**[[1]](#footnote-1)撰写交底书须知：**

专利申请文件的撰写是发明人和代理人共同完成的工作，因为技术思想只能来自发明人；只有发明人将自己的技术思想准确、完整、深刻地表达出来，代理人才能够以此为基础，进行有效地挖掘，为这些技术在法律上确定恰当的位置。

您的有效配合是我们工作效果最有效的保证。

谢谢！

**技术交底书撰写模板**

发明名称：基于“弹劾”的区块链容错与故障恢复机制

本专利申请的发明人：

技术交底书撰写人及技术联系人：施炎辰

电话：\_ 18257733256 \_\_ 手机:­­\_\_\_\_18257733256\_\_\_\_\_\_\_\_\_ E-MAIL: alex031029@gmail.com ；

**注意：**

**1、尽量避免使用外文单词；如果必须使用，所有外文都应给出中文译文或中文解释。**

**2、全文对同一事物的名称应该一致，避免出现一种事物多种叫法。**

**3、为了方便代理人理解，建议多采用一些直观图形来说明，如，数据表、信令图、效果图、界面截图等。**

**※4、撰写人必须将以下前五项内容填写完整，第六项可根据实际情况选择性填写。**

**1、请阐述一下发明名称中所涉及的核心主题**

该核心主题要完成什么功能？此处的描述要尽可能的通俗。针对较为抽象的技术，请结合具体的应用场景来说明该主题所要实现的功能。

主要实现一个基于“弹劾”区块链容错与故障恢复机制。区块链上每个区块有且仅有一个节点可以进行出块。在某些共识机制，比如股权证明（PoS）或者拜占庭容错（BFT）中，出块节点是根据一定的算法跟随机数事先指定的。然而这个指定指定的节点，在轮到它出块的时候可能会出现意外导致不能出块，这个时候需要一种容错机制来应对这种情况。此外，当整个链停止之后，我们需要一个故障恢复机制来重启整个系统。对此，我们提出了一种基于“弹劾”的区块链容错与故障恢复机制，从而让其他节点完成替出问题的出块节点履行出块义务，保证了区块链的活性与安全性。

**2、我们设计的新技术方案的设计动机或初衷**

请结合具体业务场景回答以下问题：

（1）如果是存在新的业务需求使得我们设计出新技术方案的话，该业务需求是什么？如果不清楚业务需求是什么的话，此次也可以介绍一下你的领导给你布置了什么任务？并且，该任务在技术上的要求是什么？

（2）如果是发现我们已有的技术方案或者别人已有的技术方案存在某些问题是的我们设计该新技术方案的话，请写明存在的问题是什么并最好描述出这个问题的产生原因。（需要注意的是，这里的问题一定是新技术方案针对性能够解决的问题）

目前的区块链项目中，常用的共识机制为工作量证明机制（Proof of Work,PoW）与股权证明机制（Proof of Stake, PoS）以及股权授权证明机制（Delegate Proof of Work，DPoS）。

作为一个分布式项目，这几种共识机制都需要面临一个共同的问题：如何面对出块节点并不出块的情况。面对这个问题，共识机制可以分成两大类。第一类是指定某个块高出块节点，第二类是不指定某个块高的出块节点。所谓的指定，也就是对于未来的某个块，事先就通过一系列的算法指定了这个块的出块节点。

其中，所有的工作量证明机制与部分股权证明机制均是不指定出块节点的共识算法。在这样的算法框架下，如果一个在算力上最有优势的节点由于个人问题（如宕机、网络故障）等导致不能出块，那么其他节点马上就会抢先出块。

一部分股权证明机制与全部的股权授权证明机制，都在某个块出块之前指定了出块节点。这个指定是通过一定的算法随机产生。比如在股权证明机制中，拥有越多股权（即抵押的代币越多）的节点，将有越高的概率被选中成为某个未来块的出块人。当这个指定的出块节点发生了意外导致不能出块，能采用的策略大体有以下两种 1）盲等，直至该出块节点回复正常；2）跳过该区块，让下一个出块节点出块。这其中盲等的方法虽然实现上非常简单，但是在实践中并没有可行性，因为盲等并不知道会等多久，整个链可能失去响应。第二种跳过这个出块节点便是绝大多数股权授权证明机制所采用的办法。实践上，这种跳过故障出块节点的方法，需要一个计时器来完成。当计时器提示超时的时候，下一个节点才能出块，并跳过这个故障节点。那么如何设计这个计时器将是这种类型的区块链都必须面对的问题。同时，由于下一个块的块高已经指定，这个本来由故障出块节点负责的块必将是一个空块。那么这个空块如何验证，又有谁来出代替这个空块也是一个需要解决的问题。

目前很多区块链在以上描述的共识算法中融合了拜占庭容错算法或者Paxos算法，从而提高容错率。比如脸书（Facebook）的天秤座（Libra）就融合了拜占庭容错算法。无论是拜占庭容错算法、Paxos算法亦或是其他类似的分布式共识算法，都会有一定的前提以保证能够能够处理错误。比如在经典的“实用拜占庭容错算法”（Practical Byzantine Fault Tolerance, 以下简称PBFT）算法中，一个包含3f+1数量节点的系统最多可以允许f个结点有拜占庭错误（即包括作恶、无连接等在内的任何错误类型）。理论上，PBFT算法在不超过f个节点有拜占庭错误的前提下，能够完美工作。然而事实上，一个现实的分布式系统不可能永远满足这个条件。于是我们需要一个能够在前提被打破的情况下，能够重启系统的故障恢复（failback）机制。由于区块链分布式的特点，每个验证节点本身的时间戳并不相同，让这些刚刚重启的验证节点如何达成重启之后的第一个块的共识，正是故障恢复算法设计的难点。

那么总结一下，现在的方法存在的问题是：

1. 在跳过错误节点的共识算法中，如何设计计时器。
2. 这个因为跳过的空块由谁进行验证。
3. 拜占庭算法中如何进行故障恢复。

**3、请结合2中的所确定的设计动机或初衷，用概括的语言，描述新技术方案是如何实现的**

（1）如果是实现新的业务需求（或者是完成领导布置的任务），请概括说明新技术方案是采用何种新的手段实现该业务需求的（或者完成领导布置的任务的）。在概括说明中，请突出谁是新的手段以及该新的手段和实现业务需求（或完成领导布置的任务）之间的关系。

（2）如果发现我们已有的技术方案或者别人已有的技术方案存在某些问题，请概括说明新技术方案是采用何种新的手段解决该问题的。在概括说明中，请突出谁是新的手段以及该新的手段和解决现有问题之间的关系。

我们结合了拜占庭容错算法和区块链共识算法，设计了一种可用于股权证明机制与股权授权证明制的容错方法——“弹劾”机制。并且我们利用弹劾机制，设计了用于的故障恢复的算法。

弹劾机制是基于拜占庭容错算法（如PBFT）上的改进，核心在于所有的验证节点共同出一个弹劾块以替代故障的出块节点。在PBFT算法或者其他的经典拜占庭容错算法中，验证节点（PBFT中称为replica，即拷贝）中的首领（PBFT中称为primary）负责接收一个出块节点（PBFT中称为client，即客户）发出的区块（PBFT中称为request，即请求）。如果这个首领节点发生故障，PBFT中采取的方法称之为“视图变换”(view change)，即放弃这个首领，直接更换下一个首领。

我们可以将这个方法运用在区块链中，即“弹劾”。每一个验证节点各自维护一个时钟，用于及时从上一个正常块到现在的时间。如果这些验证节点的时钟超时，就会广播一个弹劾块。这个弹劾块在所有的正常工作的验证节点中都具有相同的结构与内容。其内容可以不仅仅是一个单纯的空块，还可以包括对故障节点的惩罚。当有这些广播弹劾块的验证节点达到一定的数量，这些验证节点就形成了一个弹劾共识，从而出一个弹劾块到区块链上，并成功跳过了故障的出块节点。

而对于故障恢复，我们假设所有的验证节点全部宕机（即最坏情况）。在这个情况下，所有的块都不能得到验证，可以说整条链已经完全停止，需要重启验证节点。此时，我们将不能依赖于出块节点制造重启后的第一个块，而且需要重启的验证节点互相统一时间戳，达成共识从而出重启后的第一个块。这个块由于没有出块节点的参与，必然是一个弹劾块。对此我们设计了若干个弹劾块，各自相隔一定的时间戳，从而保证验证节点必然能够对其中某一个弹劾块达成共识，统一时间戳，并进而完成整条区块链的故障恢复。

**4、新技术方案具体是怎么做的**（提示：本部分为一个专利申请最重要的部分，需要详细提供一个完整的技术方案，不能只有原理，也不能只做功能介绍。

这个部分我们讲分四个方面来介绍：1）如何设计验证节点时钟；2）如何设计弹劾块内部结构； 3）弹劾块如何共识；4）如何利用弹劾机制设计故障重启

1. 如何设计验证节点时钟

基本上所有指定未来区块出块节点的算法，都具有一个特点：即每个区块之间相隔固定时间。比如CPChain中每个正常区块就相隔10秒钟。我们的验证节点在设计时钟的时候，要考虑一下几个方面

1. 出块节点的延时；
2. 验证节点共识本身所需的时间；
3. 不同的验证节点之间的时钟差；
4. 验证节点本身掉线而并非出块节点网络故障的可能性。

以CPChain这样相隔10秒钟的区块链为例，我们必然不能将弹劾块的超时时钟设置在10秒钟。因为假设验证节点在第9秒钟收到块，由于这还没有超时，验证节点必须正常验证。但是验证过程本身是消耗计算与带宽的，一旦超过1秒，总的时间就超过了10秒，也就打破了10秒钟一个正常块本身的规定。

为此我们设计一个更合理的时钟。5秒就是一个不错的选项，即一个区块相隔时间的一半。一旦某个验证节点上一个块出块之后五秒钟内没有收到新的块，它就认为这个出块节点故障，从而进入到弹劾流程。那么这个验证节点这个时候是不是就应该广播弹劾块了呢？答案是否。这是因为考虑上上面提出的第四点，即这个验证节点可能是自己掉线。虽然这个验证节点自己已经进入到弹劾阶段，但是如果其他的验证节点数量足够，已经验证通过了这个区块，那么这个验证节点也应当验证通过这个区块以保证所有节点上的一致性，即共识。从上一块验证通过开始计时，正确的做法如下：

1. 5秒的时刻，时钟表示超时，进入到弹劾阶段
2. 5到10秒钟内，不再接受出块节点的消息，但是能够接受其他验证节点验证通过的区块
3. 10秒钟之后，开始广播弹劾块
4. 如何设计弹劾块内部结构

弹劾块本身即一个区块链中，一个默认的区块结构。当出块节点发生故障，我们需要用一个默认的区块来填补这个空缺。对于任何一种区域链，每一个合法区块都包含以下几个成分：

1. 时间戳，即这个区块产生的时间
2. 出块节点的签名
3. 验证节点的签名
4. 交易信息

其中时间戳、出块节点签名与交易信息是由出块节点填写，验证节点签名是由验证节点填写。而弹劾块由于出块节点故障，以上4种信息将全部都由验证节点填写，填写方法如下：

1. 时间戳：上一个块的时间戳+两个块之间的间隔+时钟超时的阈值
2. 出块节点签名：空
3. 验证节点签名：自己的签名
4. 交易信息：空，或者对出块节点进行惩罚

其中时间戳这项，以CPChain为例，即在上一个块的时间戳上增加10+10=20秒的时间。可见一个弹劾块的耗时是超过一个普通块的。

1. 弹劾块如何共识

在传统的PBFT或者其他拜占庭容错算法中，一个消息得到验证需要满足一个“法定人数”（quorum），这个数字一般被设定为2f+1。其中f表示能够允许的最大作恶验证节点，而3f+1是验证节点的总数。也就是说，当约有三分之二的验证节点验证同意这个块，我们就能说这个块被验证通过了。但是对于弹劾块而言，这个数字是太大的。因为传统的PBFT算法中的前提假设是允许最多f个验证节点作恶的同时，允许f个验证节点掉线。如果我们考虑最坏情况，即f个忠诚（即不作恶）验证节点掉线的同时，另外f个验证节点作恶坚决不弹劾，那么剩下的只有f+1个验证节点能够正常的进行弹劾工作。所以我们只需要f+1个验证节点同意弹劾，我们就通过这个弹劾块。这个数量f+1在PBFT算法中也被称为“弱法定人数”（weak quorum）。相对应的，2f+1数量的也被称作“强法定人数”（strong quorum）。

同时f+1个验证节点也能够保证弹劾机制不会被作恶节点滥用。即使所有f个作恶的验证节点全部选择弹劾，但是如果没有一个忠诚的验证节点同意弹劾，那么将不足以凑足f+1的数量。

1. 如何利用弹劾机制设计故障重启

如上文所述，当所有的验证节点全部宕机的时候，我们可以通过弹劾机制完成对区块链的故障重启。首先我们需要一些前提假设：

1. 所有的2f+1个忠诚的验证节点的时钟相差不超过T
2. 验证节点之间的最大延时不超过T/2
3. 所有的验证节点之间会在相差不超过T/2的时间内重启

我们可以知道，必然存在一个时间T满足以上所有的三个条件。而且T必然是一个不大的数字，因为所以的验证节点在宕机之前都是保持高密度通讯并实时联网校正时钟的。所有的验证节点在重启之后，根据自己的本地时间t，创立两个弹劾块，其中一个时间戳ts1=(t/(2\*T)+1)\*2\*T，另一个时间戳ts2=ts1+2\*T。注意一点，这里的=为编程上的整型赋值运算，t与T均为整型，除法运算会舍去所有的小数部分。可以看出，由于整型除法运算会舍去小数部分，每个验证节点能够算出的ts1与ts2不再是任意可能的整数，而是集合 𝑇𝑆={𝑡|𝑡=2𝑘×𝑇, 𝑘为自然数}中的元素。当验证节点还并没有达成的共识的时候，它会在ts1时间广播其中一个弹劾块，在ts2广播另外一个。

由于上述的三个条件，所有忠诚的验证节点之中，只会产生出三种可能的时间戳, 即：

TS1=(tmin/(2\*T)+1)\*2\*T，TS2=TS1+2\*T，TS3=TS1+4\*T

这里的tmin表示所有的验证节点中时间戳数字最小的值。那么必然，整个区块链将在会TS1,TS2或者TS3达成一个弹劾块共识。一旦某个弹劾块的共识达成之后，整条链也就正常重启了，将能接受正常的块。

关于T的取值，一般而言1分钟已经非常足够了。CPChain就采用了1分钟。也就是说，假设CPChain的所有验证节点宕机，那么CPChain也能在4分钟之内完成故障恢复并重启。

**例如：**

**1、如果新技术方案是在客户端或服务器上运行的，请详细地描述下该新技术方案包括的所有步骤以及各步骤的详细情况，以描述流程图及相关的设计说明为主（有流程图的请将流程图附上）。**

**2、如果新技术方案是通过服务器和客户端之间的交互实现的，除了详细描述各步骤之外，还要说明哪些步骤是在服务器上执行的，哪些是在客户端上执行的。**

**3、如果新技术方案涉及的是多个服务器分布，应当给出这些服务器分布的拓扑图，同时结合拓扑图描述不同服务器之间的交互细节。**

**其中，对于涉及代码实现的新技术方案，也可以先从软件架构层面把代码分类为各个模块，并给出模块示意图；再结合模块示意图，用自然语言描述每个模块自身的输入、输出和处理过程，以及模块之间的交互过程**）

**5、新技术方案的亮点是什么**（提示：可以从技术的角度说新技术方案中的哪个做法是亮点，或者，也可以从功能的角度说新技术方案所实现的什么功能是亮点，两者取其中一种。请将所有亮点用1、2、3…的形式列出，简单列出即可，不需要像第4点那样详细地展开，以提醒代理人注意，便于双方校正各自对关键点的理解。）

1．设计了弹劾机制以及规定了弹劾块的结构，并设计了超时的时钟。

2. 设计了弹劾块在验证节点中的共识机制。证明了当有超过三分之一的验证节点的时钟超时，进入弹劾阶段，弹劾块就将被验证通过。

3. 利用弹劾机制设计了当整个区块链所有的验证节点宕机之后的故障重启过程。在最多数分钟之内，重启的验证节点就能够达成对一个弹劾块的共识，从而重启整条链。

**6、其他有助于专利代理人理解我们这个新技术方案的参考资料**（提示：可以以部门新员工需要了解一个技术时所要阅读的文献列表这样的标准来提供参考资料）

1. <https://docs.cpchain.io/detailed_algorithms/consensus.html#impeachment>
2. <https://docs.cpchain.io/detailed_algorithms/implementation.html#failback>

（以上资料只有英文版）

1. [↑](#footnote-ref-1)