Ripple协议共识算法

# 概要

一些共识算法存在于拜占庭将军问题，特别是涉及到分布式支付系统，要求网络中所有节点通信同步，这种要求所带来的高延迟会造成很多问题。在这项工作里，我们提出了一种新的共识算法，通过在较大的网络中利用集体信任子网来规避这种要求。我们发现，这些要求被“信任”的子网事实上是非常小的，并且可以通过成员节点的筛选原则进一步减少。此外，我们发现，为保持整个网络一致性，最小的连接是必要的。这个结果就是低延迟的共识算法，他在面对拜占庭失败时总是保持系统的稳定。在Ripple协议的实施过程中，我们将呈现这个算法。

# 介绍

近些年来，人们在分布式共识系统投入的兴趣和研究都显著增加，特别是集中聚焦在分布式支付网络里。这些工作允许快速、低消费的交易不受中心资源的控制。这个系统所带来的经济效益和缺点值得投入更多的研究，主要是在分布式支付系统面对的一些技术挑战。虽然问题是多方面的，我们把它们划分三个主要的分类：正确性，一致性和可用性。

正确性，我们指的是，区分分布式交易系统中正确交易和欺诈交易的差别是必要的。在传统的信托设置中，这是通过机构间的信任以及使用加密签名保证了机构确实是它声称的来自何处。然而，在分布式系统中，不存在这样的信任，甚至不知道网络成员中任何一个人的身份。因此，正确性必须采用其他的方式实现。

一致性指的是在面对去中心化的记账系统中，如何维护单一的全局事实。和正确性的问题类似，不同之处在于以下事实：网络中的一个恶意用户无法创建欺诈的交易（违抗正确性），但是在某种程度上可以创建多个彼此无关的正确的交易，并把这些结合起来形成欺诈行为。例如，一个恶意用户同时发起两个购买，但是他账户的资金只能满足其中任何一个购买，不能同时两个一起满足。因此，每个交易本身是正确的。但是以这样的方式同时执行交易，对于分布式网络而言是一个整体而不知道两者之间的关系，那么一个明显的问题就出现了，也就是通常被称为“双重消费”问题。因此该一致性问题可以概括为这样的要求，既网络中只有一个全局的公认的交易存在。

可用性是略微抽象的问题，就是我们通常定义的“有用的”分布式支付系统，但是经常简化为系统的延迟问题。例如，分布式系统是正确并一致的，但是需要一年的时间处理交易，显然这个系统是不可用的。可用性的其他方面还包括参与正确性和一致性处理的算力水平以及用户在网络中避免欺诈的技术能力等。

在现代的分布式计算系统出现之前，通过已知的“拜占庭将军问题”，许多问题都经过了长期探讨。拜占庭将军问题中，有一群将军各自掌管军队一部分，必须通过信使发送信息来协调攻击。由于他们处于陌生的敌国领土，所以信使可能无法到达目的地并可能会失败（就像分布式网络节点失败，或者发送了损坏的消息代替预期的消息）。问题的另一方面是，一些将军有可能是叛徒，单独或者一起合谋。所以可能到达的信息是旨在创建一个对忠诚的将军而言注定失败的错误计划（就像分布式系统中的恶意成员试图说服系统接受欺诈交易，或者多个版本的相同的真实交易，造成一个双重支付）。从而分布式支付系统面对一般的标准错误和“拜占庭”失败必须是健壮的，可以从网络中的多个资源进行协调和建立。

在这项工作里，我们分析了分布式交易系统的一个特定实现：Ripple协议。我们专注于利用实现上述目标中正确性，一致性和可用性的算法，并证明全部都被满足（一些公认的内部必要的和预期的容差阈值）。另外，我们提供了使用参数化网络规模和恶意用户数量以及网络延迟来模拟共识处理的代码。

# 定义，形式化和先期工作

我们从定义Ripple协议的组成部分开始。为了证明正确性，一致性和可用性等属性，我们首先把这些属性形式化为公理。这些属性，当组合在一起，就形成了共识的概念：网络中节点到达了正确和一致的状态，然后我们突出一些以前和共识算法有关的成果。最后陈述在我们的形式框架中，共识算法对于Ripple协议的目标。

## Ripple协议组件

我们从定义以下项开始我们的描述：

* **Server**：一个服务器是运行参与共识处理的Ripple Server软件（相对Ripple客户端软件，客户端只是让用户发送和接收资金）的任何实体。
* **Ledger**：总账是货币在每个用户的帐户金额的纪录，代表了网络的“地面实况”。总账反复更新，通过共识的过程成功地达成交易。
* **Last-Closed Ledger**: 最后－关闭账单是被共识过程批准的最近的账单，并代表网络的当前状态。
* **Open Ledger**: 打开账单是当前节点处理的状态（每个节点维护自己的打开账单）。交易被终端用户发起并提交到server的打开账单中，但是交易直到达成共识过程才会被考虑，这时打开账单变为最后－关闭账单。
* **Unique Node List(UNL)**:每一个服务器s都维护一个唯一节点列表，它是s在确定共识时请求其他server的集合。在确定共识时只有s的UNL中的其他成员的表决才会被考虑（相对于网络中每个节点）。因而UNL相当于网络中的一个子集，它被s集体的“信任”，而不会共谋尝试欺骗网络。注意，这个“信任”的定义并不要求UNL中每个单独的成员都被信任（详见3.2小节）。
* **Proposer**:任何server能够广播共识过程中包含的交易，当一个新的共识过程开始时，每个server都试图包含每个有效的交易。然而，在共识处理过程中，只有server s的UNL中的服务器的提议，才会被s所考虑。

## 形式化

我们使用单词*nonfaulty（无缺陷）*指示网络中的节点都是诚实的并且没有错误。相反的，一个*faulty*可以是诚实的节点，但是出现错误（数据错误，实现错误等），或者是恶意的节点（拜占庭错误等）。我们减少验证交易到一个简单的二进制决策问题的概念：每个节点必须决定形成它的值被赋予为0或1。

就像Attiya，Deolev，和Gill，1984[3]，我们根据下面三个公理定义达成共识：

**1.(C1):**在最后时间，每个nonfaulty必须作出决定

**2.(C2):**所有nonfautly节点到达一个相同的决定值

**3.(C3):**对于nonfaulty节点来说，0和1的值都是可能的（这里移除了微不足道的方案，其中所有节点决定0或者1，而不管他们所提出信息）。

## 已存在的共识算法

在面对拜占庭错误时，已经有很多达成共识的算法研究被完成。先前的工作包括在消息异步发送之前不知道网络中所有的参与者（不存在绑定单个节点能够做出决定的时间），以及强与弱共识概念的划分。

以前有关共识算法的工作成果是由Fischer, Lynch和Patterson在1985 [4]年提出的，他们证明了在异步条件下，非终结对于共识算法始终是可能的，即使就一个faulty处理。这引入了对基于时间探测的必要性，以确保收敛（或不收敛的最小重复迭代）。我们将在第3节介绍这些探测式的Ripple协议。

共识算法的强度通常使用可以容忍错误处理的数量衡量。可以证明，在容忍超过(n-1)/3拜占庭错误，或者超过33%的恶意网络行为，拜占庭问题是无法解决的。然而，该方案不要求验证节点间分发消息的真伪（数字签名）。如果确保消息不可伪造是可能的，在同步情况下，算法适用于高的多的容错。

对于异步情况下的拜占庭共识，多种算法以更大的复杂性被提出。在n个节点的网络里，Fab Paxos[5]会容忍(n-1)/5的拜占庭错误，容错恶意串通网络节点的数量接近20%。Attiya, Doyev, and Gill [3]引入了一个在异步情况下的相位算法（phase algorithm），它可以容错(n-1)/4，接近网络的25%。最后，Alchieri等人在2008[6]发表了BFT-CUP，实现在异步条件下拜占庭共识，甚至包含未知的参与者，最大的容错边界为(n-1)/3，但是附加的条件是保证底层网络的连接。

## 正式的共识目标

我们在这项工作的目的是表明，Ripple协议使用共识算法实现每一个ledger-close（甚至是所有交易被拒绝的无关紧要的共识），即使是面对拜占庭失败，在已知可能的情况下，无关紧要的共识被达成。在网络每个节点，只有受信任的节点集合（UNL中的其他节点）在proposals（提案）上表决，由于每个节点可以有不同的UNLs，我们还表明，只有一个共识将到达所有节点，而不论节点间的成员关系。这个目标也称为防止网络“分叉（fork）”：其中一种情况，两个不相交的集合的节点的每一个独立地达成共识，另一种情况是两个不同的last-closed ladgers被每一个节点集合所观察。最后，我们将表明，Ripple协议在面对(n-1)/5的失败，能够实现这些目标。在上述文献中，这并不是最强的结果，但是我们将表明Ripple协议拥有其他一些可取的特点而大大提升了实用性。

# Ripple共识算法

Rippleg协议共识算法（RPCA），由网络中的所有节点，每隔几秒应用一次，为了维护网络的正确性和一致性。当前账单被认为“关闭的”，并变为last-closed ledger。假设共识算法是成功的，并且网络中没有分支，网络中所有节点保持的last-closed ledger将是相同的。

## 定义

RPCA处理需要经过几轮，在每一轮里：

* 最初，每个server把看到的所有没被处理的有效交易放到本轮共识开始之前（这可能包含server 的终端用户发起的新的交易，也可能是上一个共识没有处理的交易，等等），并把它们公开为列表的形式，称之为“候选集”。
* 然后每个server合并它的UNL所有servers的候选集，并为所有交易的准确性表决。
* 接收超过最小百分比“yes”表决票的交易进入下一轮，而没有获得足够票的交易要么被丢弃，要么包含于共识处理下一个账单开始的候选集。
* 共识的最后一轮要求UNL中servers最小百分比80%赞同一个交易。满足这一要求的所有交易被加入将要被关闭的账单中，形成last-closed ledger。

## 正确性

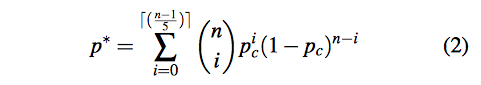
为了实现正确性，鉴于拜占庭失败的最大数量，必须证明在共识确认时，欺诈交易是不可能的。除非faulty的数量超过了容忍度。RPCA的正确性证明如下：由于一个交易只要80%UNL的servers赞同并批准，并且80%UNL的节点是诚实的，那么没有欺骗的交易被批准。因此，对于一个网络中有n个节点的UNL，共识协议维护的正确性为：

屏幕快照 2014-11-10 16.57.55.png

其中f是拜占庭失败的数量。事实上，即使面对(n-1)/5+1 的拜占庭失败，正确性在技术上仍然保持不变。共识过程是会失败的，但是仍然不可能确认一个欺诈交易。事实上，需要(4n+1)/5的拜占庭失败才可能确认一个错误的交易。我们把第二个边界称之为弱共识，前一个边界称之为强共识。

还应该注意的是，并不是所有的“欺诈”交易都构成威胁，即使在共识过程中确认。例如当一个用户试图两个交易中双重花费他的资金，即使两个交易都在共识处理过程中被确认，当第一个交易被接受后，第二个交易就会失败，因为资金不可用。这种鲁棒性是由于交易被确定性施加的一个事实，共识确保了网络中所有节点对于相同的一组交易都被施加了确定性规则。

一个稍微不同的分析，让我们假设，任何节点决定串通加入“邪恶组织”的概率为Pc，正确性的概率为P\*，那么：



这个概率代表的可能性，对于给定的Pc，邪恶组织的规模将低于拜占庭失败的最大阈值。由于这可能是一个二项式分布，Pc的值大于20%，将导致大于占网络20%“邪恶组织”规模的预期，从而阻碍了共识过程。在实践中，UNL选取不是随机的，而是意图使Pc最小化。由于节点不是匿名的，而是加密的身份。从混合洲际，国家，行业，意识形态等中的节点选取UNL，产生超过20%的Pc值可能性要小的多。作为一个例子，反诽谤联盟和威斯特布路浸信会的勾结，骗取网络的可能性，肯定是比20％要小很多。即使UNL具有相对较大的Pc，比方说15％，正确性的概率也非常高，即使只用200个节点的UNL，正确性也高达97.8％。

一幅图表明正确性的概率随着UNL大小和Pc值的变化，如图1所示：在这里，垂直轴表示邪恶组织挫败共识的概率，因此较低的值表明共识成功的概率高。可以看出，图中，即使Pc值高达10%，随着UNL的增长，超过100，很快就可以忽略不计。

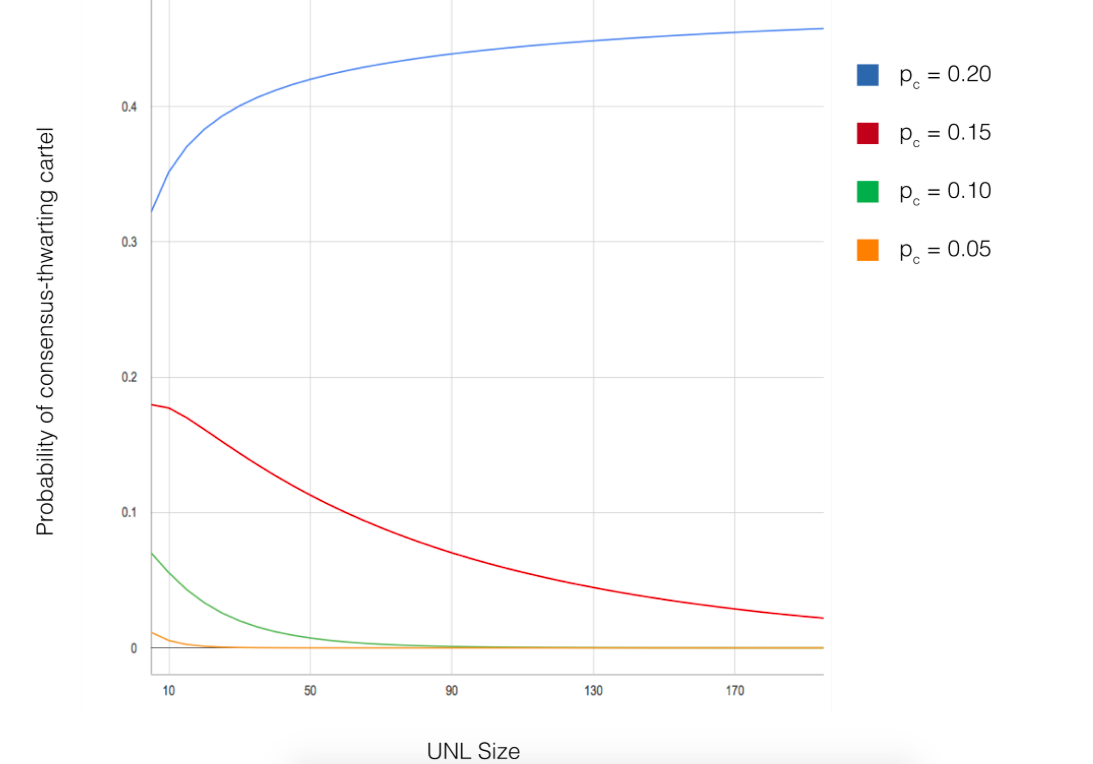


图1:“邪恶组织”挫败共识的概率和UNL规模的函数，对不不同的Pc值，就是UNL中成员相互串通欺诈的概率。越低的值表明共识成功率越高。

## 一致性

满足一致性要求，必须表明所有的nonfaulty节点在相同的交易集达成共识，而不理会它们的UNLs。由于每个server的URLs可能不同，一致性不能依赖正确性获取内部保证。例如，不会限制UNL成员之间的关系，以及UNL的规模不大于0.2\*ntotal，这里ntotal是整个网络的节点数量，并且，分支也是可能的。这通过一个简单的例子说明（如图2）。想象UNL图中两个小集团，每个都大于0.2\*ntotal。小集团意味着，内部节点的集合就是每个节点UNL节点的集合。由于两个小集团不共享任何成员，它们彼此独立的形成正确的共识，这违反了一致性。如果两个小集团的连接超过0.2\*ntotal，网络分支就不再可能，小集团之间的不一致性防止共识达到80%一致性的阈值。

证明连接性的所需要的上限由下面公式给出：

屏幕快照 2014-11-10 21.34.55.png

这个上限假定UNLs的集团状结构，即节点形成组，其UNLs包含在这些组的其它节点。这个上限保证没有两个小集团能够在相互矛盾的交易上达成共识，因为它不可能达到共识所需的80％的阈值。当UNLs之间的间接边缘被考虑，更紧密的结合是可能的。例如，如果网络的结构并不是集团状，分支变得更难以实现，因为所有节点的UNLs的成为更大的纠缠。

有趣的是注意到，在交叉节点的性质上没有作出假设。两个UNLs的交集可以包括故障节点，但只要交点的尺寸大于所述需要保证协议的上限，那么有故障的节点的总数势必小于需要满足强正确性，然后无论如何正确性和一致性将得以实现。也就是说，一致性是仅仅依赖节点的交叉点的大小，而不是在nonfaulty节点的交叉点的大小。

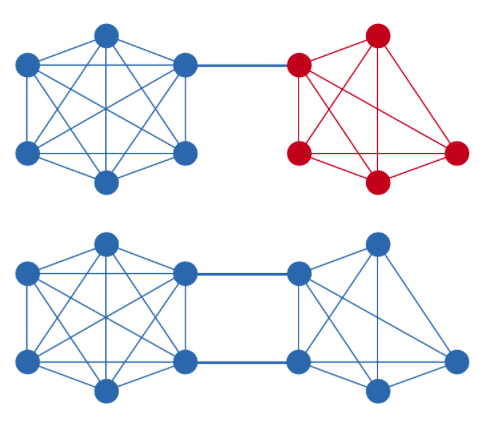


图2: 一个例子关于所需要的连接阻止两个小集团UNLs的分支

## 可用性

可用性的许多部分都是主观的，一个是确实可证是收敛：即共识的过程将在有限时间内终止。

### Convergence收敛

我们定义的收敛是在RPCA在强正确性下的账单达成共识，然后账单成为last-closed ledger的那个时间点。注意，在简单的情况下，虽然技术上弱正确性仍能表现表该算法的收敛性，但由于命题C3被侵犯，所以没有交易会被确认。从上面的结果，我们知道，强正确性在面对接近(n-1)/5的拜占庭失败时总是能够实现，而在整个网络中也只有一个共识被实现，只要UNL的连接条件被满足（等式3）。所有余下的都表明，当这两个条件都满足时，在有限时间内可以达成共识。

由于一致性算法本身是确定性的，并且预设数量执行轮数 t，共识终止之前，当前的交易集合是否申报批准或不批准（即使在这一点上没有一个交易超过80％的一致性要求，那么共识只是微不足道的共识），该算法终止的限制因素是节点之间的通信延迟。以约束该数量，节点的响应时间是被监视的，如果节点的延迟增加到预先设定的边界b时，那么节点将从UNLs中移除。从而保证了共识终止时间将有一个上限为tb，需要注意的是，边界描述的正确性和一致性必须由最终UNL满足，也就是删除之后的所有节点。如果对于初始UNLs所有节点该条件成立，但随后一些节点由于延迟从网络中去除，正确性和一致性保证不会自动保持，必须通过新的设定UNLs的来满足。

### Heuristics and Procedures

正如上面提到的，延迟边界的试探在Ripple网络中的所有节点强制执行，以保证共识致性算法将收敛。此外，对于RPCA，还有一些其他的试探性的检测和实用的程序被提供。

* 对于所有节点。有一个强制性的2秒窗口，提出在每一轮的共识中最初的候选集。这导致每一轮共识的边界下限变为2秒，这也保证了通过合理的延迟时间，所有节点有能力参与共识过程。
* 由于每一轮共识的表决都会被记录在账单上，节点可以被标记，并从网络中删除了一些常见的，易识别的恶意行为。这包括始终建议而不是通过共识来确认交易对每笔交易表决为“否”的节点。
* 一个默认UNL提供给所有的用户，这种选择最小化Pc的值，这在3.2小节里描述。而用户可以并应该选择自己UNLs，这个默认的节点列表保证甚至缺乏经验的用户会参与一个共识的过程，实现正确性和一致性具有非常高的概率。
* 网络分割检测算法也被拿来用以避免网络中分支的问题。而共识算法证明，在last-closed ledger的交易都是正确的，它并不禁止多个last-closed ledger出现在不同的连接差的网络分段的可能性。尝试并确定这种分割是否已发生，每一个节点监视其UNL的活动成员的数量大小。如果突然下降到低于预设阈值时，有可能发生了分裂。为了防止节点在UNL大部分具有临时延迟的情况下不够积极，节点被允许发布一个“局部验证“，他们并不处理或对交易进行表决，但声明仍然参与达成共识的过程，而不是一个断开连接的子网上参与不同的共识过程。
* 虽然有可能只进行一轮共识RPCA就被接受，但是可用性通过多轮会增长，每一轮都会增加一致性的最低要求的百分比，直到最后一轮的80％要求之前。这些轮过程允许检测延迟的潜在节点，这些节点很少，但却增加了网络成交率的瓶颈。这些节点能够最初在低要求的几轮里保持很好地协同工作，但是后来就被落下，并且随着阈值的增加而被确认。在一轮中就达成共识的情况下，可能很少的交易通过80％的阈值，即使缓慢的节点能够跟上，但是降低了整个网络的成交率。

# 仿真代码

提供的仿真代码演示了一轮RPCA，用参数设定的功能（网络的节点大小，恶意节点，消息延迟等的数量）。模拟器在完美分歧开始（网络中的一半的节点的最初提出的“是”，而另一半则提出“否”），随着共识过程的前进，显示出在每个阶段的有/无表决的网络节点数量网络节点根据他们UNL成员的建议进行调整。一旦达到80％的阈值，就达成共识。我们鼓励读者在“Sim.cpp”的开头定义的常量不同的值进行实验，以熟悉不同的条件下达成的共识的过程。

# 讨论

我们已经描述了RPCA，满足我们在上面已经列出的正确性，一致性和可用性。其结果是，Ripple协议能够在几秒钟之内处理的安全和可靠的交易数据：时间长度需要1轮的共识来完成。这些交易可证明安全上限在第3小节所列的范围，对比文献中的异步拜占庭共识，它们虽然不是最强的，但是能够迅速收敛，网络成员之间并有弹性。当这些结合在一起时，这些特质使得Ripple网络具有很好理解的安全性和可靠性，能够充当快速和低成本的全球支付网络。

虽然我们已经表明，Ripple协议可证明是安全的，在方程式的1和3中的边界范围都满足，值得注意的是这些是最大边界，在实践中，网络不太严格的条件下是安全的。然而同样重要的是要认识到，满足这些界限不是RPCA本身固有的，而要求所有用户对UNLs的管理。虽然提供给所有用户的默认UNL已经足够，但是用户在进行更改UNL时，必须使用上述边界的知识来实现。此外，需要在全局网络结构的进行一些监控，以确保该上限在方程式（3）得到满足，那么一致性将总是被满足。

我们相信RPCA对于分布式支付系统是显著的进步，以低延时实现那些先前由于较高延时共识方法实现困难的甚至不可能的很多种类的金融交易。

# 感谢

Ripple实验室要感谢所有参与的Ripple协议共识算法发展的人们。特别是，Arthur Britto，为他在交易集方面的工作，Jed McCaleb，为他提出原始Ripple协议共识的概念，和David Schwartz，为他在共识方面的“赞同失败是一致性的延缓”。Ripple实验室还要感谢Noah Youngs，他在编制和审查本文的努力。

# 引用

[1] Nakamoto, Satoshi. “Bitcoin: A peer-to-peer electronic cash system.” Consulted 1.2012 (2008): 28.

[2] Lamport, Leslie, Robert Shostak, and Marshall Pease. “The Byzantine generals problem.” ACM Transactions on Programming Languages and Systems (TOPLAS) 4.3 (1982): 382-401.

[3] Attiya, C., D. Dolev, and J. Gill. “Asynchronous Byzantine Agreement.” Proc. 3rd. Annual ACM Symposium on Principles of Distributed Computing. 1984.

[4]Fischer, Michael J., Nancy A. Lynch, and Michael S. Paterson. “Impossibility of distributed consensus with one faulty process.” Journal of the ACM (JACM) 32.2 (1985): 374-382.

[5]Martin, J-P., and Lorenzo Alvisi. “Fast byzantine consensus.” Dependable and Secure Computing, IEEE Transactions on 3.3 (2006): 202-215.

[6]Alchieri, Eduardo AP, et al. “Byzantine consensus with unknown participants.” Principles of Distributed Systems. Springer Berlin Heidelberg, 2008. 22-40.