Casper the Friendly Ghost

A “Correct-by-Construction” Blockchain Consensus Protocol

摘要

我们介绍了一个基于区块链的共识算法协议（Casper the Friendly Ghost）的说明和有限的实验观察结果。

该协议使用了 Y. Sompolinsky 和 A. Zohar 的 Greedy Heaviest Observed Sub-tree (GHOST) [1]的一个改编作为“分支选择规则”。它可以终结 / 决定具有异步和拜占庭容错共识安全性的区块。它允许区块在具有比特币区块链网络开销的网络操作每个接收O(1)消息/区块的节点时被终结。这与拜占庭容错状态机复制传统上所需的 O(N)形成了对比。

出于教学意义的原因，我们首先说明一个二进制共识协议（决定一个比特，0或1）。这个二进制共识协议满足了和区块链共识协议一样的共识安全性证明，因此它与区块链共识协议非常相似。

1. 介绍

共识协议被分布式系统中的节点用来决定相同的共识值，或者一个复制状态机的相同的输出列表。由于网络延迟和故障节点的存在使得它成为一个具有挑战性的问题。比如说，任意的网络延迟意味着当节点每次接收的消息可能以不同顺序到达时，它们会收到不同的消息集合。故障节点会离线，或者产生任意的行为。

粗略地讲，今天有两大类共识协议被人们熟知。一个我们称其为“传统共识”。这类共识“源自于（genetic roots）”Paxos和milti-Paxos以及80年代和90年代对“传统”共识协议的研究[2]。另一种，我们称其为“区块链共识”。这些共识是源自于在比特币区块链及中本聪白皮书[3]中的协议。然后我们给出了在这个文献中提出的协议满足的安全性证明的总述，最后我们介绍了这个已有的协议的说明。

1.1. 传统共识和区块链共识的比较

众所周知，传统共识协议（例如multi-Paxos以及pbft）难以理解[4]。另一方面，区块链共识协议就容易理解的多。这种区别至少一部分可以从相对来说更简洁的区块链说明中就可以看出来。

在状态机复制的内容中，传统协议（毫不犹豫地）决定了状态转变 / 交易的“区块”一次性加到共享操作日志。为了决定一个区块，一个节点必须接收O(N)的消息，其中N是形成共识的节点的数量。

比如像比特币这样的区块链共识协议不会一次终结/决定一个区块。实际上，尤其是比特币区块链根本不会做出“终结决定”；如果/当区块不处在最高总难度链（highest total difficulty chain）中的时候，区块会成为“孤儿”。然而，如果矿工可以在同一个区块链上挖矿，那么在区块链上达到足够深的区块就不会被转变（成为“孤儿”）。所以一个区块在区块链中的深度充当着终结的代理角色。在区块链共识协议的平均情况下，每个节点对每个区块只需要大约一个消息，O(1)。

传统共识协议的研究主要关注产生异步安全的（即区块不会因为未来事件的任意定时而被恢复原状态）并且在异步（或局部同步）环境中具有活性的（即节点最终决定新区块）协议。另一方面，比特币区块链在一个异步的网络中是不安全的但是在一个“局部同步网络”中是安全的并具有活性的（对于未知区块深度或者“确认计数”来说）。

传统的拜占庭容错共识协议已经精确地陈述了拜占庭容错的数量（一般可以容忍少于三分之一的拜占庭故障，或者在有3t + 1个节点时最多容忍t个故障）[CITE]。另一方面，比特币区块链协议可以容忍的错误数量（以哈希率的比例来衡量）还不太清楚。

1.2. 现阶段工作综述

我们给出了一个共识协议“Casper the Friendly Ghost”的说明，它既有区块链共识协议的低开销，又有通常与传统共识协议相关的异步拜占庭容错安全性。然而，在我们分享这个区块链共识协议之前，我们先给出一个二进制共识协议的说明（选择在0和1之间具有异步的拜占庭容错安全性）。

了解二进制共识协议可以更容易地理解区块链共识协议；这些协议非常相似。他们如此相似是因为他们都是为了满足相同的安全性证明而“被产生的”。

这个选择协议说明的过程在这里没有被指定或证明，但是可以在另一个更抽象的论文[CITE]中找到。本文缺少关于为什么确切地做出某些选择或者如何确切地证明某些声明的信息。即使没有全范围的“process paper”，我们的目标还是向读者提供关于区块链协议为什么/如何工作的清晰的直观说明。因此，我们给出这些协议满足的安全性证明的高层级的草图。然后，我们介绍之前承诺的协议。

1.3. 共识安全性证明

这里提出的每个协议都符合相同的安全性证明。（并且事实上，任何通过correct-by-construction过程[1]产生的共识协议都可以满足相同的证明。）

[1] See our related paper – PAPER

我们假设运行共识协议的节点具有（本地）状态Σ。这些状态在其之间具有称为“协议执行”或“协议状态转换”的定向路径。如果有从σ到σ’的执行，我们就写作σ→σ’。另外，路径是可组合的：如果有σ→σ’和σ’→ σ”，则还有一个执行σ→σ”。

证明设计一个“估计器”，它将协议状态映射到关于共识的提议上。在二进制共识中，估计器将协议状态映射到0或1。另一方面，在区块链共识中，估计器将协议状态映射到一个区块链（用作我们的“分支选择”）。

如果一个在二进制共识（0或1）中的估计值在所有未来协议状态[2]上被估计器返回，则对于一个特定协议状态，它被认为是“安全的”（具有“估计安全性”）。在区块链共识中，如果一个区块对于所有未来的协议状态也处于分叉选择中，则对于一个特定的协议状态，这个区块被认为是“安全的”。

共识安全性证明表明，安全估计的决定具有共识安全性[3]（只要拜占庭式故障不超过t）。

证明依赖于以下关键结果：如果状态为σ_1的节点1的具有安全估计值e_1，而另一个状态为σ_2的节点2具有安全估计值e_2，并且如果它们具有共同的未来状态σ_3，则节点1和节点2对于e_1和e_2的决定是一致的。结果是相当简单的，因为它遵循估计安全性的定义而没做太多工作。具体而言，如果一个状态σ有一个安全估计值e，那么σ的任何未来协议状态σ’在e上也是安全的。所以如果我们的状态σ_1和σ_2（在e_1和e_2上有安全性）共享一个共同的未来状态，那么在e_1和e_2上这个未来状态一定是安全的，这意味着它们是一致的。因此，证明的第一部分显示，对于具有一个共同未来协议状态的任何节点对，安全估计的决定都是共识安全的。

接下来，我们旨在构建保证除非在有超过t个拜占庭故障的情况下，节点都具有共同未来协议状态的协议（具有“状态转换”的“协议状态”）。从我们刚讨论的结果看，如果有不超过t个这样的故障，这样一个协议就具有共识安全性。我们在几步内达成这点。

首先，我们假设协议状态是协议消息集，然后坚持任何两个协议状态σ_1和σ_2的并集$\sigma_1 \cup \sigma_2$本身就是一个协议状态。然后，我们坚持认为，从协议状态σ到σ’⊃σ（σ的任何超集）都存在一个状态转换。这意味着$\sigma_1 \cup \sigma_2$是σ_1和σ_2的一个协议未来状态。

这个假设本身允许任何两个状态总是有一个共同的未来状态，这本身就保证了决定安全估计的共识安全性。但是有一个问题：这个协议不能满足共识的non-triviality属性。Non-triviality属性意味着协议能够在相互排斥的值之间进行选择。在我们的文章中，non-triviality属性意味着有两个协议状态σ_1和σ_2，它们在两个互斥的估计上都是安全的。具有互斥安全估计的两个协议状态不可能有一个共同的未来状态，但我们只是坚持$\sigma_1 \cup$

[2] I.e. all states accessible from that state through any valid protocol execution.
[3] Consensus safe decisions have the following property: any decisions made on safe estimates by a protocol following node will be *consistent* with decisions made on safe estimates by any other protocol following node.

σ_2会是一个这样的共同未来状态。如上所述，这个矛盾意味着我们还没有满足non-triviality属性。

相反，只要在$\sigma_1 \cup \sigma_2$中有少于t的拜占庭故障，我们就必须确定σ_1和σ_2具有共同的未来状态。这允许没有共享协议未来（允许non-triviality属性）的状态，但是仍然允许从我们以前的结果得到的共识安全性（尽管现在只是只要少于t个拜占庭故障）。

所以我们的下一步是提出一个在任何给定的协议状态下都可以证明的计算“拜占庭故障数量”的过程。在最后一步中，我们通过使用上一步骤的过程，通过排除超过t个这样的故障的状态来定义我们的初始协议的“新版本”。

实际上，这里指定的两个协议都具有这样的属性,当且仅当有不超过t个拜占庭故障时，任何协议状态对（σ_1和σ_2的并集$\sigma_1 \cup \sigma_2$）也是协议状态。因此，如果拜占庭故障少于t个，那么决定安全估计的两个节点就具有共识安全性，而这一切需要估计器做的，几乎为零！:)

2. Casper the Friendly Binary Consensus

我们现在通过先定义协议消息，然后定义估计器（将协议消息集映射为0或1）来说明二进制共识协议。我们提出了一种检测和计数拜占庭故障的方法，并随后将Casper the Friendly二进制共识的“协议状态”定义为表现出最多有t个拜占庭故障的消息集。然后我们定义协议的“状态转换”，最后定义二进制估计安全性。这些定义符合那些决定安全估计的共识安全性证明。

“协议消息”的定义是在一组“验证者名称”V中参数化的，它们被识别为共识形成节点的名称。

协议消息有三个部分。一个“估计estimate”（一个0或1），一个“发送者sender”（验证者名称validator name）和一个“证明justification”。证明本身就是一组协议消息。验证者使用这些协议消息来更新互相的当前估计值。此外，估计值不是任意的，因为只有当“估计”是对消息的“证明”应用估计器的结果时，协议消息才是“有效的”。

估计器和有效性的定义稍后给出。现在，我们把二进制共识协议中所有可能的协议消息集表示为M，并将其定义如下：

Definition 2.1 (Protocol Messages, $\mathcal{M}$).

$$\mathcal{M}_0 = \{0,1\} \times V \times \{\emptyset\}$$

$$\mathcal{M}_n = \{0,1\} \times V \times \mathcal{P}(\bigcup_{i=0}^{n-1} \mathcal{M}_i)$$

$$\mathcal{M} = \lim_{n\to\infty} \bigcup_{i=0}^{n} \mathcal{M}_i$$

M_0是“基本情况”，即具有“无效证明”的消息集。M_n是“高度”n处的消息集，在器证明中具有高度n-1（和/或更低）的消息。注意，消息M_0具有高度0。P表示“power set”函数，其将集合映射到其所有子集的集合，因此 $P(U_{i=0}^{n-1} M_i)$ 表示在高度n-1或更低处的所有协议消息集合。

估计器是一个将协议消息集映射为0或1的函数，或者是一个用∅表示的具有$\mathcal{E}(\emptyset) = \emptyset$属性的空值：

$$\mathcal{E} : \mathcal{P}(\mathcal{M}) \to \{0, 1\} \cup \{\emptyset\}$$

如果$\mathcal{E}$(J(m)) = ∅或E(m) = $\mathcal{E}$(J(m))，则一个协议消息m就被认为是“有效的”。从现在开始，我们假设M只包含有效的消息。[4]

但是在定义估计量之前，我们需要一些更基本的定义。我们定义E，它是一个“辅助函数”，用来挑选协议消息中给出的“估计”：

$$E(m) = e \iff m = (e, \_, \_)$$

类似地，我们将V定义为一个函数，用来选择“发送者”，J定义为选择“证明”的函数。

我们说消息m_1是消息m_2的“一个依赖dependency”，如果m_1在m_2的证明中，或者如果m_1在m_2的证明中的一个消息的证明中，或者如果m_1在m_2的证明的一个消息的证明的一个消息的证明……，我们写作$m_1 \prec m_2$。

如果$m_1 = m_2$，我们也称m_1是m_2的一个依赖：

Definition 2.2 (dependency, $\prec$).

$$m_1 \prec m_2 \iff m_1 = m_2 \text{ or } \exists m' \in J(m_2) \,.\, m_1 \prec m'$$

我们将消息m的“依赖”定义为所有使得m’ ≺ m的消息m’。这些全部都是可以在证明中获得的消息，或者在证明中的消息的证明……以此类推。

$$D(m) = \{m\} \cup \bigcup_{m' \in J(m)} D(m')$$

这个定义可以以一种自然的方式进行扩展，以定义一组消息的依赖（通过取各个消息的依赖的并集）。

如果$m_1 \prec m_2$，那么我们也可以说m_2比m_1“更晚”，写作$m_2 \succ m_1$。

我们现在有一种语言来讨论来自一个发送者v，从消息集M中得到的最新（latest）消息，我们表示为L(v, M)：

---

[4] All protocol messages at height 0 are valid because they have null justifications and $\mathcal{E}(\varnothing) = \varnothing$. Then we can find the valid protocol messages at height 1 by applying the estimator to their justifications (which are sets of [valid] messages at height 0) for each message, only keeping valid messages. Similarly, we can find the valid protocol messages at height *n* by applying the estimator to the justification of these messages (which are sets of valid messages at height $h < n$). We are thereby able to collect all valid protocol messages and restrict *M:*

Definition 2.3 (Latest message).

$$m \in L(v, M) \iff \nexists m' \in D(M) \text{ such that } V(m') = v \text{ and } m' \succ m$$

最新消息（latest messages）对于定义估计器是至关重要的，如果“更多”节点具有估计值为0而不是1的最新消息，则返回0。我们使用节点的“权重”来衡量哪个估计具有“更多”共识形成节点，通过映射验证器名称到正实数来实现。我们这样做不失一般性（因为可能所有的权重是相等的）。

$$W : V \to \mathbb{R}_+$$

现在我们将一组消息M中的估计值e的“分数”定义为具有估计e的最新消息的验证器的总权重。

Definition 2.4 (Score of a binary estimate).

$$\text{Score}(e, M) = \sum_{\substack{v \in V \\ \text{such that } m \in L(v,M) \\ \text{with } E(m)=e}} W(v) \tag{1}$$

最后，我们定义二进制共识的估计器，如果有的话返回最高分数的估计，否则返回∅。

Definition 2.5 (Binary estimator).

$$\mathcal{E}(M) = 0 \quad \text{if Score}(0, M) > \text{Score}(1, M), \tag{2}$$
$$\mathcal{E}(M) = 1 \quad \text{if Score}(1, M) > \text{Score}(0, M), \tag{3}$$
$$\mathcal{E}(M) = \emptyset \quad \text{if Score}(1, M) = \text{Score}(0, M) \tag{4}$$

在这个阶段，我们有协议消息和一个估计器。如果我们可以从一组协议消息中定义一个计算拜占庭故障的方法，那么我们可以给出一组协议状态以及它们的状态转换，用于容忍t个拜占庭故障的二进制共识协议。

每个协议消息m应该表示验证者V(m)看到的消息的记录。任何“正确的”节点都有越来越多的收到和发送的消息记录。具体来说，一个关联节点从来就不是一对消息m_1和m_2的发送者，使得既没有$m_1 \prec m_2$也没有$m_1 \succ m_2$。我们把这样一对消息称为“一个equivocation”。

Definition 2.6 (Equivocation).

$$Eq(m_1, m_2) \iff V(m_1) = V(m_2) \text{ and } m_1 \not\succ m_2 \text{ and } m_1 \not\prec m_2 \tag{5}$$

在一个协议消息集M中的具有equivocation的发送者v，被称为“拜占庭”，或者在M中表现为一个“拜占庭故障”。

Definition 2.7 (Byzantine faulty node).

$$B(v, M) \iff \exists m_1, m_2 \in D(M) \text{ such that } v = V(m_1) \wedge Eq(m_1, m_2) \tag{6}$$

然后我们定义在M中可见的拜占庭节点为

$$B(M) = \{v \in V : B(v, M)\}$$

在一组消息M中证明的拜占庭式故障的权重是M中拜占庭验证者的权重总和。

Definition 2.8 (Fault weight).

$$F(M) = \sum_{v \in B(M)} W(v) \tag{7}$$

我们现在定义我们的协议状态$\Sigma_t \subset P(M)$为表现出少于t个故障的协议消息的集合（按权重）：

Definition 2.9 (Protocol States).

$$\Sigma_t = \{\sigma \subseteq \mathcal{M} : F(\sigma) \leq t\} \tag{8}$$

而且我们也将“协议执行”定义为Σ_t中的状态之间的有向路径，从而使得当且仅当$\sigma \subseteq \sigma'$时，从$\sigma \in \Sigma_t$到$\sigma' \in \Sigma_t$的执行存在。我们把存在这样的状态转换写作$\sigma \rightarrow \sigma'$。

Definition 2.10 (Protocol Executions).

$$\forall \sigma, \sigma' \in \Sigma_t \quad : \sigma \rightarrow \sigma' \iff \sigma \subseteq \sigma' \tag{9}$$

协议执行构成$(\sigma \rightarrow \sigma' \wedge \ \sigma' \rightarrow \sigma'' \Longrightarrow \sigma \rightarrow \sigma'')$，因为不正确的子集关系（⊆）组成。

我们现在可以给出二进制共识的估计安全性的定义，对于协议状态σ中的估计e。

Definition 2.11 (Binary Estimate Safety).

$$S_t(e, \sigma) \iff \forall \sigma' \in \Sigma_t : \sigma \rightarrow \sigma', \ e = \mathcal{E}(\sigma')$$

因为这个结构满足了我们共识安全性证明的条件，所以我们知道在这个协议中这样的安全估计的决定是共识安全的（如果拜占庭故障（按权重）少于t个）。

我们仍然必须讨论运行二进制共识协议的节点如何能够检测到他们处于一个具有安全估计的状态。我们按照区块链共识协议的说明讨论这个问题，因为我们用这两种协议的相同方式来检测估计安全性。

现在我们已经介绍了二进制共识协议，指定区块链共识协议要简单得多了。

3. Casper the Friendly Ghost

Casper the Friendly Ghost的规范与Casper the Friendly Binary Consensus非常相似。事实上，我们可以在不做任何修改的情况下重用大部分的定义。

我们再次首先定义所有协议消息的集合，然后定义估计器，这次是贪婪最重观察子树（GHOST）“分支选择规则”。然后，我们将协议状态定义为消息集合，它表现出高达t个拜占庭故障（按权重），然后我们定义协议的“状态转换”。最后，我们定义区块估计安全性。这些定义符合安全估计决定的共识安全性证明中的定义。

协议消息被称为“区块”，并且具有与二进制共识协议中的消息相同的三个组件：一个估计，一个发送者和一个证明。估计是一个区块，称为“prevblock”或“父区块”。对于有效的消息，估计值是由证明中GHOST分支选择规则选择的区块链头部的区块。“发送

者”（验证器名称）字段的定义与之前的完全相同。最后，“证明”又一次只是一组协议消息。

正式地，在Casper the Friendly Ghost，我们有具有以下形式的协议消息，M，（又一次在一组验证器名称V中参数化）：

Definition 3.1 (Blocks).

$$\text{Genesis Block} = \{\emptyset\}$$
$$\mathcal{M}_0 = \{\text{Genesis Block}\} \times V \times \{\text{Genesis Block}\}$$
$$\mathcal{M}_n = \bigcup_{i=0}^{n-1} \mathcal{M}_i \times V \times \mathcal{P}(\bigcup_{i=0}^{n-1} \mathcal{M}_i)$$
$$\mathcal{M} = \{\text{Genesis Block}\} \cup \lim_{n\to\infty} \bigcup_{i=0}^{n} \mathcal{M}_i$$

在前面的章节二进制共识中给出的“辅助函数”（E，V和J），“依赖”，“更晚”，“最新的消息”和“验证器权重”的定义在这里以完全相同的方式定义。因此，我们使用它们而无需再给出定义。

我们确实也需要一些新的术语来定义分支选择规则。我们写$b_1 \downarrow b_2$，并声称区块b_1在区块b_2“的区块链之中”，如果

Definition 3.2 (Blockchain membership, $b_1 \downarrow b_2$).

$$b_1 \downarrow b_2 \iff b_1 = b_2 \text{ or } b_1 \downarrow E(b_2) \tag{10}$$

我们定义给定一组协议消息M的区块b的“分数”作为具有最新区块b’的验证器的总权重，使得b↓b’。

Definition 3.3 (Score of a block).

$$\text{Score}(b, M) = \sum_{\substack{v\in V \\ m\in L(v,M) \\ b\downarrow E(m)}} W(v) \tag{11}$$

一组协议消息M中的区块b的“子区块”是以b为prevblock的区块。

$$C(b, M) = \{b' \in M : E(b') = b\}$$

我们现在有语言需要定义区块链共识的估计器，即GHOST规则！

Definition 3.4 (The Greedy Heaviest-Observed Sub-Tree (GHOST) fork-choice rule, $\mathcal{E}$).

Data: A set of blocks, M
Result: The block at the head of the fork choice
b = Genesis Block
while *b has children ($C(b, M)$ is nonempty)* **do**
  scores = dict()
  for *each child of block b, $b' \in C(b, M)$* **do**
    scores[b'] = Score(b', M)
  end
  if *scores has a unique maximum* **then**
    b = argmax(scores)
  else
    b = the max score block with the lowest hash
  end
end
return b

Algorithm 1: The Greedy Heaviest-Observed Sub-tree Fork-choice rule, $\mathcal{E}$

我们假设“哈希hash”具有这样的属性，在任何区块集合中，只有一个是具有最低哈希值的。使用区块的散列消除“关系tie”意味着区块链共识的估计器永远不会输出异常。当0和1有相同的分数时，二进制估计器以前返回∅。这意味着如果E(m) = $\mathcal{E}$(J(m))，则消息m是有效的，正如在二进制共识中我们坚持认为所有的区块都是有效的一样。[5]

这里定义的“Equivocation”，“拜占庭故障”，“故障权重”，“协议状态”和“协议执行”与二进制共识完全相同。因此，我们不再给出定义。

我们现在可以给出Casper the Friendly Ghost的估计安全性的定义，对于协议状态σ中的区块b。

Definition 3.5 (Blockchain Estimate Safety).

$$S_t(b, \sigma) \iff \forall \sigma' \in \Sigma_t : \sigma \to \sigma',\ b \downarrow \mathcal{E}(\sigma')$$

因为这个结构满足了我们共识安全性证明的条件，所以我们知道在这个协议中安全估计的决定是共识安全的（如果拜占庭故障少于t（按权重））。

我们现在提出一个机制，节点可以使用它来检测它们的二进制和区块链共识中哪个估计是安全的。

4. 一个简单的安全性预言（A Simple Safety Oracle）

估计安全性在一些协议状态σ中是参数化的，并且是其所有未来协议状态的属性。估计安全性的挑战是（至少在我们的情况下）有可能的未来协议状态的数量是无限的，但是节点在某种程度上需要确定一个估计是否在一个有限的（理想的非常短的）时间上是安全的。

估计安全性在这些在未来状态上的估计器$\mathcal{E}$中某种程度是不变量的估计值上也是参数化的。我们把这个安全性已经确定的估计称为“候选估计”。

[5] Following the process decribed in the footnote about excluding invalid messages from the binary consensus.

当且仅当在Σ_t中存在这种协议状态时，一个“理想的对手”返回一个“攻击”，一个使得$\sigma \to \sigma'$的未来协议状态σ'，和一个与候选估计“不一致”的估计器$\mathcal{E}(\sigma')$。我们可以想象，这样一个对手将是他们搜索未来可能的状态的策略。一个“理想的安全性预言”只要理想的敌人没有发现攻击，就代表具有安全性。

如果我们只能检测到理想对手未能发现攻击的一些情况，那么我们就可以构建一个安全预言器，它能够检测带有估计安全性的状态的安全性。出于效率方面的考虑，这些情况足以使其在实现中非常有用，即使它可能不是理想的安全预言。

我们确定了这样一组情况。为了更一般地讨论，我们把估计e和估计e'之间的“一致”表示为$e \equiv e'$。这对应于二进制共识中的$e = e'$，以及区块链共识中的$e \downarrow e'$。不一致将用$\not\equiv$表示。

我们说一个验证器v_i“看到验证器v_j与在一组协议消息M中的估计e一致”，如果：

- v_i在M中只有一个最新消息（我们将这个消息表示为$L(v_i, M)$）
- v_j在v_i的最新消息证明中，$J(L(v_i, M))$，只有一个最新消息（我们表示为$L(v_j, J(L(v_i, M)))$）
- 该消息的估计值与e一致，即$E(L(v_j, J(L(v_i, M)))) \equiv e$

Definition 4.1 (v_i sees v_j agreeing with e in M).

$$v_i \xrightarrow[M]{\equiv, e} v_j \iff E(L(v_j, J(L(v_i, M)))) \equiv e$$

我们说一个验证器v_i“可以看到v_j与一组协议消息M中的估计e不一致”，如果：

- v_i在M, L(vi, M)中只有一条最新的消息
- v_j在v_i的最新消息的证明中，$J(L(v_i, M))$，只有一个最新消息（我们表示为$L(v_j, J(L(v_i, M)))$）
- v_j有一个“v_i的新最新消息”$m \in M$，使得$m \succ L(v_j, J(L(v_i, M)))$
- 这个m与e不一致，$E(m) \not\equiv e$

Definition 4.2 (v_i can see v_j disagreeing with e in M).

$$v_i \xRightarrow[M]{\not\equiv, e} v_j \iff \exists m \in M : V(m) = v_j \wedge m \succ L(v_j, J(L(v_i, M))) \wedge E(m) \not\equiv e$$

M中的一组互相可以看到在M中与e一致并且互相看不到在M中与e不一致的非拜占庭节点，被称为M中的“e-clique”。

我们的目的是证明如果在M中有一个总权重为W'且$2 * W' > \Sigma_{v \in V} W(v)$的e-clique，那么对于在$t = 0$具有状态$\Sigma_t$的协议，e是安全的。我们不会给出严密的证明，但这个想法很简单。如果一个e-clique中的节点互相看到对方与e一致，并且看不到彼此与e不一致，那么clique内部就不会存在任何会导致他们给e分配更低的分数的新消息。此外，如果clique按权重拥有一半以上的验证器，那么clique外部的任何信息都不会将这些验证器分配给

竞争估计的分数提高到高于他们分配给e的分数。因此，来自clique内外的发送者的消息不能改变clique内部发送者的估计值，理想的对手必然会失败。

我们还可以证明，如果M中的e-clique的总权重W'并且$2 * W' > \Sigma_{v \in V} W(v)$，则e是安全的，只要$2 * W' - \Sigma_{v \in V} W(v) \leq t - F(M)$，但是我们将这个工作留给以后来继续做。

我们提出了使用这个结果来检测估计安全性的"clique oracle"的伪代码：

Data: An estimate e, a set of messages $M \in \Sigma_t$
Result: True or False, an indicator when e is safe in $M \in \Sigma_t$
N = a fully connected network with undirected edges and validator names V as nodes
for *each non-Byzantine validator,* $v_i \in V$ **do**
  for *each non-Byzantine validator,* $v_j \in V$ **do**
    if v_i *doesn't see* v_j *agreeing with* e *in* M *(not* $v_i \xrightarrow[M]{\equiv, e} v_j$*)* **then**
      N.remove_edge((v_i, v_j))
    end
    if v_i *can see* v_j *disagreeing with* e *in* M *(*$v_i \xRightarrow[M]{\not\equiv, e} v_j$*)* **then**
      N.remove_edge((v_i, v_j))
    end
  end
end
N' = N.maximum_weight_fully_connected_subcomponent(weights=W)
clique_weight = $\sum_{v \in N'.nodes} W(v)$
total_weight = $\sum_{v \in V} W(v)$
if *2*clique_weight* $\leq$ *total_weight* **then**
  return *False*
end
if *2*clique_weight* $>$ *total_weight* **then**
  initial_fault_weight = $F(M)$ (this will be $\leq t$ because $M \in \Sigma_t$)
  fault_tolerance = 2*clique_weight - total_weight
  if *fault_tolerance + initial_fault_weight* $> t$ **then**
    return *True*
  else
    return *False*
  end
end

Algorithm 2: The "Clique Oracle", S_t

现在我们有一个安全预言的实现，我们可以给Casper the Friendly Binary Consensus和Casper the Friendly Ghost装备有检测估计安全性能力的节点。这允许节点实际上而不是仅仅假设地在安全估计上做出（共识安全的）决定。

5. 主观容错阈值

可以调整共识安全性证明，以讨论在不同的容错级别上运行任何这些"correct-by-construction"共识协议的节点对安全估计的共识安全性。

一个节点可以在Σ_{t1}中执行协议，而另一个节点可以在Σ_{t2}中执行。只要他们的状态的并集展示出少于t_1和t_2故障（按权重）的最小值，他们的决定（当然只有在安全估计上才会发生）就具有共识安全性。

这是因为在这些不同协议中的协议状态是相同的，除了其中一个也包含比另一个有更多拜占庭故障的协议状态。例如，如果$t_1 < t_2$，则$\Sigma_{t1} \subset \Sigma_{t2}$。这意味着，只要不处于使得$F(\sigma_1 \cup \sigma_2) > t_1$的状态$\sigma_2$，$\sigma_2 \in \Sigma_{t2}$处的节点与$\sigma_1 \in \Sigma_{t1}$处的节点就共享一个共同的未来状态，。

从协议中删除容错阈值并将其设置为每个节点可独立配置的设置具有许多有吸引人的属性。举一个例子，攻击者可能不知道其目标的容错阈值。它也有经济效益，但这些都超出了本文的范围。[6]

6. Casper the Friendly Ghost with Validation Rotation

我们可以定义一个修改版本的Casper the Friendly Ghost，它允许动态改变共识形成节点（和/或它们的权重）的集合。协议规范正常工作，所以我们在细节上可以更轻松。

区块结构被修改，以便证明也可以包括权重映射。

Definition 6.1 (Blocks).

$$\text{Genesis Block} = \{\emptyset\} \times \{\emptyset\} \times (\{\emptyset\} \times \{W_{genesis}\})$$
$$\mathcal{M}_0 = \{\text{Genesis Block}\} \times V \times (\{\text{Genesis Block}\} \times \mathcal{W})$$
$$\mathcal{M}_n = \bigcup_{i=0}^{n-1} \mathcal{M}_i \times V \times (\mathcal{P}(\bigcup_{i=0}^{n-1} \mathcal{M}_i) \times \mathcal{W}))$$
$$\mathcal{M} = \{\text{Genesis Block}\} \cup \lim_{n\to\infty} \bigcup_{i=0}^{n} \mathcal{M}_i$$

我们使用W来表示所有权重映射的集合 $W : V \to \mathbb{R}_+$

我们重新定义一个区块的“分数”来使用父区块中的权重。记住，这个分数习惯于由GHOST分支选择规则在某个区块的子区块之间选择。那些子区块（自然地）有同样的父区块，因此它们也会有来自同一验证者的最新消息所决定的分数。

Definition 6.2 (Score of a block, for Casper the Friendly Ghost with validator rotation).

$$\text{Score}(b, M) = \sum_{\substack{v \in V \\ m \in L(v,M) \\ b \downarrow E(m)}} J(E(b)).W(v) \tag{12}$$

使用“J(m).W”从m的证明中选取权重。

[6] Briefly, removing in-protocol fault tolerance thresholds allows every node to marginally contribute to consensus (since more fault tolerance may allow more nodes to find safety) and therefore have a positive contribution to any cartel (and thus it would not be in the cartel's interest to censor them). It also removes focal points for cartelization.

协议中其余的定义保持完全相同（包括估计器，协议状态/转换和安全性预言），因此不再重述。因此，我们已经说明了包含验证轮换的Casper the Friendly Ghost，一个容忍多至t个拜占庭故障的共识协议。[7]

7. 活性的考虑

传统上，我们希望证明协议“具有活性”，即运行共识协议的节点最终决定一个值。对于我们来说，相应的保证是节点最终检测到他们有一个安全估计。

FLP不可能性表明，也就是说，像我们这样的共识协议（不使用非确定性或加密技术）是不可能在“异步网络”中，在缺少关于消息传播时间的假设（或者消息到达的顺序）[CITE]情况下，具有活性的。

我们还没有任何形式的同步假设，这就是为什么这里给出的共识协议是异步安全的。而且，我们还没有谈到节点应该何时发送协议消息。相反，我们对节点可以采取的协议执行施加了相对宽松的约束（只要σ’没有证明有太多的拜占庭故障，他们就可以接收消息并从σ移动到任何状态σ’⊃σ）。

这种验证器应该如何制造区块的特定策略的缺失意味着我们目前无法提供活性的证明。但是，我们现在看到一些实验性的观察，其中估计安全性在下一节中完成。这些协议执行对应于具有活性的消息的到达顺序，因此，只要节点可以协调超时以便（最终）产生期望的协议消息的“形状”，我们就可以构建用于实现活性的可靠策略。在同步或局部同步网络（即在消息到达时间存在已知或未知边界的网络）中，超时协调（至少最终）是可能的。

尽管如此，活性方面的考虑很大程度上被视为范围之外，应在未来的工作中予以处理。

8. 实验观察

Casper the Friendly Binary Consensus和Casper the Friendly Ghost的早期原型已经实现，下面的图表显示了以不同顺序传递消息的协议执行的观察结果。[8]

在所有这些图表中，每个消息都由图中的一个节点表示。来自同一个验证器的消息是垂直对齐的。后来的消息始终显示为高于其依赖关系。

---

[7] We allowed completely unrestrained changes in the validator set between any block and any of its children. This means that while we always have estimate safety unless there are t or more faults for nodes running Σ_t, we have not restricted the total sum of the weights to be constant (or even greater than t). In practice, we either restrict the sum of the weights to be constant or redefine t so that it refers to a proportion of the weight.

[8] The code is available at https://github.com/ethereum/casper-cbc

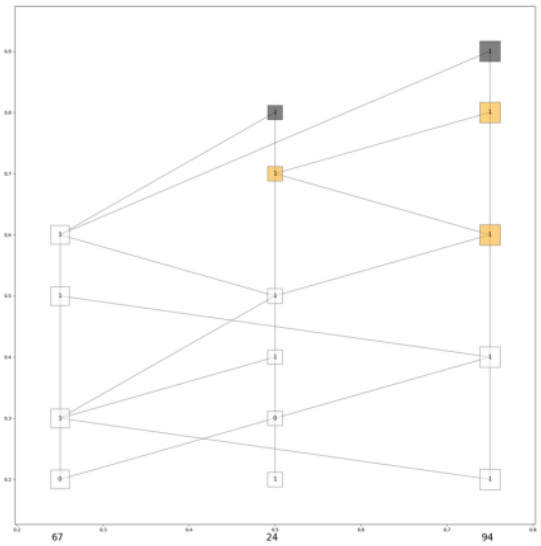

Figure 1: A Binary consensus protocol execution. Dotted lines are messages included in the justification of the later message. The label on the nodes represents the estimate of the message. A message is coloured if it has achieved some amount of Byzantine fault tolerant estimate safety, accoring to a clique oracle given its justification. The darker the colour, the more faults are tolerated by the estimate

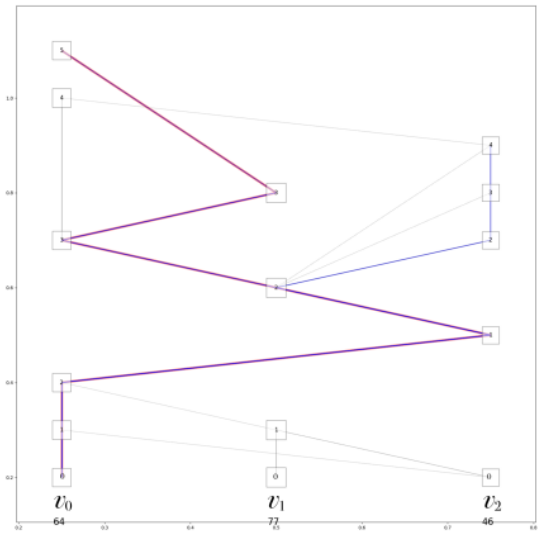

Figure 2: Blockchain protocol execution with 3 validators, v_0, v_1, v_2. Each node labeled 0 is the first message from that validator, and the nodes verically aligned above each validator represent messages made by that validator. Dotted black lines are messages included in the justification of the later message. Blue lines represent the forkchoices of the validators given by their latest blocks. Solid grey lines are prevblock pointers (that aren't blue because they are no longer the validator's forkchoice). The red line is the result of applying GHOST to the set of messages displayed here.

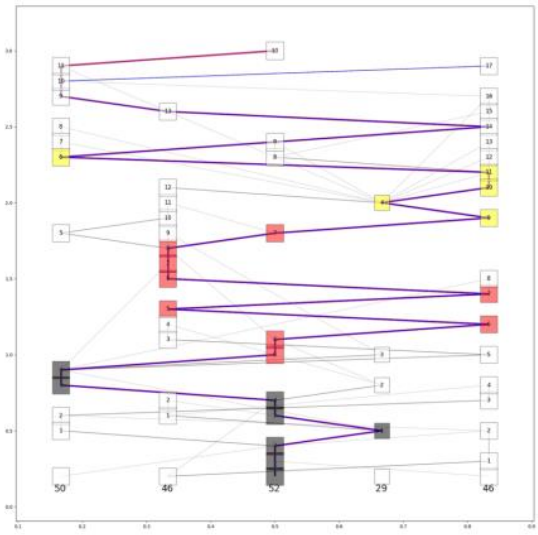

Figure 3: In this execution of the blockchain consensus protocol, we observe some safe blocks. They are colour coded the same way as in the earlier, however this time they represent the blocks that are seen to be safe from the view that includes all of the nodes. In contrast, in the binary consensus we displayed safety that was detected locally.

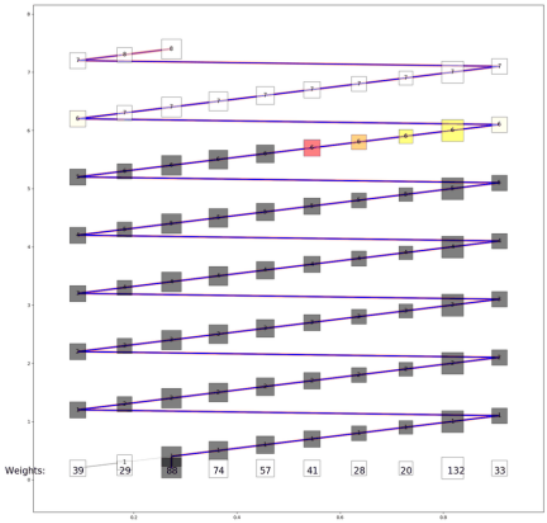

Figure 4: Finally, in this execution of the blockchain consensus protocol where the validators are able to pass blocks around in a "round-robin" configuration, we observe the $\mathcal{O}(1)$ messages received per (finalized) block. This is possible because each block contributes to the consensus safety of many blocks.

9. 结论

我们简要概述了Casper the Friendly Ghost已被推导为满足共识安全性的证明（以"correct-by-construction"的方式）。然后，我们定义了一个满足证明的二进制共识协议（Casper the Friendly Binary Consensus），从而从结果中受益。然后，我们明显地稍微修改了二进制共识协议，以给出基于区块链的共识协议（可以采用，或者甚至直接命名为顺序状态机复制）。

然后，当我们证明我们可以从协议中去掉容错阈值并且实现验证器轮换而不对协议进行重大改变时，我们进一步目睹了安全性证明的威力。最后，我们看到（正如宣称的

那样），Casper the Friendly Ghost决定具有共识安全性的，在每个区块中容忍具有O(1)消息的拜占庭故障（在异步网络中）的区块是可行的。

我们希望这项工作具有教育意义，并将引导许多有趣和有用的共识协议的发展。

10. 致谢

我要感谢Nate Rush和Danny Ryan在Devcon3上准备发布代码库的努力工作，感谢他们在这篇（草稿）论文上的帮助。感谢Karl Floersch，他是第一个开发代码库的开发人员。我想感谢Greg Meredith对协议设计的“correct-by-construction”方法的介绍，以及许多帮助形式化想法。最后，我要感谢Vitalik Buterin，三年前我就与其合作进行这项研究的初期工作。

参考文献

[1] Sompolinsky, Y. & Zohar, A. Secure high-rate transaction processing in bitcoin (2013). URL https://eprint.iacr.org/2013/881.pdf.

[2] Lamport, L. The part-time parliament. *ACM Transactions on Computer Systems* **16**, 133169 (1998).

[3] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash systems (2008). URL https://bitcoin.org/bitcoin.pdf.

[4] Paxos made moderately complex. URL http://paxos.systems/.