全文摘要：3.3 协议为满足协议要求，必须证明所有非故障节点在同一组交易中达成共识，而不管它们的 UNL 如何 ... 因此，对于网络中的 n 个节点的 UNL，一致协议将保持正确性，只要：（1）其中 f 是拜占庭故障的数量 ... 2.4 正式共识目标我们在这项工作中的目标是证明 Ripple Protocol 使用的一致性算法将在每个分类账结束时达成共识（即使共识是所有交易被拒绝的普遍共识），并且只会达成共识一个已知的概率，即使在拜占庭失败的情况下 ...

注：此版本由瑞波 2014 版英文白皮书翻译而来。

摘要

尽管拜占庭将军问题存在多种共识算法，特别是与分布式支付系统有关的问题，但许多人都有由网络内所有节点同步通信的要求引起的高延迟。在这项工作中，我们提出了一种新的共识算法，通过在更大的网络中利用集体信任的子网络来规避这一要求。我们表明，这些子网络所需的“信任”实际上是最小的，并且可以通过成员节点的原则性选择进一步降低。另外，我们表明，需要最小的连接来维持整个网络的协议。结果是低延迟一致性算法在拜占庭式失败的情况下仍然保持鲁棒统计。（“鲁棒统计：一种在其假定被数据产生在的真实模型所违背的情况下依然能工作良好的统计技术。”——蛋蛋注） 我们在 Ripple 协议的实施例中介绍这种算法。

引言

近年来，对分布式共识系统的兴趣和研究显着增加，重点在于分布式支付网络。这种网络允许快速，低成本的交易，不受中央资源的控制。虽然这种系统的经济效益和缺点值得自己进行大量的研究，但这项工作关注的是所有分布式支付系统必须面对的一些技术挑战。虽然这些问题各不相同，但我们将它们分为三大类：正确性，一致性和实用性。

通过正确性，我们的意思是分布式系统有必要能够辨别正确和欺诈性交易之间的差异。在传统的信托环境中，这是通过机构之间的信任和加密签名来完成的，以确保交易确实来自它声称来自的机构。然而，在分布式系统中，不存在

这种信任，因为网络中的任何和所有成员的身份可能甚至不可知。因此，必须使用正确性的替代方法。

协议是指面对分散的会计制度时维护单一全球真相的问题。虽然与正确性问题相似，但区别在于，虽然网络的恶意用户可能无法创建欺诈性交易（违反正确性），但它可能能够创建多个正确的交易，这些交易在某种程度上不知道彼此，从而结合起来造成欺诈行为。例如，恶意用户可以同时进行两次购物，其账户中只有足够的资金可以分别支付每笔购买，但不能同时购买。因此，每个交易本身都是正确的，但如果同时执行，使得整个分布式网络都不知道这两者，就会出现一个明显的问题，通常称为“双重支出问题”[1]。因此，协议问题可以概括为在网络中只存在一组全球认可的交易的要求。

效用是一个稍微更抽象的问题，我们通常将其定义为分布式支付系统的“有用性”，但实际上这通常会简化为系统的延迟。例如，一个既正确又协调，但需要一年处理交易的分布式系统显然是一种不可靠的支付系统。实用程序的其他方面可能包括参与正确性和协议流程所需的计算能力级别或最终用户为避免在网络中受骗而需要的技术熟练程度。

许多这些问题早在现代分布式计算机系统出现之前就已经被探讨过了，这个问题通过一个被称为“拜占庭将军问题”的问题[2]。在这个问题中，一群将军每个控制一个军队的一部分，并且必须通过发送信使来协调攻击。由于将军们在陌生和敌对的领土上，信使可能无法到达目的地（正如分布式网络中的节点可能失败，或发送损坏的数据而不是预期的消息）。问题的另外一个方面是，一些

将军可能是叛徒，不论是单独的，还是共谋的，所以消息可能会到达，这些消息的目的是为忠诚的将军制造一个注定要失败的错误计划（就像恶意成员一样的分布式系统可能会试图说服系统接受欺诈性交易，或者导致双重支出的同一真实交易的多个版本）。因此，分布式支付系统必须在面对标准故障和所谓的“拜占庭式”故障时保持稳健，这些故障可能由网络中的多个来源协调并产生。

在这项工作中，我们分析了分布式支付系统的一个特定实现：Ripple 协议。我们专注于用于实现上述正确性，一致性和实用性目标的算法，并且显示所有算法都已满足（在必要的和预先确定的公差范围内，这些都是很好理解的）。此外，我们提供的代码可模拟具有可参数化网络大小，恶意用户数量和发送消息时间的共识流程。

2.定义，形式化和以前的工作

我们首先定义 Ripple 协议的组件。 为了证明正确性，一致性和效用属性，我们首先将这些属性形式化为公理。 这些属性在组合在一起时形成共识的概念：网络中的节点达到正确协议的状态。 然后，我们重点介绍以前与共识算法有关的结果，并最终在我们的形式化框架中阐述 Ripple 协议的共识目标。

2.1 瑞波协议组件

我们通过定义以下术语来开始描述瑞波网络：

服务器：服务器是运行 Ripple 服务器软件的任何实体（与仅允许用户发送和接收资金的 Ripple 客户端软件相反），其参与共识过程。

分类账：分类账是每个用户账户中货币数量的记录，代表了网络的“基本事实”。分类账重复更新，成功通过共识流程。

最后关闭分类账：最后关闭分类账是最近一次由共识流程批准的分类账，因此代表了网络的当前状态。

开放式分类账：开放式分类账是当前节点的运行状态（每个节点维护自己的开放分类账）。由给定服务器的最终用户发起的交易应用于该服务器的开放分类帐，但在通过共识流程之前，交易不会被视为最终交易，此时开放分类帐会成为最后一笔结账分类帐。

唯一节点列表（UNL）：每个服务器都维护一个唯一的节点列表，该列表是一组在确定共识时进行查询的其他服务器。在确定一致意见时（而不是网络上的每个节点），只考虑联合国其他成员的选票。因此 UNL 代表了网络的一个子集，当被集体采用时，它是“可信”的，不会串谋欺骗网络。请注意，这个“信任”的定义并不要求 UNL 的每个成员都是可信的（参见 3.2 节）。

Proposer：任何服务器都可以广播要包含在共识流程中的事务，并且每当一个新的共识回合开始时，每个服务器都会尝试包含每个有效的事务。然而，在协商一致过程中，只有来自服务器的 UNL 上的服务器的建议才会被考虑。

2.2 形式化

我们使用 nonfaulty 这个术语来指代网络中的节点，这些节点的行为诚实且没有错误。相反，故障节点是一个可能是诚实的错误（由于数据损坏，执行错误等）或恶意（拜占庭错误）。

我们将验证交易的概念简化为一个简单的二元决策问题：每个节点都必须根据已给定的值 0 或 1 来决定。

正如 Attiya，Dolev 和 Gill，1984 [3]中所定义的，共识 根据以下三个公理：

1.（C1）：每个非故障节点在有限时间内做出决定

2.（C2）：所有非故障节点达到相同的决策值

3.（C3）：0 和 1 都是可能的值 所有非故障节点。 （这消除了所有节点决定 0 或 1 的简单解决方案，而不管它们已经被呈现的信息）。

2.3 现有的一致性算法

在面对拜占庭误差时达成共识的算法已经做了很多研究。此前的工作包括扩展到网络中的所有参与者未提前知道的情况，消息以异步方式发送（单个节点为达成决策所花费的时间没有限制），以及在何处强弱共识的概念之间有一个界限。

以前在共识算法方面的工作的一个相关结果是 Fischer，Lynch 和 Patterson，1985 [4]，这证明了在异步情况下，即使只有一个错误的过程，非终止始终是共识算法的可能性。这引入了基于时间启发式的必要性，以确保收敛（或至少反复迭代非收敛）。我们将在第 3 节中描述 Ripple 协议的这些启发式算法。

一致性算法的强度通常以它可以容忍的错误进程的比例来衡量。可以证明，拜占庭将军问题（已经假定同步性和已知参与者）的解决方案不能容忍超过（n-1）/ 3 个拜占庭故障，或 33％的恶意网络行为[2]。但是，该解决方案不需要验证在节点之间传递的消息的真实性（数字签名）。如果可以保证消息的不可伪造性，那么在同步情况下算法存在更高的容错性。

针对异步情况下的拜占庭共识提出了几种更复杂的算法。 FaB Paxos [5]将容忍（n-1）/ 5 个 n 节点网络中的拜占庭故障，相当于网络中多达 20％的节点恶意串通。 Attiya，Doyev 和 Gill [3]为异步情况介绍了一种相位算法，该算法可以容忍（n-1）/ 4 次失败，或者高达 25％的网络。最后，Alchieri 等，2008 [6]提出了 BFT-CUP，即使在未知参与者的情况下，也可以在异步情况下实现拜占庭共识，并具有（n-1）/ 3 个失效容限的最大界限，但有额外限制关于底层网络的连接。

2.4 正式共识目标

我们在这项工作中的目标是证明 Ripple Protocol 使用的一致性算法将在每个分类账结束时达成共识（即使共识是所有交易被拒绝的普遍共识），并且只会达

成共识一个已知的概率，即使在拜占庭失败的情况下。由于网络中的每个节点只会对来自可信节点集（UNL 中的其他节点）的提案进行投票，并且由于每个节点可能具有不同的 UNL，因此我们还表明，只有一个共识将在所有节点之间达成，无论联合国成员资格。这个目标也被称为阻止网络中的“分支”：两个不相交的节点集各自独立达成共识，并且每个节点集上的节点观察到两个不同的最后关闭分类账。

最后，我们将证明 Ripple 协议可以在（n-1）/ 5 次失败的情况下实现这些目标，这不是文献中最强烈的结果，但是我们也会表明 Ripple 协议具有其他一些理想的特性，大大提高其实用性。

3.波纹共识算法

波纹协议共识算法（RPCA）每隔几秒钟由所有节点应用，以保持网络的正确性和一致性。 一旦达成共识，当前分类账被视为“封闭”，并成为最后封闭的分类帐。 假定一致性算法是成功的，并且网络中没有分叉，那么网络中所有节点维护的最后一个封闭式分类账将是相同的。

3.1 定义

RPCA 轮流进行。 在每一轮中：

最初，每个服务器在合意轮开始之前已经看到所有尚未应用的有效事务（这些事务可能包括由服务器的最终用户发起的新事务，从先前的共识流程持续的事务等）。 ），并以被称为“候选集合”的列表的形式公布它们。

然后，每台服务器合并 UNL 上所有服务器的候选集，并对所有事务的真实性进行投票。

如果收到超过最低百分比的“是”票的交易被传递到下一轮（如果有的话），而没有获得足够票数的交易将被丢弃或被包括在候选组中以用于下一个分类账的共识流程。

最后一轮共识要求服务器 UNL 的 80％的最低百分比同意交易。所有符合此要求的交易都将应用于分类账，并且该分类账已关闭，成为新的最后一笔结账分类账。

3.2 正确性

为了实现正确性，给定最大数量的拜占庭式失败，必须证明，在共识期间不可能确认欺诈性交易，除非有错误的节点数量超过容忍度。 直接证明 RPCA 的正确性：因为只有在服务器的 UNL 的 80％同意它时才能批准交易，只要 UNL 的 80％是诚实的，则不会批准欺诈交易。 因此，对于网络中的 n 个节点的 UNL，一致协议将保持正确性，只要：

（1）其中 f 是拜占庭故障的数量。 事实上，即使面对（n-1）/ 5 + 1 拜占庭式的失败，正确性在技术上仍然保持着。 共识流程将失败，但仍不可能确认欺诈性交易。 事实上，需要（4n + 1）/ 5 拜占庭式的失败才能确认不正确的交易。 我们称这第二个边界为弱正确性的边界，而前者是强正确性的边界。

还应该指出，并非所有“欺诈”交易都构成威胁，即使在协商一致期间得到确认。 例如，如果用户尝试在两次交易中双倍使用资金，即使两次交易在共识流程期间得到确认，在第一次交易应用后，第二次交易将失败，因为资金不再可用。 这种稳健性是由于事务被确定性地应用的事实，并且该共识确保网络中的所有节点将确定性规则应用于同一组事务。

对于稍微不同的分析，让我们假设任何节点决定串通并加入恶意卡特尔的概率是 pc。那么正确性的概率由 p *给出，其中：

（2）这个概率表示恶意卡特尔的大小将保持在拜占庭失败的最大阈值以下的可能性，给定 pc。由于这种可能性是二项分布，PC 值大于 20％会导致预期卡特尔尺寸大于网络的 20％，从而阻碍了共识过程。实际上，UNL 不是随机选择的，而是为了尽量减少 pc。由于节点不是匿名的，而是密码可识别的，因此从大陆，国家，行业，意识形态等混合选择 UNL 节点将产生远低于 20％的 pc 值。例如，反诽谤联盟和 Westboro Baptist Church 勾结欺骗网络的可能性肯定很大，远低于 20％。即使 UNL 具有相对较大的 pc，比如说 15％，即使 UNL 中只有 200 个节点：97.8％正确率的可能性也非常高。

图 1 描述了不正确的概率如何作为 UNL 大小的函数对 PC 的不同值进行缩放的图示。请注意，此处垂直轴代表恶意卡特尔挫败共识的概率，因此较低的值表示较大的概率共识成功。从图中可以看出，即使 pc 高达 10％，随着 UNL 增长超过 100 个节点，共识被挫败的可能性很快变得微不足道。

3.3 协议

为满足协议要求，必须证明所有非故障节点在同一组交易中达成共识，而不管它们的 UNL 如何。由于每台服务器的 UNL 可能不同，因此协议并非由正确性证明所固有。例如，如果 UNL 的成员资格没有限制，并且 UNL 的大小不大于 0.2 * ntotal，其中 ntotal 是整个网络中的节点数，则可以使用分叉。这可以通过一个简单的例子来说明（如图 2 所示）：想象 UNL 图中的两个派系，每个派系都大于 0.2 * ntotal。根据派系，我们指的是一组节点，其中每个节点的 UNL 是同一组节点。由于这两个派系不共享任何成员，因此每个人都有可能彼此独立达成正确的共识，违反协议。如果两个派系的连通性超过 0.2 *总计，那么分叉将不再可能，因为派系之间的分歧会阻止达成达成 80％协议阈值所需的共识。

连接所需的上限

图 2.防止两个 UNL 派系之间分叉所需的连接示例。

证明协议由以下条款给出：

（3）这个上界假设了一个类似团结构的 UNLs，即节点形成其 UNL 包含这些集合中其他节点的集合。 这个上限保证了没有两个派系能够就冲突交易达成共识，因为达成共识所需的 80％阈值变得不可能。 当 UNL 之间的间接边缘也被考虑在内时，可能更紧密的边界。 例如，如果网络的结构不是类似团体的，由于所有节点的 UNLs 更大的纠缠，分支变得更加难以实现。

值得注意的是，没有关于相交节点的性质的假设。 两个 UNL 的交集可能包含故障节点，但只要交集的大小大于保证一致所需的界限，并且故障节点的总数少于满足强正确性所需的界限，则两个正确性 并达成协议。 也就是说，协议仅取决于节点交集的大小，而不取决于非故障节点交集的大小。

3.4 效用

虽然效用的许多组成部分都是主观的，但确实可证明的是收敛性：共识过程将在有限的时间内终止。

图 1.一个恶意卡特尔可能阻碍共识作为 UNL 规模的一个函数，对于 pc 的不同价值，联合国任何成员将决定与他人勾结的可能性。 这里，较低的值表示共识成功的可能性较高。

3.4.1 合流点

我们将收敛定义为 RPCA 在分类账上达到一致并具有较强正确性的点，并且该分类账成为最后一笔收入分类账。请注意，虽然技术上的弱正确性仍然代表算

法的收敛性，但它只是在平凡的情况下收敛，因为命题 C3 被违反，并且不会确认交易。根据以上结果，我们知道，面对高达（n-1）/ 5 次拜占庭式失败时，始终可以实现强大的正确性，只要 UNL 连通性条件为满足（等式 3）。剩下的就是要表明，当这两个条件都满足时，就会在有限的时间内达成共识。

由于共识算法本身是确定性的，并且具有预设数量的轮次 t，所以在共识终止之前，并且当前一组交易被宣布为批准或未批准（即使此时没有交易具有超过 80％需要达成协议，而共识只是微不足道的共识），终止算法的限制因素是节点之间的通信延迟。为了限制这个数量，监控节点的响应时间，并且延迟增长的节点比预先设定的界限 b 大的节点从所有的 UNL 中移除。

虽然这保证了共识将以 tb 的上限结束，但重要的是要注意，上面描述的正确性和一致性的边界必须由最终的 UNL 满足，在所有将被丢弃的节点都被丢弃之后。如果条件适用于所有节点的初始 UNL，但由于等待时间而导致某些节点从网络中丢弃，则正确性和一致性保证不会自动保持，但必须由新的 UNL 集满足。

3.4.2 启发式和程序

如上所述，在 Ripple 网络中的所有节点上实施延迟束启发式以保证一致性算法将合流。此外，还有一些启发式和程序为 RPCA 提供了实用性。

所有节点都有一个强制性的 2 秒钟窗口，以在每轮协商一致中提出初始候选集。虽然这确实给每个共识轮次引入了 2 秒的下限，但它还保证具有合理延迟

的所有节点都将有能力参与共识流程。由于投票记录在分类账的每一轮共识中，因此可以标记网络中的节点并将其从网络中删除，以获取一些常见的，易于识别的恶意行为。这些包括对每个事务投票为“否”的节点以及一致提出未通过一致性验证的事务的节点。

为所有用户提供策划的默认 UNL，这是为了最大限度地减少 pc，如第 3.2 节所述。尽管用户可以并且应该选择他们自己的 UNL，但这个默认的节点列表保证了即使是天真的用户也将参与一致的过程，以极高的概率达到正确性和一致性。网络分裂检测算法也被用来避免网络中的分叉。虽然共识算法证明最后关账分类账上的交易是正确的，但它并不禁止在网络的不同分部存在多于一个最后关账分类账的可能性，因为连接不畅。为了试图识别是否发生了这样的分裂，每个节点监视 UNL 的活动成员的大小。如果此大小突然下降到预设阈值以下，则可能发生拆分。为了防止在 UNL 的大部分具有暂时延迟的情况下误报，节点被允许发布“部分验证”，其中他们不处理或投票交易，但声明仍然参与一致的过程，而不是对断开的子网络采取不同的共识过程。

虽然可以仅在一轮协商一致中应用 RPCA，但在最后一轮之前需要 80％的要求，可以通过多轮获得效用，每个轮次的协议最低要求百分比都在增加。这些回合允许检测潜在节点，这是因为少数这样的节点正在创造网络事务速率的瓶颈。这些节点将能够在较低需求轮次期间最初保持上升，但落后并且随着阈值增加而被识别。在一轮协商一致的情况下，可能出现这样的情况，即很少的事务通过 80％的阈值，即使是较慢的节点也能跟上，降低整个网络的事务速率

4.仿真代码

所提供的仿真代码演示了一轮 RPCA，具有可参数化的功能（网络中的节点数量，恶意节点数量，消息延迟等）。 模拟器开始时完全不一致（网络中的一半节点最初建议“是”，而另一半建议“否”），然后继续进行共识流程，在每个阶段显示在 网络作为节点根据 UNL 成员的建议调整提案。 达到 80％的门槛后，达成共识。 我们鼓励读者尝试在“Sim.cpp”开头定义的不同常量值，以便熟悉不同条件下的一致过程。

5.讨论

我们已经描述了 RPCA，它满足了我们上面概述的正确性，一致性和实用性的条件。 其结果是 Ripple Protocol 能够在几秒钟内处理安全可靠的事务：完成一轮共识所需的时间。 这些交易可证明是安全的，直到第 3 节中概述的界限，虽然不是异步拜占庭共识文献中最强大的一个，但确实可以实现网络成员资格的快速收敛和灵活性。 综合起来，这些特质使 Ripple Network 成为一个快速和低成本的全球支付网络，并具有广为人知的安全性和可靠性。

虽然我们已经证明，只要满足公式 1 和 3 中描述的边界，Ripple Protocol 就可证明是安全的，但值得注意的是这些是最大边界，实际上网络在严格不太严格的条件下可能是安全的。 然而，认识到这一点也很重要，满足这些界限并不是 RPCA 本身固有的，而是需要管理所有用户的 UNLs。 向所有用户提供的默认 UNL 已经足够了，但是如果用户对 UNL 进行了更改，则必须在知道上述界限

的情况下完成。 另外，为了确保方程式 3 中的界限得到满足并且始终满足该协议，需要对全球网络结构进行一些监测。

我们认为 RPCA 代表了分布式支付系统向前迈出的重要一步，因为低潜能性允许以前使用其他更高延迟一致性方法难以或甚至不可能实现的许多类型的金融交易。

6.致谢

Ripple Labs 想要感谢所有参与 Ripple Protocol 共识算法开发的人员。 具体而言，Arthur Britto 因其关于交易集的工作，Jed McCaleb 就原始 Ripple Protocol 共识概念和 David Schwartz 所做的关于“未能达成一致意见是同意推迟”共识方面的工作。 Ripple Labs 还要感谢诺亚杨斯在编写和审阅本文时的努力。

参考

[1] Nakamoto，Satoshi。 “比特币：一个 P2P 电子现金系统。”咨询 1.2012（2008）：28.

[2] Lamport，Leslie，Robert Shostak 和 Marshall Pease。 “拜占庭将军问题。”ACM 编程语言和系统交易（TOPLAS）4.3（1982）：382-401。

[3] Attiya，C.，D. Dolev 和 J. Gill。 “异步拜占庭协议”。3。 年度 ACM 分布式计算原理研讨会。1984 年

[4] Fischer，Michael J.，Nancy A. Lynch 和 Michael S. Paterson。 “分布式共识不可能具有一个错误的过程。”ACM 杂志（JACM）32.2（1985）：374-382。

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

[6] Alchieri，Eduardo AP 等人。 “拜占庭与未知参与者的共识。”分布式系统原理。 Springer Berlin Heidelberg，2008. 22-40。