VRF + BFT 共识引起交易失败的问题

2022-09-03

昨天遇到一个问题，是 Ontology 的节点不出块了。节点使用的是 VBFT 共识，网络互通没有隔离，也就没有分叉，报的错是提案（proposal）过来的块，和预期的块哈希（MerkleRoot）不一样。由于种种原因，昨天的问题没有深入去查，用比较暴力的方法先让网络恢复正常。不过提到 VBFT，我想到了以前公司的一些事情。

我是在之前的公司开始接触区块链的，当时的项目号称自研区块链，也是用 VBFT（VRF + BFT）的共识，不过在共识方面不那么说，叫成 UBFT 还是什么。

我猜测 VRF + BFT 的主意是 Ontology 提出来的，我之前的公司把这种创意抄了过来，模仿着实现了一下。说来讽刺。

主要想说的是之前公司由于对 VRF + BFT 不靠谱的实现，引起的一个隐藏的 bug。那件事情距今快 2 年了，不记得当时为什么没有写博客记录一下，可能是在忙着做 PPT？昨天遇到共识相关的问题，我想起有那么一回事，正好现在有时间有心情写一下。由于过去时间太长了，细节上可能有出入。

背景

先介绍一下那个项目的情况，主打的特点有几个。

一个是异构多链，含义是可以在同一个节点上包含多条异构的链。异构是指一条链可以使用不同的共识机制、基于不同数据库运行起来。多链是指多条链可以在同一个节点上运行，因为觉得一条链不够用，一条链就相当于传统业务里的一张数据表，多条链可以方便地进行数据拆分，在联盟链的场景下更好地支持业务。异构多链，可以理解为把以太坊不同 chain id 的链，用同一个二进制包启动了。现在的某条开源联盟链，还在拿从异构多链演变来的灵活装配作为一大特点呢，猜猜为什么。

再一个是多数据库的支持，同时支持很多关系型数据库和非关系型数据库，做法是针对各种数据库，写数据操作的中间层做适配。

还有就是共识机制方面，基于开源的 Tendermint 项目。Tendermint core 是一个对 BFT 类共识的实现，在那个基础上，做的改动是把轮询选择提案节点，用 VRF 函数，替换为随机选择提案节点。另外还增加了对分层共识的支持，也就是共识组的概念，每隔多少个块换一次共识组，换共识组的方式借鉴 BFT 的流程，保证换共识组过程的安全性。分层共识这个理念也不知道起源于哪儿，可能同时期的项目流行这个？

项目的这些特点都是在我接触之前就已经开发完成的，我也只是有所了解。

我当时遇到的问题是，如果向区块链发送一笔失败交易，节点会立即返回交易失败的结果，然后如果再次发送一笔失败交易，第二笔交易的结果将迟迟不返回，节点不出块了。接下来如果仍然是失败交易，第三笔、第四笔，都会是同样的现象。这个时候，如果发送一笔正确的交易，节点会立即返回结果，之后一切恢复正常。而且这种现象是概率性出现的，并不是每一次失败交易都会引起问题。

前提

首先是失败交易指合约返回执行结果为失败的交易。区块链系统的交易失败有两种，一种是交易不能被执行，另一种是交易能被执行，但是合约中返回了合约层面的失败。那个项目并没有严格区分这两种失败的类型，合约有权限返回交易层面的失败，这其实是有问题的设计。

不正确的交易，将会在提案的时候，被忽略掉，因为不正确交易没有必要记录在区块链上。加上项目对失败交易错误的处理，造成的现象就是，合约执行失败的交易，会被忽略掉。这是前提。

BFT 共识的基础，是投两轮票，最终确定一个块。不管是什么 BFT，在前面加什么字母，不管通过多么复杂的流程决定出哪个节点提案、如何提案，不管对共识的效率做什么优化，是并行提案还是流水线共识什么的，只要是 BFT 类的共识，都是投票两轮。对两个阶段的命名可能不一样，不管是用 proposal 还是 prepare 来描述，都是那样一个过程。

BFT 的流程，是先有一个节点生成一个块，然后把这个块发送给其他节点，如果超过 2/3 节点同意，会进行下一轮投票。第二轮投票如果超过 2/3 节点同意，这个块就算确认下来了。两轮投票是理解 BFT 共识的关键。至于为什么投票两轮就可以达到 3f + 1 的容错效果、为什么至少要两轮，我也不知道。

忽略掉失败交易的操作，是在检查交易的过程中完成的。项目里有两次检查，共识前检查和共识后检查。有一些交易是没办法在共识前进行检查的，比如在合约里进行的写数据库操作，如果共识前就写库了，然后共识失败了，数据不就乱套了吗。所以只能共识后进行检查。这是第二个前提。

第一笔失败交易

我们根据 bug 的现象分析一下，第一笔失败交易的流程是正常的。一笔失败交易进来，在共识之前是不会检查出失败的，所以预提案的节点正常提出了一个块，分发给其他节点，进行第一轮投票。之后正常进行第二轮投票，在确认块的阶段，写入块之前，会进行共识之后的检查，检查过程中发现交易失败，并且这个块只包含这一笔交易，这个块就作废了，没出块。同时，其他节点也都会返回消息告诉提案节点，这个块没出来，这笔交易失败了。所以第一笔失败交易是正常返回结果的。

第二笔失败交易进来，按照同样的处理流程，一切都应该是正常的才对。因为即使是失败交易，即使是在确认块的阶段，如果检查失败了，也会广播处理结果给其他节点。整个协议中投票失败或者落块失败，都是用空消息表示。其他节点不会因为交易失败，就收不到消息苦苦等待超时。那么既然 BFT 协议的流程没有问题，为什么还是出现 bug?

这个时候要提到项目在 VRF 方面的改造。

在 BFT 的协议中，是需要一个节点去生成一个块，分发给其他节点开始进行第一轮投票的。那么由哪个节点来进行这个生成块的操作呢？总不能是同一个节点吧，那就太中心化了。Tendermint 的做法是依次进行，比如有 4 个节点，第一次节点 A，第二次节点 B，这样轮询。

VRF（Verfiable Random Function）做的事情，是改变依次选择节点的方式。因为如果按照顺序来，那很容易预测到下一轮要由哪个节点去生成块，顺序可以预测之后，就存在节点被贿赂、节点被攻击等安全隐患。VRF 的功能是参数相同结果一定相同，参数不同则结果随机。把块高度、投票的轮数作为参数，就可以很好地实现，每一个块都能由随机的节点来生成，无法预测。这个改动也是作为项目的一个亮点的。

不过 VRF 存在一个问题，既然是随机的，那就有一定可能，第一次随机到节点 A ，第二次也随机到节点 A，这样的概率还是不小的。如果节点 A 是恶意节点，然后由节点 A 连续两次生成块，会给网络带来一些负担，虽然不至于破坏网络，但也是一点小小的麻烦。所以项目为了解决这个问题，在 VRF 的基础上加了黑名单的机制。

如果上一轮是节点 A 生成块，就把节点 A 放到黑名单里。如果 VRF 的结果在黑名单里，就再 VRF 一次，避免重复选择相同的节点。

第二笔失败交易

不返回结果

结合 VRF 和黑名单，再来看看第一笔失败交易发生了什么。节点 A 收到交易，会先把这笔交易广播给其他节点，然后打包成块进行投票的流程。此时节点 A 在黑名单里。投票失败后，节点 A 返回失败，并且这笔交易已经不在节点 A 的交易池里了，因为已经处理过了。

那么节点 B 呢？块里面的交易验证失败了，但是交易池里收到的交易还在，因为这笔交易还没有处理啊，处理的只是广播过来的块里面的交易。这个时候是不是应该把交易池里面的交易删掉？对，但是没删。所以造成一个问题，节点 B 被选作生成块的节点，把这笔交易打包了一下，广播了出去。这个块当然也是提案失败的。此时节点 A、节点 B 都在黑名单里。

以此类推，就这一笔交易，一轮下来，4 个节点全在 VRF 的黑名单里。但是对这笔交易结果的返回是没有影响的，因为交易结果在节点 A 的时候就已经返回了。

第二笔失败交易过来了，所有节点全在黑名单里，会发生什么？当然不能选不出节点，节点全在黑名单里，黑名单就失效了。VRF 的结果是哪个节点，就是哪个节点。

分析一下第二笔失败交易。同样是节点 A 收到交易，假如这一次是节点 B 负责生成块，然后这个块验证失败了，节点 B 就会删掉这笔交易，对吧，这个没问题。

注意，删掉交易的同时，通知客户端，交易失败了，返回交易结果。节点 B 被选中，节点 B 生成块，节点 B 返回通知。但提交这笔交易的客户端，连的是节点 A 啊！

第一笔失败交易为什么会收到响应？因为黑名单还没有失效，所有节点都处理了一遍交易，所有节点都返回了一遍交易结果。现在黑名单失效了，只有节点 B 会返回结果，所以节点 A 的客户端收不到交易结果。

那为什么黑名单失效，就不能像第一笔交易一样，所有节点都处理一遍？

阻塞后续交易

接着分析一下第二笔失败交易，节点 C 还有交易啊，节点 C 上面的这笔失败交易还没处理呢，节点 C 就开始用 VRF 选节点了。

刚才提到，此时黑名单失效，VRF 选出哪个就是哪个。刚才选出了节点 B，这一轮有没有可能再选一次节点 B？黑名单失效，就变得可能了。这个时候如果又是节点 B 负责生成块，会发生什么？

节点 B 生成不了块，因为节点 B 已经没有交易了，它已经把唯一的失败交易，在上一轮就删掉了。也就是说，在新一轮的共识过程中，4 个节点全部在等节点 B 生成块，节点 B 自己也知道该自己了，但是节点 B 拿不出块，节点 B 直接放弃这一轮共识，进入下一轮，并且节点 B 没有发出任何消息。

在分布式系统中，没有消息是一件可怕的事情。其他节点都在等节点 B 呢，节点 B 自己玩了。这个时候，节点 B 的轮数要比其他节点快一轮。

在 BFT 共识中，有两个索引值，一个是块高度，一个是共识的轮数。同一个块高度，有可能因为块没确认，就经过很多轮共识。由于节点 B 自己没生成块，轮数增加了，其他节点还不知道。

现在，所有节点都在 VRF 的黑名单里，节点 B 的共识轮数高于其他节点，

如果节点 A 再收到失败交易，有两种情况。一种情况是 VRF 又选中节点 B 了，节点 B 提出的块会被拒绝，因为其他节点还在等节点 B 上上轮的块，它拿出了高轮数的块，是对不上的。另一种情况是，VRF 选中了其他节点，那其他节点首先要等节点 B 上上轮的出块超时。超时之后，其他节点把轮数最高的数值同步一下，共识就算恢复正常了，然后 VRF 再选。

但是注意，这可是一笔失败交易，此时黑名单仍然失效，即使共识恢复正常，也还是有概率重蹈整个覆辙，节点 A 仍然收不到交易结果。至于具体的概率是多少，就懒得算了。

总结

可以看到，这个 bug 是由很多系统性的不合理设计共同造成的，直接原因在于 VRF 黑名单的失效，因为所有节点都在黑名单里了。或者说，黑名单没有及时清空，原先的错误之处在于，只有

在块高度变化的时候才清空黑名单，可能是认为每个块的产生都应该由不同的节点来处理。这种想法的失误在于忽略了相同块高度的时候，共识的轮数也会发生变化，每一轮都会产生一个新的块。所以只要在共识轮数发生变化的时候，也清一下黑名单就好了。实际的代码改动只有两行。

上面写的东西，可能我自己也不想仔细去看，不好理解、抽象，而且文字的表达能力也弱，看起来费劲。这种类似状态机状态转换的文字描述，看起来是很痛苦的事情。尤其是内容和当时项目的耦合很深。总的来说，对于这个问题的分析和解决，我认为在逻辑上是自洽的，能很好的解释成因和现象，以及用最简单的方式在表面上修复它。

时隔近 2 年的时间，我竟然还能记起来这些，感觉也是很奇怪。