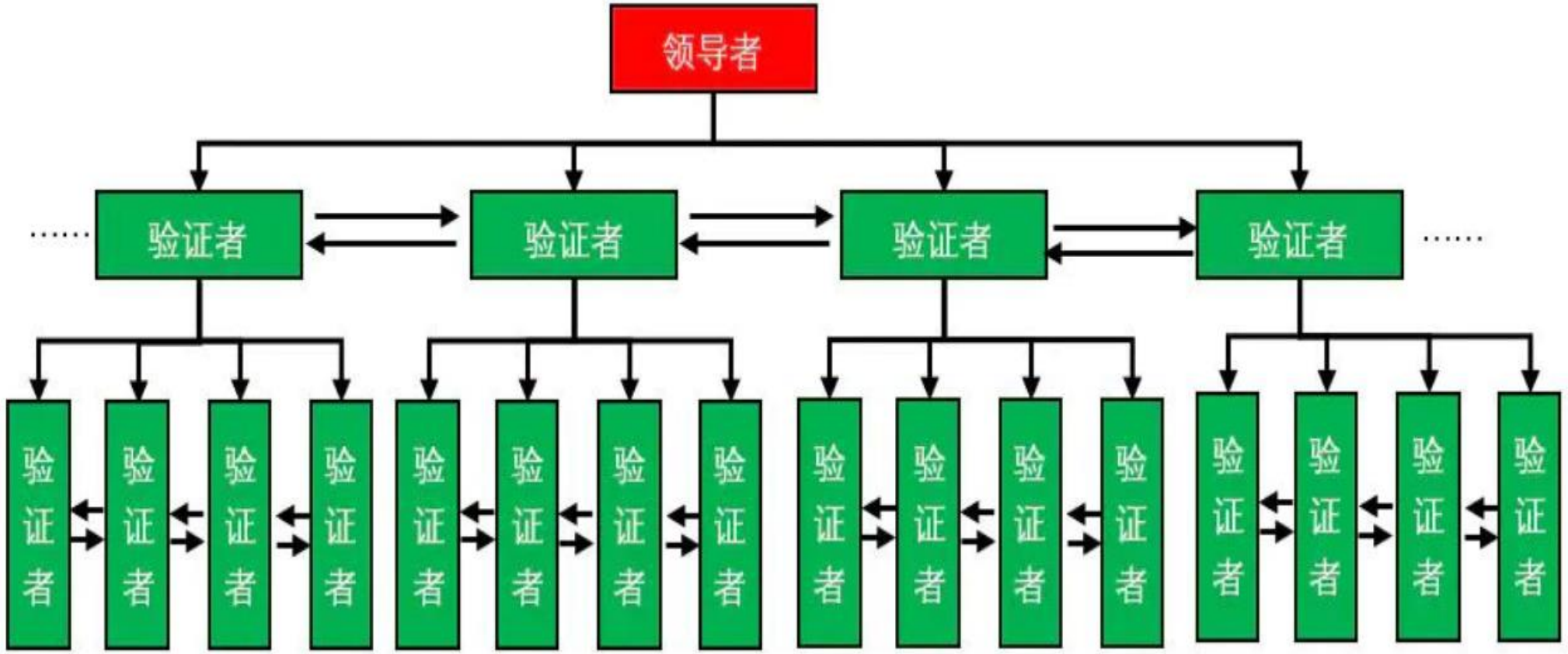
为了便于理解，我们可以将每个不同的碎片视作数据量缩减的小区块。Leader一次对外分发X个碎片，相当于发出X个不同的小区块，让不同的节点接收并进一步扩散。

Solana的这种消息分发方式很特别，**灵感来自于BT种子。**（同时利用多个节点的闲置带宽，并行式的传输数据）。一般而言，交易序列被切分的碎片数越多，节点群体扩散碎片、拼凑交易序列的速度越快，**数据同步效率也会显著提升。**

而在其他公链中，出块者会向X个邻居节点发送相同的区块，相当于把一个区块复制X份发出去，而非分发X个不同的碎片（小区块），这种做法产生的数据冗余和带宽浪费很严重。究其根源，传统的区块Block式结构不可切分，根本无法灵活传输，而**Solana干脆以交易序列Sequence替代区块式结构，结合类似BT种子的Turbine协议，可以实现高速的数据分发，极大提升了吞吐量TPS。**

同时，在Turbine协议下，节点按照其质押资产的权重，被划分为不同的层级（优先级），**质押资产多的Validator率先收到Leader发出的数据，之后由这些节点传递给下一层。**在这种机制下，占全网质押资产2/3权重的节点群体，会最先收录Leader发出的交易序列，加快账本（区块）的确认速度。

2.3.3.3 类似BT种子的Turbine传输协议



2.3.3.4 PoH（历史证明）

前文提到，Turbine协议允许Leader将交易序列切碎，把不同的碎片发布出去。这种做法要有一个保障：交易序列被切碎后，要容易被拼凑复原。为了解决这个问题，Solana特意在数据包中掺杂纠删码（可防止数据丢失），并且引入独创的POH（Proof Of History）机制为交易事件排序。

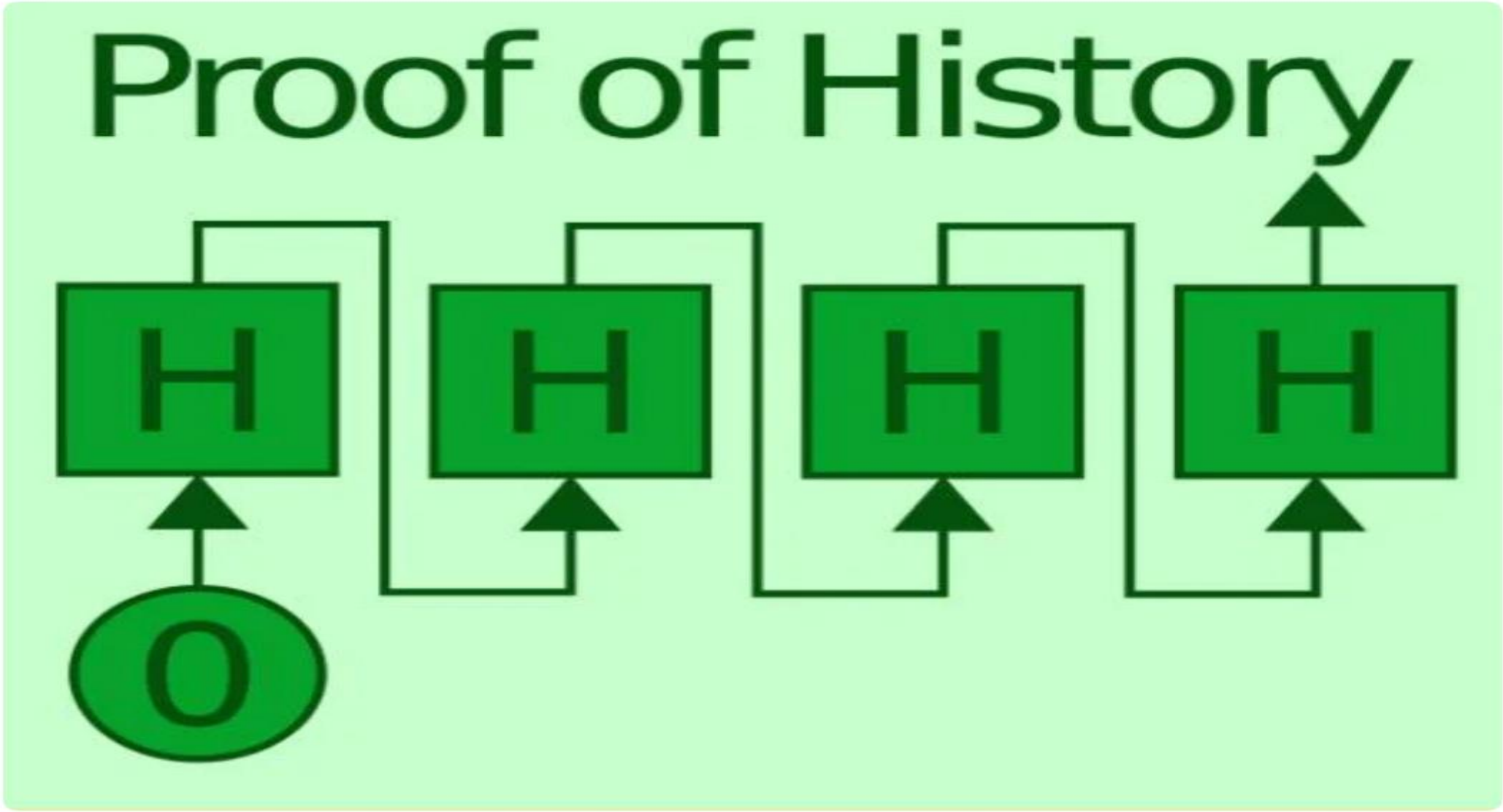
在Solana白皮书中，Yakovenko以哈希函数SHA256为案例，展示了POH的原理。为了便于理解，本文将以下面的例子解释POH机制：

1. SHA256函数的输入值和输出值是唯一映射的（1对1），输入参数X后，仅有唯一的输出结果SHA256（X）=？；不同的X会得到不同的 ？=SHA256（X）；

2. 如果循环、递归的计算SHA256，比如： 定义X2 = SHA256 （ X1 ）， 再用X2计算X3 = SHA256 (X2)，再算X4 = SHA256 (X3)，如此重复迭代下去，Xn = SHA256 （ X[n-1] ）；

3. 反复执行这个过程，最终我们会得到一个X1，X2，X3.....Xn的序列，该序列有个特点：Xn = SHA256 （ X[n-1] ），排在后面的Xn是前面X[n-1]的“后代”。

2.3.3.4 PoH（历史证明）

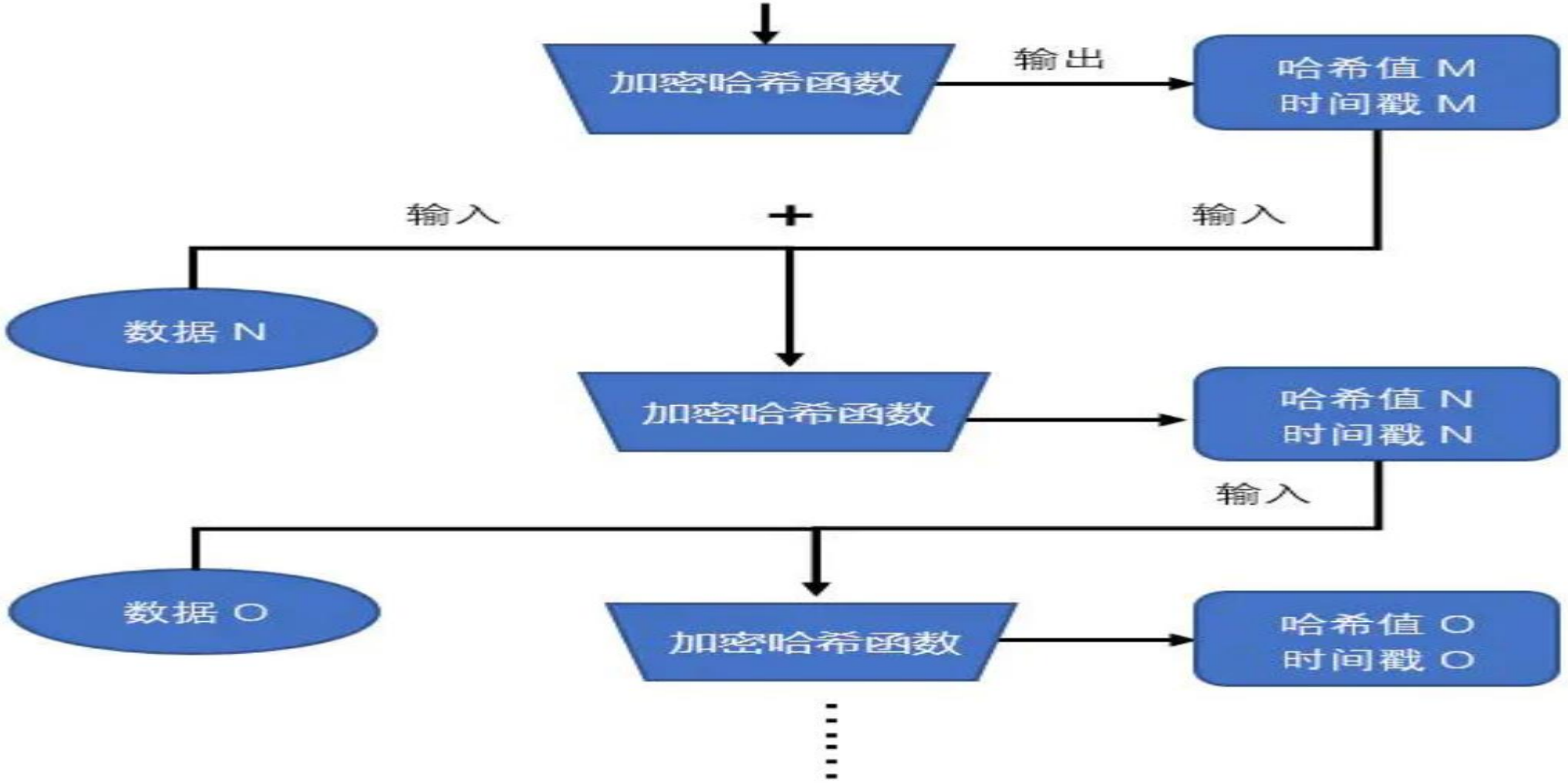


2.3.3.4 PoH（历史证明）

4. 将该序列公开发布后，如果有外人想验证序列的正确性，比如他想判断Xn是否真的是X[n-1]的“合法后代”，可以直接将X[n-1]代入SHA256函数去算，看结果和Xn是否相同。
5. 在这种模式下，没有X1就得不到X2，没有X2得不到X3.....没有Xn得不到后面的X[n+1]。这样一来，序列就具有了连续性和唯一性。
6. 最关键的一点：交易事件可以被插进序列里。比如： 在x3出生后、x4未出现时，交易事件T1可作为外部输入，和X3迭加在一起，得到x4 = SHA256（X3+T1）。其中，X3的出现略早于T1，X4则为（T1+X3）孕育的后代，**T1实质被夹在X3和X4的“生日”之间。**

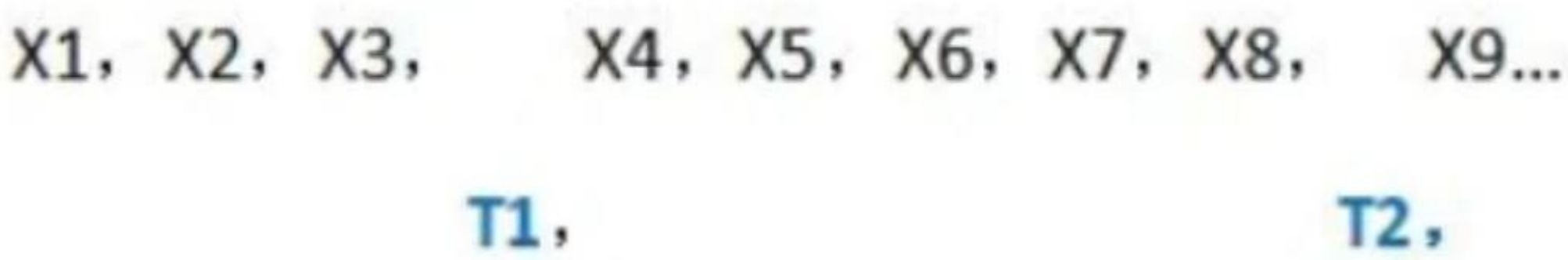
以此类推，T2可以在X8产生后，作为外界的输入参数，计算X9=SHA256（T2+X8），这样T2的出现时间就被夹在x8和x9的“生日”之间；

2.3.3.4 PoH（历史证明）



2.3.3.4 PoH（历史证明）

在上面的场景中，**实际的POH序列为以下形式：**

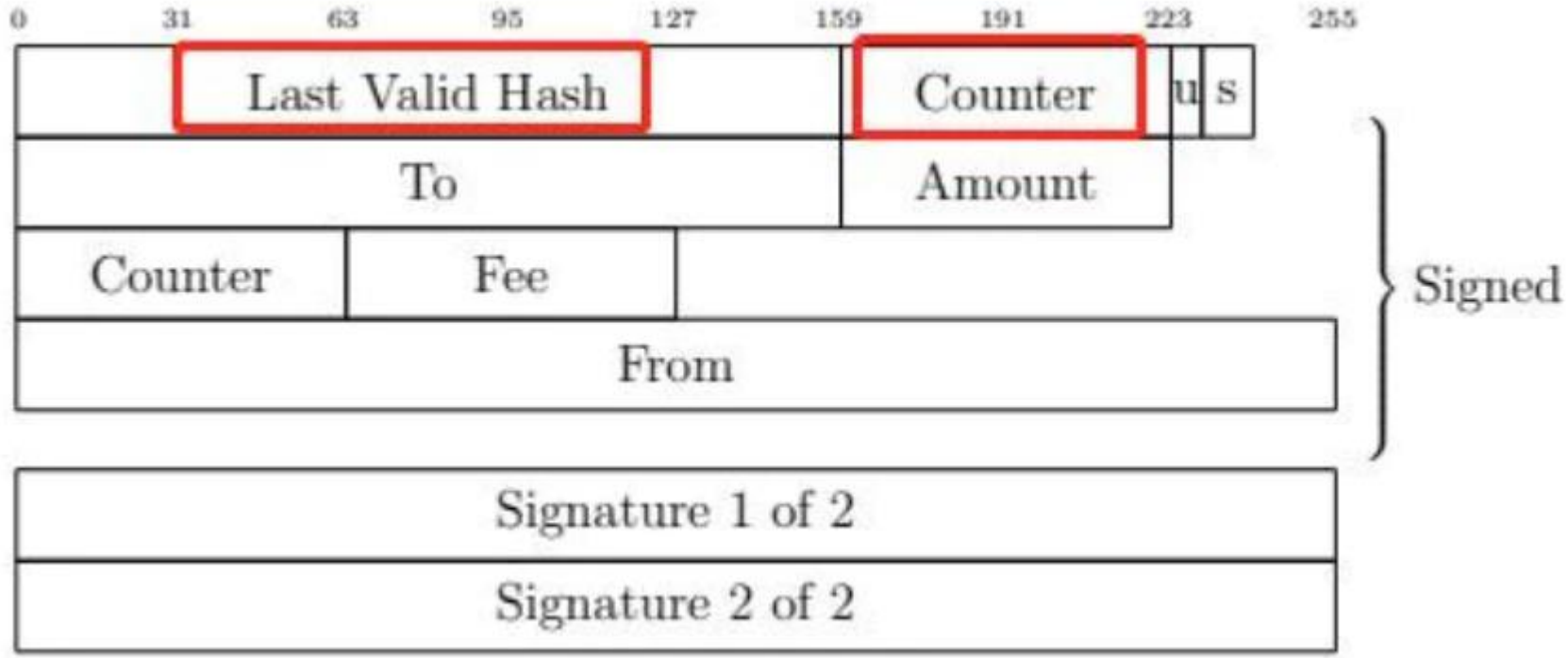


其中，交易事件T1和T2是外界插入序列里的数据，在POH序列的时间记录上，他晚于X3早于X4。

只要给出T1在POH序列里的次序号，可得知它之前发生了多少次SHA256计算（T1前面有X1、X2、X3，发生了3次SHA256计算）。 同样的道理，T2前面有X1~X8八个X，发生了8次计算。

2.3.3.4 PoH（历史证明）

7. Leader在对外发布交易序列时，只要在T1的数据包里给出X3的数值，并告知X3的序号（第3个），接收数据包的Validator便可解析出T1之前的完整POH序列； 只要在T2的数据包里，提供X8的数值，及其序号8，Validator便可解析出T2之前的完整POH序列；



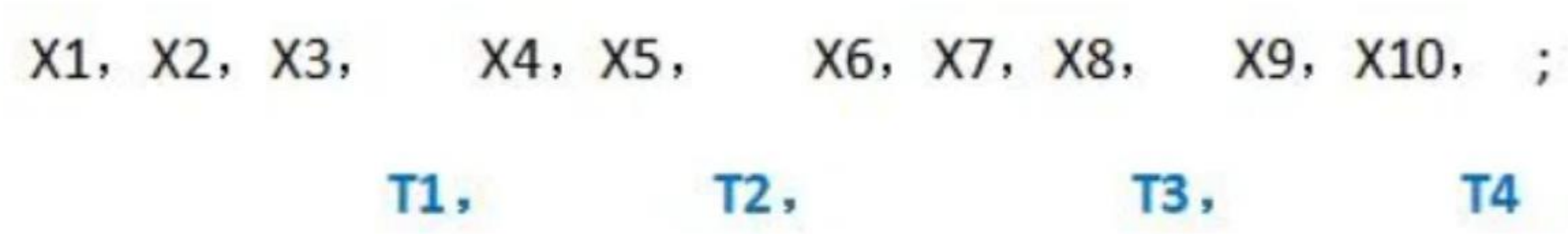
2.3.3.4 PoH（历史证明）

8.按照POH的设定，只要标记出每笔交易在POH序列里的序号（Counter），并给出紧挨着它的X值（Last Valid Hash），就可以披露出每笔交易的次序。由于SHA256函数本身的特性，这种通过哈希计算来敲定的次序，难以被篡改。

同时，Validator知道Leader得出POH序列的方式，他们可以执行相同的操作，还原出完整的POH序列，验证Leader发布的数据的正确性。

比如：如果Leader发布的交易序列数据包为： T1，序号3，紧邻X3； T2，序号5，紧邻X5； T3，序号8，紧邻X8； T4，序号10，紧邻X10； POH序列初始值X1；

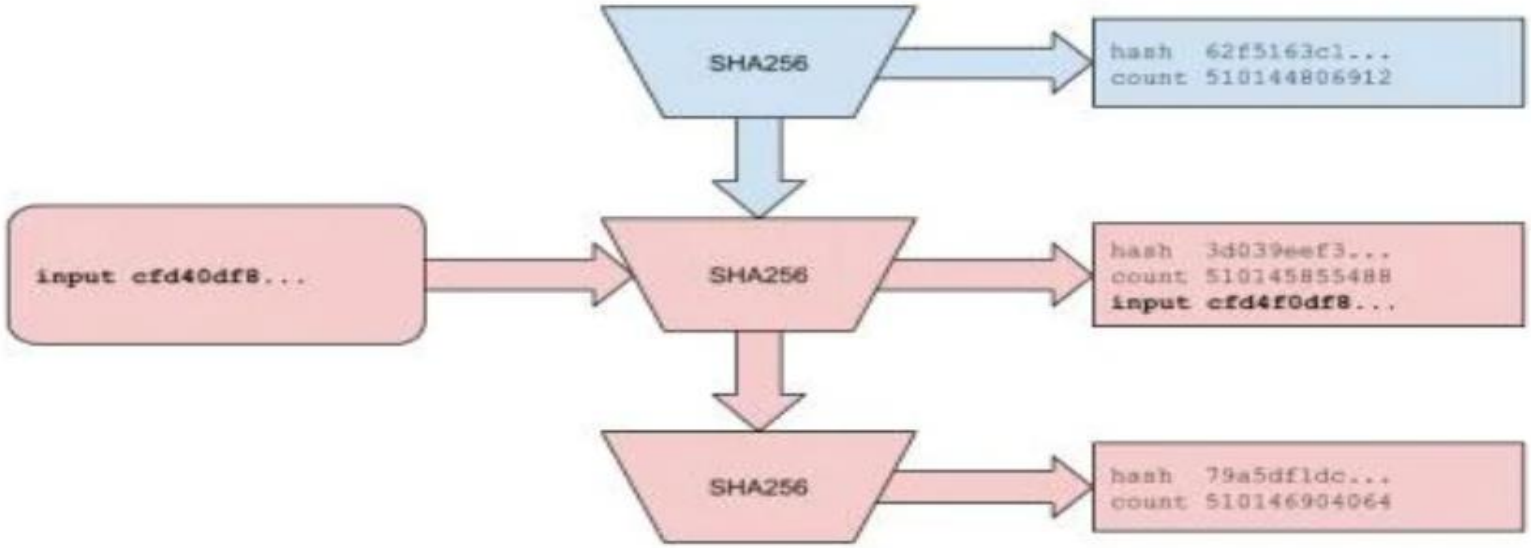
Validator接收到以上数据包后，便可把X1作为初始参数，循环代入SHA256函数自行计算，解析出完整的POH序列为：



2.3.3.4 PoH（历史证明）

这样一来，只要知道序列里总共包含多少个X，就可得知计算者做了几次SHA256计算。事先**估计好每次哈希计算的耗时，就可以知道不同交易的时间间隔**（比如T1和T2之间隔了2个x，就隔着2次SHA256计算，约为？？毫秒）。得知了不同交易之间相隔的秒数后，可以更方便的确定每笔交易发生的时间点，省去了很多麻烦的工作。

9. 一般而言，Leader会时刻不停的执行SHA256函数，得到新的X，把序列不断向前推进。如果有交易事件，就将其作为外部输入，插进序列里；



2.3.3.4 PoH（历史证明）

10. 如果有节点尝试在网络中发布掺水的序列，替换Leader发布的版本，比如把上文中的X2替换为X2’ ，序列变为X1，X2’，X3.....Xn，显然其他人只要对比X3和SHA256（X2’），就可以发现两者对不上号，序列造假了。 所以，造假者必须把X2’之后的X全部替换才行，但这样做成本很高，尤其在X的个数很庞大的情况下，造假将非常浪费时间。此情此景，最好的办法就是不去造假，收到了Leader发出的序列后原方不动的转发给别的节点。 再考虑到Leader会在发布的每个数据包里加上自己的数字签名，在网络内传播的序列其实是“唯一的”“难以被篡改的”；

2.3.3.5 全网一致的时间推进

Solana的创始人Yakovenko曾强调，POH最大的作用是提供了一个“全局一致的单一时钟”（其实应该转译为：全网一致的时间推进），这句话其实可以这样理解： **Leader节点在网络内发布了一个唯一的、难以篡改的交易序列。根据该交易序列的数据包，节点可以解析出完整的POH序列，而POH序列是Solana独创的计时方式，可以作为时间参照物。**

如前文所述，由于Leader会时刻不停的执行SHA256哈希计算，把POH序列不断向前推进，这个序列记录了N次哈希计算的结果，对应着N次计算过程，包含了时间推移。而Solana把计算的次数当做独特的计时方式。

在原始的参数设定中，默认200万次哈希计算对应现实中1秒，每个Slot出块周期为400 ms，也就是说每个Slot产生的POH序列包含80万次哈希计算。

以上只是理想状态下的设定，在实际的运行过程中，每秒可产生的哈希计算次数往往不固定，所以实际的参数应该是动态调整的。但以上说明可以解释POH机制的大致逻辑，**这种设计让Solana节点在收到POH序列后，根据其中包含的哈希计算次数，判断1个Slot是否结束，以及是否到了下一个Leader该出现的时候（4个Slot一轮换）。**

2.3.3.5 全网一致的时间推进

由于可以判断每个Slot的 **起始点**，Validator会把 **起始点** 中间夹着的交易序列划分为一个区块。一旦得到敲定，就相当于把账本向前推进了一个区块，系统则向前推移了一个Slot。

换言之，只要节点都收到相同的交易序列，那么他们解析出的POH序列，及对应的时间推进都是一致的。这就创造了一个“全局一致的单一时钟”（全网一致的时间推进）。

此外，由于交易事件在POH序列中的序号是给定的，节点仅凭自己计算就可获知，不同的交易之间隔了多少次哈希计算（隔着几个X），也就能大致估算出不同交易之间的时间差△T。

有了大致的△T和某个初始的时间戳TimeStamp 0后，就可以像多米诺骨牌一样，粗略估算每起事件的发生时刻（时间戳）。

2.3.3.5 全网一致的时间推进

比如： T1发生于01:27:01，T2与T1之间隔着1万次哈希计算（1万个X），如果1万次哈希计算耗时约为1秒，则T2大概发生在T1的1秒后，也就是01:27:02。以此类推，所有交易事件发生的时间（时间戳）都可以粗略推算出来，这带来了巨大便利，允许节点独立确认某些数据的送达时间。

同时，POH机制也方便统计各节点给出投票的时间点。Solana白皮书中提出，Validator应该在Leader每次发布State状态信息后的0.5秒内提供投票。

如果0.5秒对应着100万次哈希计算（前文的100万个X），而Leader发布State后，后面的序列里连续100万次哈希计算都没有收录进某节点的投票，大家就可以得知这个节点在偷懒，没有在规定时间内履行投票的义务，届时系统可以执行相应的惩罚措施（**Tower BFT**）。

2.3.3.6 针对节点本身的纵向扩容

以上是Solana在出块流程、共识机制和数据传输协议上的改良，除此之外，Solana还创建了名为 Sealevel和Pipeline、 Cloudbreak的机制，支持多线程、并行、并发的执行模式，并支持以GPU来作为执行计算的部件，**大幅提高了节点处理指令的速度**，优化了硬件资源的利用效率，属于纵向扩容的范畴。

2.3.3.6 针对节点本身的纵向扩容

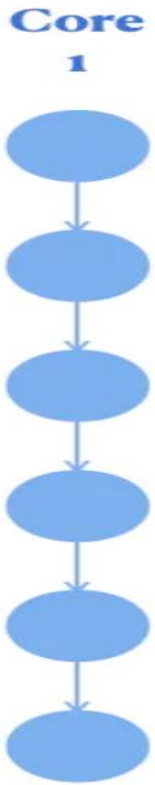
并行计算

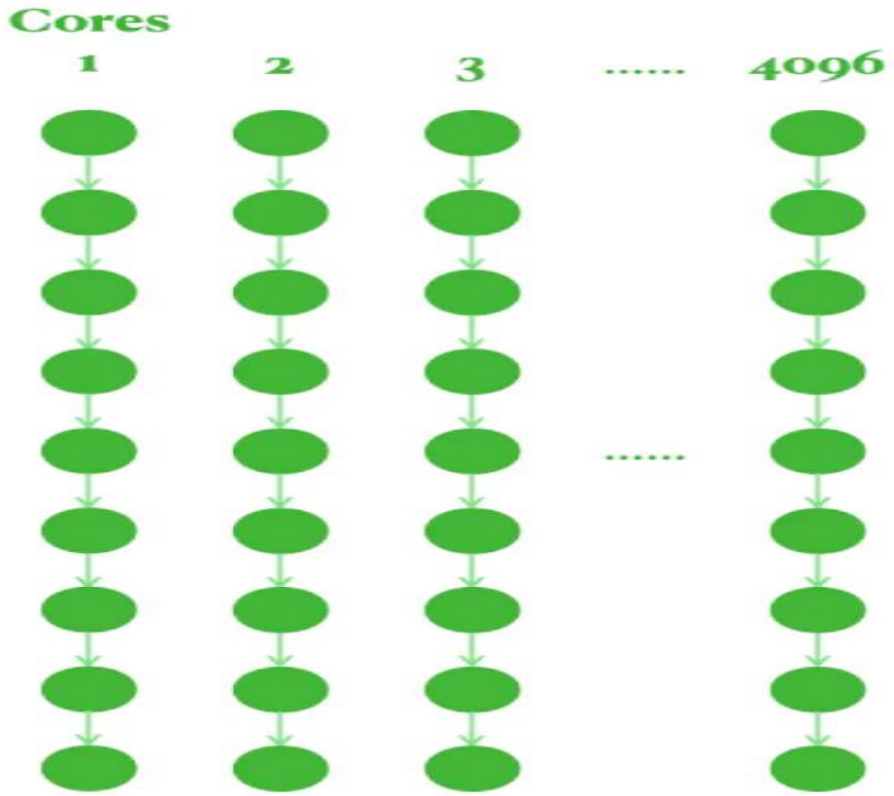
以太坊虚拟机（EVM）是单线程的——EVM 只能利用一个 CPU 核心来按顺序处理交易。由于单核产生的热量随着速度的提高而呈指数级增长，物理学限制了单核性能的上限是很低的。

解决方案是什么？更多的核心！四个 2GHz 的核心比一个 8GHz 的核心温度要低很多，但也更强大。2007 年，英特尔推出了双核的奔腾处理器，从而结束了单核时代。今天的计算机消费者拥有的 GPU 和 CPU 有 4 到 4096 个核心。让更多的核心合作得更好，而不是拥有更强大的单核，已经成为了十多年来半导体行业的研究重心。

为了实现原生多线程，Solana 必须放弃 EVM 的兼容性。Solana 的智能合约可以利用 Nvidia GPU 的 4096 个核心来并行地运行计算。

2.3.3.6 针对节点本身的纵向扩容





2.3.3.6 针对节点本身的纵向扩容

虽然Solana的纵向扩容大幅提升了节点设备处理交易指令的速度，但也抬高了对硬件配置的要求。目前Solana的节点配置要求很高，被许多人评为“企业级硬件水平”，并被斥责为“节点设备最昂贵的公链”。

以下为Solana的Validator节点硬件要求： Cpu 12或24核，内存至少128 GB，硬盘2T SSD，网络带宽至少达到300 MB/s，一般为1GB/s。 再对比当前以太坊节点的硬件要求（转型POS前）： CPU 4核以上，内存至少16 GB，硬盘0.5 T SSD，网络带宽至少25 MB/s。

考虑到以太坊转型POS后节点硬件配置要求会调低，Solana对节点硬件的要求远远高于前者。根据部分说法，一个Solana节点的硬件成本，相当于几百个转型POS后的以太坊节点。由于节点运行成本过高，Solana网络的运行工作很大程度上成为了鲸鱼和专业机构、企业的专利。

2.4 Solana的弊端

牺牲了安全性，容易宕机；

削减了去中心化；

北京大学
PEKING UNIVERSITY

PART 03

总结

3.1 总结

1. Solana扩容主要基于：高效利用网络带宽、减少节点间通讯次数、加快节点处理事务的速度 三大方面，这些措施直接缩短了出块和共识通讯的时间，但也降低了系统可用性（安全）。

2. Solana提前公开每个出块周期Slot内的出块者Leader名单，实质揭示了单一可信的数据来源，借此大幅精简了共识通讯的流程。但公开Leader信息会带来贿赂、针对性攻击等潜在安全隐患。

3. Solana将共识通讯（投票信息）作为一种交易事件来处理，TPS成分中往往超过70%都是共识讯息，**真实与用户交易相关的TPS约为500—1000；**

4. Solana的Gulf Stream机制实质取缔了全局性交易池，虽然这提高了交易处理速度，但普通节点无法高效拦截垃圾交易，Leader会面临巨大压力，容易致使其宕机。若Leader宕机，则共识讯息无法正常发布，网络容易分叉甚至崩溃；

3.1 总结

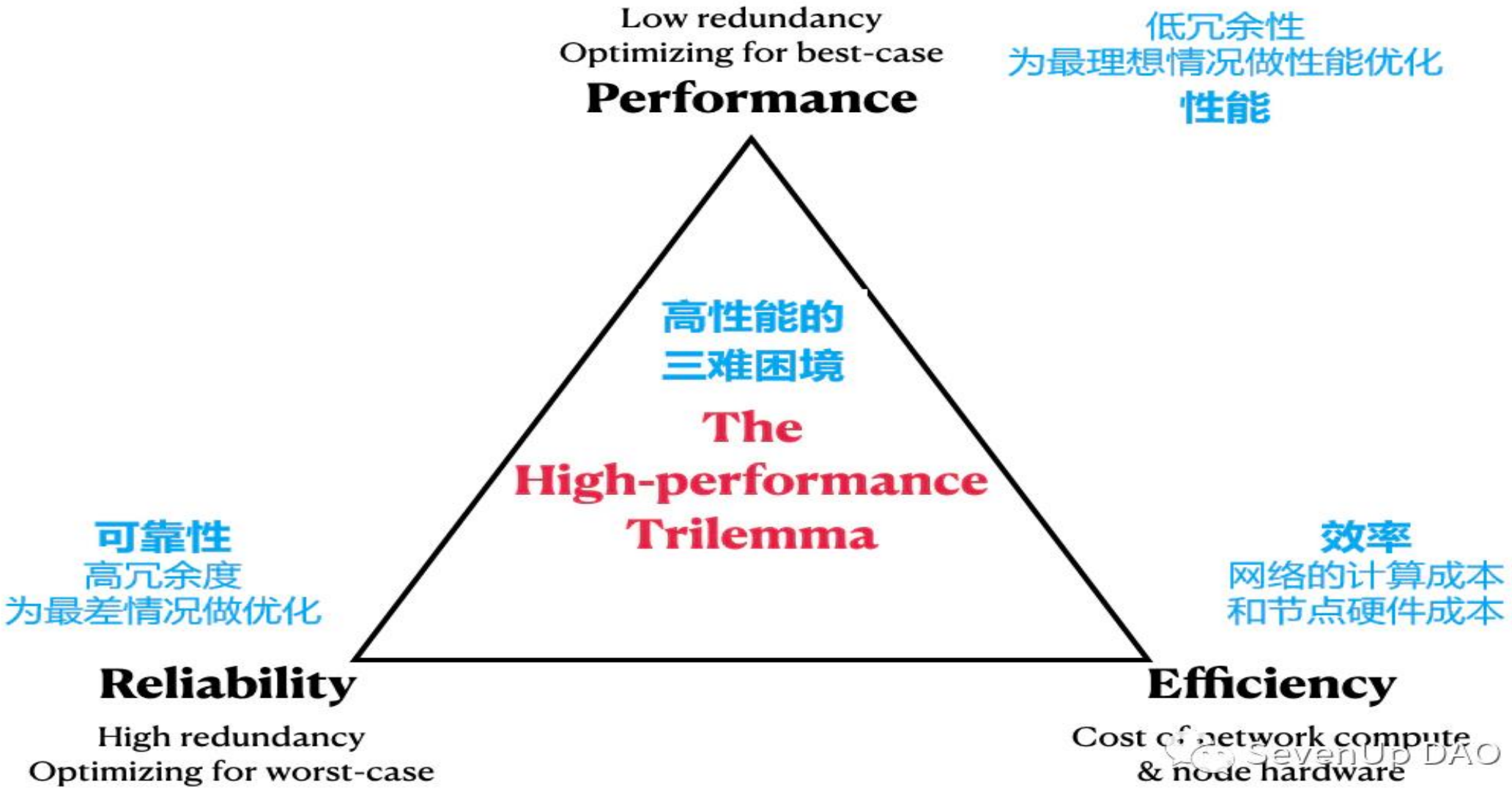
5. Solana的Leader节点发布的是交易序列，而非真实的区块。结合Turbine传输协议，交易序列可以被切碎后分发给不同节点，最终的数据同步速度极快。

6. POH（Proof Of History）实质为一种计时和计数方式，它可以给不同的交易事件盖上不可篡改的序号，生成交易序列。同时，由于同一时间只有单一的Leader发布交易序列，其中蕴含POH计时序列，Leader实质上发布了全网一致的计时器（时钟）。在一个很短的窗口期内，不同节点的账本推进、时间推移都是一致的；

7. Solana有132个节点占据67%的质押份额，其中的25个节点占据33%的质押份额，基本构成了“寡头政治”或“元老院”。如果这25个节点串谋，足以导致网络陷入混乱；

8. Solana对节点硬件水准要求较高，它在抬高设备成本的基础上，实现了纵向扩容。但这也致使运行Solana节点的个体多为鲸鱼或机构、企业，不利于真正意义的去中心化。

3.1 总结



3.1 总结

从某种角度来看，Solana实际成为了公链中最特立独行的存在，它以高级的节点硬件水准、颠覆性的共识机制与网络传输协议，将Layer1扩容的叙事推向了极端，基本触及了无分片公链可长期维持的TPS瓶颈，但Solana的多次宕机，似乎已经说明了牺牲 可用性/安全性来换取效率的最终结局。因此，在安全性、去中心化和速度的标准上新公链Aptos在以太坊和Solana之间进行了平衡，也许会将Layer1的发展带到一个新的高度。

从长远看，去中心化和安全性始终是公链领域的核心叙事，虽然Solana靠着一时的TPS数值与SBF等金融大鳄的推波助澜，一度成为资本簇拥下的瑰宝，但EOS的结局已经昭示，Web3世界不需要单纯的营销和高效率，只有真正具备可用性的事物，能够在历史洪流的冲刷中屹立不倒，永世长存。

Thanks