冰山之下：以太坊协议的中心化因子

> 一位好友多次敦促我将这份三年前的研究写成文字。如果读者读完之后觉得有些收获，那么这大概印制了一种古老的智慧：作为个体，如果我们能在或长或短的生命中做到什么事，很大的一个原因是我们并非孤身一人。可靠的朋友可以在关键时刻托付信任，弥补我们在判断力上的不足。

一. 引言

以太坊协议诞生已经 10 年了 [1]。在诞生的时刻，它是以一种解放者的姿态出现的。

以太坊协议的支持者们声称，比特币协议的可编程性是有缺陷且无可救药的 [2]；而且一些比特币的开发者们还借着一个主流客户端 Bitcoin Core 对交易转发的限定规则来审查人们想要开发的应用；以太坊协议要做的事情是将广大的应用开发者从这种可怜的境地中解放出来、随心所欲地开发各种应用。至于何以能做到这一点，则是因为它有一套精心设计的新架构。关于这套架构的特点，支持者们选择的宣传词汇是 “图灵完备（Turing Complete）”。在计算机科学中，这个词意味着你可以开发一切可计算的函数。那么，它的暗示是什么，也就不言而喻了。

然而，自实现以太坊协议的第一个点对点网络 —— 以太坊网络 —— 从 2015 年 [3] 启动以来，其协议的开发和治理却常常被批评是 “中心化” 的，并且这种批评还日复一日变得更加刺耳。这是怎么回事呢？

本文尝试论证的是，这种社区氛围和利益协调（“治理”）的中心化 —— 强烈地偏向某一个或某一小群人的意见 —— 并非因为以太坊网络的开发和以太币（ETH）的推广采取了某种历史路径、也不是因为特定人物的存在，而深深根植于以太坊协议这套精心设计的架构中。

计算机科学意义上的技术架构与政治学意义上的治理中心化，看起来似乎没有关联，但本文既不是借着跨学科的名义来掩饰思想的混乱，也不是为了故作惊人之语，更不是为了表达一种抱怨。相反，它只是为了提醒读者一种内在的危险性；当然也同意，消除这种危险是一件有意义的事。

二. 以太坊协议

以太坊协议在技术上可以归纳出这么几大特点：

“图灵完备”

在以太坊协议定义的交易执行环境中，你可以执行任意计算。这是相当吸引人的特性。

但是，这种可以编程任意计算的能力会立即带来一个实践上的难题：“图灵完备” 意味着原则上你无法判断出一段程序最终会不会结束运行、在什么时候结束运行。在点对点网络的语境下，这意味着，如果没有任何限制，你可以让整个网络中的所有计算机都执行同一个永远不会终止的程序；这可能会耗尽它们的计算资源并导致整个网络崩溃 [4]。

为此，以太坊协议提出了一个度量计算资源消耗量的虚拟单位 **Gas**，用经济代价来约束交易对节点资源的消耗：每一种计算（用于编程的每一种操作码）都会消耗确定数量的 Gas；用户在发起交易（调用一些计算）时，需要指明允许该交易消耗的 Gas 数量（`Gas Limit`），同时，指明愿意为一单位 Gas 支付的价格（`Gas Price`）。在交易进入区块、被实际执行之后，发起交易的账户（称为 “EOA（外部账户）”）将被扣除 `实际的 Gas 使用量 * Gas Price` 数量的 ETH。也即，如果运行指定的计算所需消耗的 Gas 数量未超出交易所指明的 Gas Limit，用户就无需付出预期那么大的代价。

相对应地，以太坊协议中的区块也有 Gas 消耗量限额，决定了一个节点在单位时间内要拨出多少计算机资源来处理新发生的状态转换。

> 注 1：在 EIP-1559 激活 [5] 之后，用户会以更复杂的方式表达自己愿意为一单位 Gas 付出的代价（`Max Fee` 及 `Max Priority Fee`），但基本模式不变，用户会被扣除的 ETH 数量是 `实际的 Gas 使用量 * 实际的 Gas 价格`，这个价格由一个逐块自动波动的 `base fee` 和用户指定的 `Max Priority Fee` 相加而成，但不会超过 `Max Fee`。后面我们就会看出，这种改变对我们的论证毫无影响。

> 注 2：在以太坊网络转向基于权益证明的共识机制 [6] 之后，据说还添加了一些被称作 “账户抽象” 的特性。在具体的语境下，这意味着让协议中的 “智能合约账户” 也能发起交易（原来是做不到的）。但这并不改变此处所描述的模式。也可以说，笔者懒于去证实这一点，因此单纯地假设它并不改变模式。

全局状态（Global State）

以太坊协议定义了两种不同的账户：EOA 与智能合约账户；它们各自被允许携带不同的状态，例如：前者具有 ETH 余额，还会携带一个流水号（称作 “账户 nonce 值”，记录的是该账户已确认发起过多少笔交易）；而智能合约账户则可以携带代码，还可以记录自定义的状态。

> 注 3：账户抽象希望模糊化这两种账户的区别，使它们都具备对方的一些特性。但由于重要的并非 “账户的类型” 而是 “要对状态达成共识”，因此账户抽象特性并不影响我们的论证。

重要的是两个方面：

1. 所有账户的状态也都是敞开（可见）的；也许对第三方来说这些状态无法解读、无法利用，但依然是敞开的（一些专门用来实现隐私性的智能合约账户就是这样的）；（这正是 “Global” 最初的含义）
2. 以太坊协议的运行的目标就是要对所有账户的状态达成共识。

这里的第 2 点，结合 Gas 的设计，就意味着：参与同一个运行以太坊协议的网络的节点（计算机），将不仅需要预先对各操作码的计算语义（哪个操作码到底执行什么计算）达成共识，还要预先对各操作码的 Gas 消耗量达成共识。倘若两个节点对同一个操作码的 Gas 消耗量有分歧，那么，在同一段计算执行完成之后，也可能对实际消耗的 Gas 数量产生分歧、进而对发起交易的账户应该扣除多少 ETH 产生分歧，然后脱离同步（无法达成分布式网络中的共识）。

富状态性

有一些人将此特性归结为 “图灵完备”。笔者不知道这样归结在计算机科学中算不算一种错误，但是，将两者区别开来有极大的好处，可以更准确地描述事物；而且，这种区分和定名并非笔者的原创，以太坊协议的创造者 Vitalik Buterin [7] 就使用了这种区分。

在以太坊协议中，“富状态性（rich statefulness, or statefulness）” 意味着，合约之间可以相互调用，并且，调用的深度仅受交易所指定的 Gas Limit 的限制，不受其它因素的限制。一笔交易可以调用 A 合约，而 A 合约会在运行中调用 B 合约，B 合约会在运行中调用 C 合约，除了 Gas 数量，没有什么会是这个过程的硬性阻碍。

小结

显然，我们已经触及到了以太坊协议魅力所在。可以编程任意计算，加上智能合约账户可以存储自定义状态，就意味着你可以编程出各种各样的智能合约：发行 token 的合约；使用被发行的 token 的合约；让发行出来的 token 可以相互交换的合约；等等。而富状态性意味着，不同合约可被层层叠加，最终实现一个更高级的功能（这被叫作 “可组合性”）。

此处，我们不再讨论这些结构在实际运行中会产生的一些副作用，仅仅将上述设计归纳为这样一幅图片：

“Contract States” 是智能合约账户的状态，在现实语境下，它意味着用户的各种资产（token）。但是，用户只能借助被一个具体版本的以太坊协议定义执行环境（`the EVM`，“以太坊虚拟机”）来访问这些资产。

你是否嗅到了某种危险?

> 注 4：如果读者看出，以太坊协议诞生之后出现的许多协议，也都具有我们这里所描述的设计，笔者只能说，我并不是故意的。

三. 协议的修改与治理

在基于区块链的点对点网络语境下，“治理” 意味着修改网络所运行的协议。不论是要修复某种一开始没有发现的漏洞，还是要在协议层面增加某种特性，都需要修改协议。

然而，人们可能会有各种各样的改进想法，而这又是一个承载了真实价值的点对点网络：这些想法对不同群体的利益可能有不同影响，并且，至少在一定程度上，无法单纯依靠政治命令来完成更新，而必须有某个预先沟通和理解的过程；这也是保证安全性和稳定性的必然要求。这就构成了区块链网络治理的基本边界。

显然，政治学分析不会因为这里没有暴力机构（政府）、没有明确的政治制度（开展政治竞争的程序）就失去用武之地，因为政治学分析的对象是权力的结构，而权力结构不会因为没有这些机制就消失。

回到以太坊协议。这样一套精心设计的协议，会在什么时候需要修改呢?

从历史上看，我们可以把以太坊执行过的修改大致上分成三类。

增设新特性

在这方面，一个典型的例子是所谓的 “合约原语（contract primitives）”。在以太坊的合约账户内，当然可以编程任意计算，然而，创建合约账户、合约调用时传递调用语境等操作，却不是凭借图灵完备可以实现的。仅举两个例子：

- 2016 年的 Homestead 升级中的 EIP-7 [8]，它增加了一种操作码 `DELEGATECALL`，在合约 A 调用合约 B 的时候，可以传递调用语境、将合约 A 的调用者（比如用户 U）作为合约 B 的调用者（此前已有的 `CALLCODE` 在此情形下，会让 A 成为 B 的调用者）。
- 2019 年的 Constantinople 升级中的 EIP-1014 [9]，它增加了一种操作码 `CREATE2`，可以凭借合约代码的内容得到一个确定的合约账户地址。

改变某种资源的定价

尽管在合约账户中可以编程任意计算，却不能排除这样一种可能性：相关的计算因为 Gas 消耗量过大（过于昂贵）而无法得到市场参与者的采用。这方面的一个典型的例子就是密码学相关的计算。以太坊开发者们的应对措施是，添加一种被称作 “预编译合约” 的合约账户，在调用此种合约账户的运算时，Gas 消耗量可以大幅减少，从而增强这些密码学原语在以太坊协议中的经济性。仅举一例：

- 2017 年的 Byzantium 升级中的 EIP-196 等 [10]，不仅预编译了让 ZK-Snarks 技术变得经济的椭圆曲线和标量乘法操作，还预编译了大整数模幂运算（这是 RSA 签名验证的基础）。

在另一个例子中，可以说，被影响的并不是计算的定价，而是 “持久化静态存储空间” 的定价。

- 2019 年的 Istanbul 升级中的 EIP-2028 [11]，降低了 `CallData` 的 Gas 消耗量。交易的 `input data` 字段所携带的数据被称为 `CallData`，它不是合约的状态，也不能被后续交易回访，因此，对节点而言，此种数据只是区块的一部分，只会占据一定的硬盘空间（不会占据内存）。
  - EIP-2028 将这样的静态存储资源的价格从 68 Gas/字节 降到了 16 Gas/字节。这就大大提高了此时孕育中的一类叫作 “rollup” 的 Layer-2 项目的经济性：为了让用户可以免许可退出，这些 rollup 要将自身网络中的区块发布到以太坊区块中，正是以 `CallData` 的形式发布的。（在笔者看来，这同时也意味着，在这种技术的 “可扩展性” 效果的论证中，有虚伪的一面。）

在以太坊网络的历史中，为数众多的协议层修改都可归入此类。

> 注 5：或许对读者而言，将 “Gas 消耗量” 作为 “使用某些资源的价格” 有些难以理解：因为 Gas 消耗量自身不会完全决定用户要付出的代价，这样的代价在很大程度上是由发送交易的时刻、竞争进入区块的需求 —— 表现为 Gas Price —— 决定的。对此，有一个简单的解释：价格总是相对的；当一种操作码的 Gas 消耗量降低、而其它操作码的 Gas 消耗量不变，那么这种操作码就变得 “更便宜了”；如果不同的代码能达到相同的目标，那么更密集使用这种操作码的代码就会更受消费者欢迎（给定 Gas Price，其代价更低）、开发者也会被吸引去开发更多使用这种资源的应用。

应对状态膨胀

尽管这一类升级在一定程度上也可被归入上一种，但我将它单列出来，因为本质上，上一类升级的出发点是增加吸引力（刺激采用），而这一类升级的出发点却是自我保存。

在上一章的 “全局状态” 一节，我们提到，每个 EOA 和智能合约账户都会形成自己的状态。出于一些安全上的原因，这些状态无法随意清除（请问 EOA 为什么要携带流水号？）。而且，根据全局状态的定义，即使要清除，也只能全体一致地清除（否则就会打破分布式共识）。

因此，在一个运行以太坊协议的网络中，随着时间（也就是 EOA 数量和智能合约账户数量和复杂度）的积累，状态的数量会越来越多。这就是所谓的 “状态膨胀” 问题。

状态膨胀对网络的长远生存是个威胁：对节点而言，状态数据是一种（在验证区块期间）可能随时会被访问的数据，而且，无法预知将要访问哪些片段，因此，它要放进随机读写性能更高的存储区域（比如内存或固态硬盘）中，然而，这类存储空间不仅更稀缺、更难升级，也更昂贵。因此，它会提高运行节点的门槛。更具体地说，用户要访问与自己相关的某一合约账户的某一状态，就如同在水池里捞一根针；如果水池会越来越大、越来越深，捞出这根针自然就会变得更困难。当一个节点难以在那么大的水池里搜索一根针，其区块验证时间会自然拉长，然后无法跟上网络中的其它节点的步伐，然后自然凋亡。

也就是说，在状态膨胀的大背景下，要么我们需要逐渐提高 读取/写入 合约状态的操作码的 Gas 消耗量，以提高其相对于其它资源的价格、鼓励使用其它资源（也即减少某种使用模式，鼓励平均情形的出现、同时减低最差情形的危险性）；要么，我们需要降低区块的 Gas 消耗量限额（在整体上降低节点的处理量要求）。否则，在最差情形（一个区块满是合约状态 读取/写入 操作的情形）中，可能许多节点都会崩溃。

以太坊协议开发者的选择几乎总是前者。

- 2019 年的 Istanbul 升级中的 EIP-1884 [12]，提高了合约账户状态 读取/写入 操作的 Gas 消耗量。
- 2021 年的 Berlin 升级中的 EIP-2929 和 EIP-2930 [13]，前者类似于 EIP-1884，后者则会给提前指明自身要访问哪些合约状态的交易提供 Gas 折扣（它们处理起来要更为容易）。

小结

本章的用意并不是分析以太坊网络历史上发生过的协议修改的内容。或者说，如果仅看这些内容，读者可能会认为，在以太坊网络中，协议的修改总归是有些理由的，是理性的。

然而，你是否想过，这些 “升级” 究竟是采取什么方式进行的、它的过程是什么样的？

本章几乎所有的参考文献都来自同一个网站 `ethereum.org`。尽管无法确定网站制作者的身份，它看起来是相当正式的。然而，在列举以太坊历史 “升级” 的清单页面中，不知有意还是无意的，却完全没有回答这个问题。甚至它都没有提到我们接下来会介绍的一种重要的分类。

> 注 6：此处将升级两字都打上了双引号，并不是为了讽刺上面提到的任何一项协议修改，只是笔者在撰文过程中突然发现，这个词会表达一种强烈的暗示。也正因此，上文都使用更为中性的词语：“修改”。

四. 硬分叉、软分叉、Gas 与中心化

介绍区块链网络治理的文献常常从治理的参与者群体以及它们的互动过程的角度，来描述协议修改的过程。通常，被划分出来的群体有：

- 矿工。在使用 PoW 共识机制的网络中，他们是出块者，在提供工作量证明的过程中获取新增发的货币以及交易的手续费。
    - 在挖矿活动日益集约化的现在，其他群体往往会把 “矿池运营者” 当作矿工的代表。
- 用户。本文此处采取的是较为泛化的定义，也是更为通行的定义，主要包括持币者和各类应用的使用者。
    - 在部分语境下，“用户” 指的是 “全节点运营者”。为便于讨论，笔者将它单列出来。
- 开发者。又可以区分为 “协议开发者” 和 “应用开发者”。除了工作方向的区别，还暗示了他们可能在部分问题上立场有别。
- 其它利益相关方。比如大交易所的老板，或者大矿机企业的老板。

协议的修改有自身的技术内容，这些内容首先会（也必须获得）开发者群体内部的认可，然后向用户和矿工宣传；部分提议可能引发矿工或应用开发者的不满；这些不满可能在各方让步中得到解决，或引发剧烈的冲突 …… 差不多就是这样吧。

真的吗？

本文要指出的是，治理参与者的视角有其意义，但某种程度上只触及了表象。还有更根本的东西，先一步地决定了这些参与者的选择范围和相对优势。为了理解这一点，我们先要解释两个概念。

硬分叉与软分叉

在区块链网络语境下，“硬分叉（hard fork）” 指的是对共识规则作扩大处理，这种扩大化，会将一些修改前的节点认定是无效的行为定义为有效的。这就意味着，在修改之后的（使用新的共识规则）的节点之间，可能产生让修改前的节点认为是无效的区块 —— 一旦出现了这种情形，两种节点就无法共存在同一个网络中。网络分裂之后，具体会发生什么事 —— 谁将存活、谁将消亡、会不会共存 —— 还取决于其它因素，但有一点是确定的：使用新旧两套共识规则的节点无法共存在同一个网络中；必定只有一方才能继承原来的名字、货币符号以及关注。

相反，“软分叉（soft fork）” 则是收窄共识规则；也就是说，修改后的节点仅仅会适用更严格的交易验证规则，而不可能产生旧节点认为是无效的区块；相应地，旧节点即使不升级，也不会跟新节点分裂成两个网络。

要指出的是，尽管可以说硬分叉可以添加的特性的范围会比软分叉更大，但没有人否认两种分叉都可以添加新特性；关键是这些新特性的实现方式：它是否做到了向后兼容（backward compatible）。做到了就是软分叉，没做到就是硬分叉。

在我们进一步阐述它对社区氛围影响之前，猜猜看，上一章所提及的以太坊协议修改，是采取了硬分叉形式，还是软分叉形式？

它们无一例外，全部采取了硬分叉形式；不仅是它们，在上文引用的那个 ethereum.org 网页，整个网页记录的全部是硬分叉。

为什么会这样呢？

全是硬分叉

答案就藏在以太坊协议自身的设计中。而且，有第三章的细节相对照，是一目了然的。

- 增加了新操作码的修改，必然是硬分叉：使用了新操作码的交易，对新节点来说是可以理解的，但对旧节点来说就是无法理解的无效交易；
- 改变某种资源的定价，也必然是硬分叉：操作码的计算语义虽然没有变，但 Gas 消耗量改变了，旧节点如果不更新，是无法跟新节点达成同步的（双方在交易处理完成后会对应扣除的 ETH 数量产生分歧）；
- 应对状态膨胀，也采用改变操作码的 Gas 消耗量的方式，也是一样的道理。

> 注 7：还有一类多次被应用的修改，是更改 PoW 工作量证明机制的参数，当然更只能是硬分叉。

既有的 Gas 机制和全局状态的结合，就已经决定了，不论是为了刺激采用、完善编程体验还是纠正积累出来的偏误、保证自我生存，服役中的以太坊协议的修改只能采用硬分叉。

硬分叉与社区氛围

孟德斯鸠将权力分成立法、行政、司法三种类型，并认为，仅当三种权力相互制衡、没有一家独大的时候，公民才能享有自由。

这种看似简单的学说提供对权力结构的惊人洞察：仅在统治者无法肆意妄为的地方，多样性才能得到生长。

曾经有人指出，比特币的治理有三种参与者：（协议）开发者、全节点运营者、矿工；他们之间存在类似于三权分立的制衡结构：开发者只能开发软件，全节点运营者可以自己决定是否要使用新的软件和新的共识规则；矿工虽然负责出块，但也无法决定人们使用什么共识规则；全节点运营者有想法，也需要依赖于开发者来实现，并希望矿工会尽快提供服务。

然而，可悲的是，这样的结构在以太坊协议的治理中是不存在的。一个简单的思想实验就足以说明问题：如果一个用户不支持一项（开发者们都支持）的修改，他能怎么办呢？

- 如果矿工们并不反对，那么他就无法反对。该用户（及其少量同志）孤立出来的小网络会因缺乏矿工而无法存活。
- 如果大部分用户都不反对，那么他也无法反对。因为预料之中矿工也不会反对。

在这两种情形中，他都无法说 “不”。说 “不” 的唯一结果就是他自己的节点被抛离网络，无法继续验证交易。

唯一一种可能反抗的情形是，半数的矿工和用户都反对：也即，唯有能够形成一个势钧力敌的网络，才能让协议保持原样。但凡不能反对者不能达到半数，最终都会是开发者一派胜出。这是一种天然倾向于开发者的权力结构，也是一种天然排挤不同意见者的结构。

相反，在软分叉中，用户是可以拒绝、可以投不信任票的，只需不换用新软件即可，这样做既无需承担由更严格的共识规则带来的更大的验证负担，也不会被踢出网络 —— 你依然可以用你自己的节点、你接受的那一套共识规则，来验证跟你相关的交易。比特币对你来说将依然是免信任的，仅仅是无法使用新特性而已。

> 注 8：敏锐的读者可能认为，这里的两种 “拒绝” 并不相同：在硬分叉例子中，拒绝指的是阻止协议被修改；而在软分叉例子中，拒绝仅指不让自身受到影响；后者当然比较容易。但其实，只要你从全节点（希望自主验证所有区块、从而免信任地使用协议的用户）的角度来看，会发现它们的目的是一样的：不希望自身受到协议修改的影响，希望继续自主验证。只不过，在硬分叉模式中，达成这个目的的唯一方法就是阻止协议被修改；而在软分叉模式中，则可以通过不升级来达成。
>
> 在后文中我们会发现，另一位作者尝试利用同样的误解来论证 “两种分叉都一样会带来强制”。

这就是为什么有人将比特币的采用模式总结为 “opt-in”，而以太坊的采用模式则是 “opt-out”。前者是在选择什么时候加入，一旦你认为某一套交易验证规则是足够安全的、你愿意接受的，你就可以加入了，而且不必担心这套规则会被修改甚至摧毁。相反，如果你采用了以太坊，新的协议修改并不一定总能让你满意，甚至会破坏掉你心爱的东西，你所得到的免信任性是朝不保夕的，你始终有一天可能发现自己待不下去了，要离开。

这也是为什么比特币社区有更浓重的平等说服文化。观念不合、怀疑和恐惧，都不必在逼仄的时间内解决，智性有足够多的时间和空间来成长。

因为比特币几乎只采用软分叉 [14]。

Gas 之难

最后，还有一个难题根植于 Gas 这个概念本身。前面我们已经说过，以太坊协议允许用户使用各种计算机资源：计算、内存（合约状态）、带宽和静态存储。理想情况下，我们应该让用户使用它们的代价与节点提供这些资源的代价相匹配，这样才能保证网络的长期生存。

然而，这些资源的使用代价，却是用同一种单位 Gas 来衡量的。这想想就很难吧？

单就计算本身，我们可以将一段计算分解为一系列的基本运算，从而恰当地将这段计算的代价定义为各个基本运算的代价之和。但这一套却无法适用在带宽和静态存储上。因为根本上，这是两种不同的计算机资源。我们既无法断定它们的恰当换算比例，也无法保证这些比例不会在未来发生改变。

关于合约状态 读取/写入 的操作码也是如此。一个在某一时刻恰到好处的 Gas 消耗量，会在日后因状态膨胀而逐渐偏离实际，甚至与实际相去甚远。没有人能科学地证明应作多久的前瞻，或应以多少时间间隔调整一次。

这些因素意味着，调整 Gas 消耗量的决定几乎总是一种政治决定，决定者无法诉诸科学而只能诉诸自身的权威。

就分工协作而言，本应尽可能减少需要这样的政治决定的范围。但以太坊协议的设计却使得这样的政治决定无法回避、无日无之。

至此，所有导向治理中心化的因素都已经准备好，且根植于以太坊协议本身。

五. 一些回顾

本章准备回顾几件可能塑造了以太坊社区成员观念的文字作品。在笔者看来，这些文字或印证了上述分析，或露出了獠牙而没被注意到，又或展现了某种避重就轻的 “风格” 。

《软分叉，硬分叉，默认和强制性》（2017）

这是 Vitalik 发表于 2017 年的作品 [15]。文章的内容与 2017 的时事（BCH 硬分叉）相关。但除此之外，笔者认为，这篇文章的写作目的是希望以太坊社区的参与者能够接受硬分叉这种修改协议的方法。

该文没有采取 用户-全节点 的视角，而强调软分叉和硬分叉都改变了协议，然后声称：只要大部分人都接受、你不接受（但你又不想离开），就对你产生了强制。首先，“意见不合” 不等于，也不必然带来 “强制” 。其次，它利用了人们对区块链网络共识规则的一种直觉式误解：不论如何，一个网络中有且只能有一种 交易/区块 验证规则，所有节点必须对此达成严格共识。然而，这种假设对以太坊这样的协议是成立的，但对于有软分叉可能性的协议来说却是错的。

该文甚至声称：硬分叉才给了用户自由，因为一定是先尽到告知义务，然后让用户自由选择；而软分叉可以在用户完全不知道的前提下进行，因此可以完全不给用户选择。

如果你无法抵御这种论证，或觉得摸不着头脑，可以想一想：一种是明白地给你强加义务；一种是如果你不知道，就无法给你强加义务。请问哪一种给了你更多自由？

笔者在读到这篇文章的三年半以后，才明白这篇文章里都是胡言。

《一种权益证明设计哲学》（2016）

这是 Vitalik 发表于 2016 年的作品 [16]。该文旁证了许多概念来论述设计 PoS 权益证明共识机制的方法。该文可能最早在 PoW vs. PoS 的争论中使用 “社会共识（social consensus）” 的概念。此后，Vitalik 多次在争论中使用这一概念，作为 PoS 机制安全性论证的一环。

笔者认为，Vitalik 提出 “一个区块链的共识规则是其参与者社会共识的自动化”，无非是想暗示 “只要社会共识改变了，共识规则就可以改变” 。

问题是，Vitalik 从未论述过，这个社会共识是如何抽取得到、或通过什么东西体现出来的。他似乎也没有考虑过政治异议的存在。因此，这等于是主张，如果有人能夺得 “社会共识” 的垄断解释权，就可以为所欲为。

这就不能不让人想起卢梭。其著作《社会契约论》讨论的是政治权力的来源，而其观点是，政治权力来自于一国公民集体的 “公共意志（公意）” 。卢梭先生同样没有讨论，该使用什么样的政治程序来汇集这样的公意、得到公权力的人又对何种程序负责。因此，这种理论实际上只是对民族国家的一种浪漫想象，而并非可以帮助确定政治架构的有用理论。实际上，可以说是一种有害的理论。你完全可以设想，一个大权在握的人可以大谈特谈自身是公意的代表，同时将权力不受限制地伸到任何地方去。—— 反正，卢梭是没有说这种公意也该受到约束。

《何为 “以太坊核心开发者”？》（2020）

这是 Hudson Jameson 于 2020 年发表的作品 [17]。Hudson 曾长期作为 “以太坊核心开发者双周视频会议（Ethereum Core Developer call）” 的组织者。这篇文章的出现背景是，一些人在社交媒体上讨论：到底什么是 “以太坊核心开发者”？谁是，谁不是？

该文 “优雅” 地给出答案：以太坊核心开发者是当前给以太坊底层开发提供重大贡献的人；但是，由于以太坊的开发是去中心化的，所以，“核心开发者” 不是某个机构能授予的头衔，也不能阻止人们自称。（这里的 “以太坊”，在上文中对应的是 “以太坊网络”）。

如果以太坊网络的开发真的是去中心化的，那么 “核心开发者” 称号具有上述特点当然是应有之意。但是，“去中心化” 在这里是个断言，并没有得到证明。

更重要的是，Hudson 完全回避了另一个问题：这群核心开发者到底能干什么？他们是不是具有一种我难以抵御的权力？

假设我们信任 Hudson 的断言，那也不过意味着：这个群体是开放的，可能容纳各种背景的来者。问题是，我们并不知道，这些人会遵守什么样的原则、受到什么样的约束。可以说，“去中心化” 被当成了一种挡箭牌，为已经引起人们顾虑的行为作辩护。

这就让人想起了以赛亚・伯林提出的著名区别 “积极自由” 和 “消极自由”，前者关系的是 “谁能统治我”，后者关系的是 “我会被统治到什么地步”。当人们隐约担心后者时，Hudson 用对前者的回答来搪塞。

小结

在笔者看来，这些文章都暗藏着一种愿望，希望人们适应甚至是服从以太坊协议所创造的世界：既然不可能实现软分叉，就干脆把硬分叉说成更好的；如果无法说明行动的理由，就诉诸 “公意”，反正只有一小部分人能利用这个名义；最重要的是，不能质疑 “以太坊核心开发者” 的身份和决定。

国家主义与自由主义在观念上的竞争，使二十世纪的自由主义已开始生产出这样的观念：处在特定权力结构中的人，即使用心善良，也无法真正给他人带来福祉，因为他们**不知道**如何才能做到。笔者希望这些治理中的得益者是且只是这种情形，但经验使笔者很难不往相反的方向想。

中心化的社区氛围形成之后，带来了许多更让人啼笑皆非的事情。但人们无疑都很聪明，日复一日，外部观察者可以看到一个同心圆结构变得越来越明显、越来越庞大。

六. 结语

本文勾勒了以太坊协议的结构，并以此解释了何以其协议修改总是采取硬分叉的形式；这种硬分叉习惯，又如何导致了社区氛围的中心化。在本文末尾，文本分析和政治学分析提供了独立的视角和模糊的印证。

本研究脱胎于三年前为 “无状态以太坊” 作的研究，这是一个旨在解决 “状态膨胀” 问题的技术方向。然而，当你发现所有的协议修改最终都要落脚为 Gas 消耗量的规范，而 Gas 消耗量的设计如此强有力地塑造了一个倾向于开发者的权力结构时，实在很难不让人感到疲惫。

从以太坊协议研究的角度看，这两份研究都提出了有价值的话题，值得珍视，也可以作为后续进步的垫脚石。不过，笔者已不再关心这些话题了。如果这篇文章会引来什么批评与争议，当然也都与笔者无关，笔者也不会再关心。

（完）

参考文献

1. https://ethereum.org/en/whitepaper/ ↩

2. 在 2017 年的以太坊开发者大会上，以太坊协议的创造者 Vitalik Buterin 在开场演讲《Ethereum in 25 Minutes》中，将比特币协议 “生动” 地比喻成一台中学生用的科学计算机；或至多不过是把瑞士军刀。见：archive.devcon.org 。 ↩

3. https://ethereum.org/en/history/#frontier ↩

4. 2019 年出版的《Mastering Ethereum》 一书在第一章 “Ethereum Basics” 就指出：图灵完备的计算机是很容易设计出来的；困难的反而是如何限制这种灵活性。 ↩

5. https://ethereum.org/en/history/#london ↩

6. https://ethereum.org/en/history/#paris ↩

7. https://vitalik.eth.limo/general/2019/12/26/mvb.html ↩

8. https://ethereum.org/en/history/#homestead ↩

9. https://ethereum.org/en/history/#constantinople ↩

10. https://ethereum.org/en/history/#byzantium ↩

11. https://ethereum.org/en/history/#istanbul ↩

12. https://ethereum.org/en/history/#istanbul ↩

13. https://ethereum.org/en/history/#berlin ↩

14. https://www.btcstudy.org/2022/09/02/has-bitcoin-ever-hard-forked/ ↩

15. https://vitalik.eth.limo/general/2017/03/14/forks_and_markets.html ↩

16. https://medium.com/@VitalikButerin/a-proof-of-stake-design-philosophy-506585978d51 ↩

17. https://hudsonjameson.com/2020-06-22-what-is-an-ethereum-core-developer/ ↩