第3章 区块链共识

本章主要介绍共识机制的有关知识，它通常是区块链系统的核心。在理解共识机制前，需要学习分布式系统理论的基础知识。为帮助读者逐步掌握该方面知识，本章从拜占庭将军问题出发，通过 Dolev-Strong 协议为读者提供基础案例。进一步地，从现代分布式系统理论过渡到共识机制的现代应用：比特币共识（又称中本聪共识）。通过案例分析，让读者对区块链共识协议有一个直观的认识。


视频讲解

3.1 拜占庭将军问题

▶ 3.1.1 问题背景

Lamport 等在其开创性论文中，用下面的例子说明了共识需要解决的问题。想象一下，拜占庭军队的几个师扎营在敌方城市外，每个师由各自的将军指挥。将军之间只能通过信使联系。在观察敌情后，他们必须决定共同的行动计划。然而，有些将军可能是叛徒，试图阻止忠诚的将军达成一致。具体地，假设有 n 位将军，其中一位称为统帅。统帅想向所有将军下达一个要么进攻要么撤退的命令，则

（1）所有忠诚的将军都做出同样的决定。

（2）如果统帅是忠诚的，那么所有忠诚的将军都会服从统帅的命令。

Lamport 等将这一问题命名为拜占庭将军问题，它通常也被称为拜占庭广播（Byzantine Broadcast，BB）。需要注意的是，如果统帅被保证是忠诚的，那么这个问题就微不足道了：统帅可以向所有其他将军发送命令，而所有其他将军可以简单地服从（假设他们能核实命令确实来自统帅）。当然，统帅也可能是叛徒，在这种情况下，他可以向不同的将军提出不同的命令；因此，上述解决方案会导致不一致的决定。

如果统帅有腐败的可能，那么忠心耿耿的将军们是否仍有可能通过相互沟通来商定进攻计划呢？

上述假想情况可类比于计算机系统，在计算机系统中，一个或多个组件可能会出现故障。更确切地说，分布式系统中的分布式共识问题在于，虽然系统中的一些节点可能会以任意的方式行事，但正常运行的节点之间仍然需要就共同的价值达成一致。现实生活中的共识协议（如比特币）通常需要随着时间的推移反复达成共识，从而使节点共同维护一个不断增长、线性排序的交易日志，有时也称区块链。

▶ 3.1.2 问题定义

现在，将问题形式化。此后，把将军（或委员会成员）称为节点，把指挥官（或计划主席）称为指定发送人（简称发送人）。考虑一个由 n 个节点组成的分布式网络，节点编号分别为 $1,2,\cdots,n$。我们通常使用符号 $[n]:=\{1,2,\cdots,n\}$ 表示所有节点的集合。在不失一般性的前提下，可以假设发送者是节点 1，把自始至终遵守规定协议的节点称为诚实节点。然而，有一部分节点可能

是腐败的。这些腐败节点不必遵守规定的协议，它们可以在任意时间发送/传输任意信息、省略发送信息、停止或采取不正确的步骤。所有腐败节点都可以组成一个联盟，相互共享信息，实施协同攻击。为简单起见，设想所有腐败节点都由一个对手控制，但事先并不知道哪些节点是腐败节点。换句话说，无论哪个子集的节点被腐败，只要被腐败的节点总数的上限是 $f<n$，拜占庭广播协议就一定有效。

假设每一对节点都能相互通信，而且每个节点都有一对与数字签名方案相对应的公钥和密钥，所有节点的公钥都是公知的。在 3.2 节的 Dolev-Strong 协议中，每条信息都将由其创建者签名。使用符号 $\langle m\rangle_i$ 表示一对符号 (m,σ)，其中，σ 是对信息 m 的有效签名，可以用节点 i 的公钥验证。

简便起见，除非另有说明，假设在协议开始时选择了腐败节点集（事先遭受到腐败但不知道哪些节点腐败）。这种模式通常被称为静态腐败模型。在分布式系统和密码学文献中，另一种被称为自适应腐败的模型也得到了广泛研究。在自适应模型中，对手可以在观察到诚实节点发送的信息后，在协议执行过程中决定腐败哪些节点。

▶ 3.1.3 同步网络

假设协议是在同步网络中进行的，也就是说，当诚实节点发送信息时，诚实的接收者保证在一定时间内（如 1min）收到信息。无论信息延迟时间有多长，都可以将其定义为一轮。因此，假定协议的执行以轮为单位。在每轮开始时，所有节点都会接收到来自网络的信息。然后，它们执行一些本地计算，并发送新信息。以下内容由网络保证。

同步假设：如果一个诚实的节点在第 r 轮向诚实的接收者发送信息，那么接收者将在第 $r+1$ 轮开始时收到信息。

▶ 3.1.4 拜占庭广播

如定义 3.1 所示，在协议开始时，指定的发送者会收到一个输入比特 $b\in\{0,1\}$。然后，节点运行某个协议；在协议结束时，每个诚实的节点都会输出一个比特。

定义 3.1 拜占庭广播（BB）。

协议应该满足以下两个要求（无论节点如何腐败）。

一致性：如果两个诚实节点分别输出 b 和 b'，那么 $b=b'$。

有效性：如果发送者是诚实的，并接收到输入比特 b，则所有诚实的节点都输出 b。

需要注意的是，如果没有有效性要求，仅要求一致性就很容易实现：协议可以简单地要求每个节点都输出一个标准的 0 位。因此，只有同时要求这两个属性，问题定义才是非平凡且有意义的。这里，将假设签名方案是理想的，即不会发生签名伪造。此外，假设后面将要描述的 Dolev-Strong 协议是确定性的。由此一来，可以假定上述要求是确定性满足的。在后面的章节中，将考虑随机协议，在这种情况下，只要求上述属性在一般情况下成立（不成立的概率可忽略）。

3.2 Dolev-Strong 共识

▶ 3.2.1 一个简单但有缺陷的共识

如前所述，试想每个人都简单地遵循发送者发送的比特，这是行不通的。如果发送者是腐败的，可能向不同的节点发送了不同的比特，那么一致性就会受到破坏。

这样一来，自然的解决办法是基于投票的方法。方便起见，假定所有节点在发送信息前都会对所有信息进行签名，只有签名合法的信息才会被视为有效信息。所有无效信息都会被立即丢弃，不予处理。

如算法 3.1 所示，设想发送方在第一轮中签名并向所有人发送其输入比特。在第二轮中，每个人都会对他们从发送者那里听到的比特进行投票。如果没有听到任何比特或两个比特都听到了，则他们就会对一个标准比特 0 进行投票(输出)。现在，如果一个节点听到大多数节点投票支持(输出)b，那么它就会输出 b。下文将正式地描述这个简单的基于投票的协议，使用符号$\langle m\rangle_i$ 表示信息 m 及来自节点 $i\in[n]$的有效签名。

算法 3.1　一个简单的多数投票协议

(1) 发送者(即节点 1)接收比特 b 作为输入。

(2) 第一轮：节点 1 将$\langle b_1\rangle$发送给每个节点(包括自己)。

(3) 第二轮：每个节点 $i\in[n]$做如下操作，如果收到单个比特$\langle b'\rangle_1$，则发送投票$\langle b'\rangle_i$；否则发送投票$\langle 0\rangle_i$。

(4) 第三轮：如果没有比特或两个比特都从不同节点收到了超过 $n/2$ 的投票，则输出 0；否则输出从不同节点收到了超过 $n/2$ 的投票的比特。

这个协议有效吗？需要注意的是，腐败节点的行为可以是任意的，包括向不同节点发送不同的选票。事实证明，即使只有一个腐败节点，这个简单的协议也是不安全的。下面将描述一种一致性可能会被破坏的攻击，在这种攻击中，指定的发送者是唯一的腐败节点，而其他节点都是诚实的。

▶ 3.2.2　针对简单共识的攻击

假设 $n=2k+1$ 为奇数，只有发送方(即节点 1)是腐败的。将 $2k$ 个诚实节点分成两个不相交的集合，分别表示为 S_0 和 S_1，每个集合的大小为 k。在第一轮中，发送者向集合 S_0 发送$\langle 0\rangle_1$，向集合 S_1 发送$\langle 1\rangle_1$。在第二轮中，S_0 中的节点投 0 票，S_1 中的节点投 1 票。现在，腐败发送者巧妙地在集合 S_0 中投 0 票，而在集合 S_1 中投 1 票。需要注意的是，S_0 中的节点得到的多数票是 0，而 S_1 中的节点得到的多数票是 1，因此它们的输出不一致。

▶ 3.2.3　Dolev-Strong 共识

1. Dolev-Strong 共识分析

那么，如何设计一个安全的拜占庭广播协议呢？著名的 Dolev-Strong 协议解决了拜占庭广播问题，如算法 3.2 所示。

假设节点编号为 $1,2,\cdots,n$，指定的发送者编号为 1。还假定每个节点 i 都维护着一个集合 extr_i，也称节点的“提取集”，其中包含迄今为止所选择的不同有效比特。简洁起见，将使用符号来表示附有有效签名的信息 b，该签名可根据 $S\subseteq[n]$中节点的公钥进行验证。在下面的协议中，f 表示损坏节点数量的上限。

算法 3.2　Dolev-Strong 协议

(1) 初始时，设置每个节点 i 的提取集合为空集，$\mathrm{extr}_i=\varnothing$。

(2) 第 0 轮：发送者向每个节点发送$\langle b\rangle_1$。　　//符号$\langle b\rangle_S$ 用来表示附有 i 的有效签名的信息 b

对于每条带有来自不同节点(包括发送者)的 r 个签名的消息$\langle \widetilde{b}\rangle_{1,j_1,\cdots,j_{r-1}}$，节点 i 操作为：如果 $\widetilde{b}\notin \mathrm{extr}_i$，则将 $\widetilde{b}$ 添加至 extr_i 中，并向每个人发送$\langle \widetilde{b}\rangle_{1,j_1,\cdots,j_{r-1},i}$。

//注意，这里节点 i 将自己的签名添加至它收到的 r 个签名集合中

(3) 在第 $f+1$ 轮结束时，如果 $|\text{extr}_i|=1$，则节点 i 输出 extr_i 中的比特；否则，节点 i 输出 0。

2. 为什么 Dolev-Strong 协议需要 f 轮

假设所有节点都必须在第 f 轮而不是第 $f+1$ 轮结束时输出数据，则构建的攻击如下。在第 0 轮，腐败发送者向所有诚实节点发送 $\langle 1\rangle_1$。因此，在第 1 轮中，所有诚实节点都会在它们的提取集上添加 1。现在回顾一下，可能有 f 个腐败节点，包括发送者。在第 f 轮开始时，腐败节点会让一个诚实节点 v 接收到比特 0 和所有 f 个签名，但不会将此消息传递给其他每个诚实节点。在这种情况下，v 最终将在其提取集合中获得两个比特，而所有其他诚实节点在第 f 轮中的提取集合中只有 1 个比特。因此，v 将与所有其他诚实节点不一致。

如果多运行一轮算法，即节点在第 $f+1$ 轮结束时输出，则上述攻击不可能发生。直观地说，这是因为标记了 $f+1$ 个签名的比特 b 必须已被至少一个诚实节点（如节点 i）签名。然而，当诚实节点 i 较早地签署了比特 b 时，如在第 $r<f+1$ 轮，它肯定已经向所有其他诚实节点传播了一批关于 b 的 $r+1$ 签名（包括它自己的），因此所有其他诚实节点都会在第 $r+1\leqslant f+1$ 轮开始时收到带有 $r+1$ 签名的比特 b，并将比特 b 添加到它们的提取集中（如果不是更早的话）。

▶ 3.2.4 Dolev-Strong 共识的分析

有了上述直觉，证明 Dolev-Strong 协议同时满足一致性和有效性（如定义 3.1 所示）。

引理 3.1 设 $r\leqslant f$。如果在第 r 轮结束时，某个诚实节点 i 的 extr_i 中有 $\tilde{b}$，那么在第 $r+1$ 轮结束时，每个诚实节点的提取集中都有 $\tilde{b}$。

证明：在第 r 轮结束时，节点 i 在extr_i 中拥有比特 $\tilde{b}$。这个比特 $\tilde{b}$ 一定是在更早的某一轮被添加到 extr_i 中的。假设 $t\leqslant r\leqslant f$ 是比特 $\tilde{b}$ 第一次被添加到 extr_i 的那一轮。根据协议，在第 t 轮中，节点 i 收到的 $\langle\tilde{b}\rangle_{1,j_1,\cdots,j_{t-1}}$ 有 t 个不同的签名，其中包括一个来自发送方的签名。此外，这些签名中没有一个必须来自节点 i 本身，因为如果节点 i 早先签署了 $\tilde{b}$，它就会早先将 $\tilde{b}$ 添加到 extr_i 中。因此，节点 i 一定在第 t 轮向其他节点发送 $\langle\tilde{b}\rangle_{1,j_1,\cdots,j_{t-1}}$。现在，根据我们的同步假设，所有其他节点将会在第 $t+1\leqslant f+1$ 轮收到这条带有 $t+1$ 个不同签名的消息，并因此将 $\tilde{b}$ 在 $t+1$ 轮中添加到其提取的集合中（如果尚未添加的话）。

上面的推论讲到，如果某个诚实节点在 $r<f+1$ 轮中包含某个比特 $\tilde{b}$，那么所有诚实节点都会在紧接着的下一轮中包含相同的比特。为了证明一致性，需要证明，如果某个诚实节点在最后一轮 $f+1$ 中包含一个比特 $\tilde{b}$，那么所有诚实节点一定都在同一轮中包含这个比特。最后一轮，即第 $f+1$ 轮，是魔术发生的地方，在这里可以达成共识。

引理 3.2 如果某个诚实节点 i 在第 $f+1$ 轮结束时在extr_i 中有 $\tilde{b}$，那么在第 $f+1$ 轮结束时，每个诚实节点的提取集中都有 $\tilde{b}$。

证明：考虑以下两种情况：

- 情况 1（节点 i 在 $r<f+1$ 轮中首次将 $\tilde{b}$ 加入其提取集）：根据引理 3.1，一旦 $\tilde{b}$ 在第

r 轮结束时出现在 extr_i 中，那么到第 $r+1 \leqslant f+1$ 轮时，每个诚实节点的提取集中都会有 $\widetilde{b}$。

- 情况 2(节点 i 在第 $f+1$ 轮中首次将 $\widetilde{b}$ 添加到其提取集中)：要出现这种情况，节点 i 必须收到其他节点在第 $f+1$ 轮开始时在 $\widetilde{b}$ 上面的不同签名。由于最多有 f 个节点是腐败的，因此至少有一个诚实的节点必须在更早的一轮 $r<f+1$ 中签署了 $\widetilde{b}$。因此，根据引理 3.1，包括节点 i 在内的每个诚实节点都会在第 $r+1 \leqslant f+1$ 轮结束时将 $\widetilde{b}$ 加入其提取集。

定义 3.2 Dolev-Strong 协议能在最多 $f \leqslant n$ 个损坏节点的情况下实现拜占庭广播。

证明根据引理 3.2，在第 $f+1$ 轮结束时，所有诚实节点必须具有相同的提取集，因此一致性已经实现。有效性的证明留给读者自行完成。

3.3 分布式系统理论


视频讲解

▶ 3.3.1 FLP 不可能性理论

FLP(Fischer-Lynch-Paterson)不可能性理论是分布式领域中一个非常著名的结果，该结果在专业领域被称为“定理”，FLP 给出一个令人吃惊的结论，如定义 3.3 所示。

定义 3.3(FLP 不可能性理论) 在异步通信场景，即使只有一个进程失败，也没有任何算法能保证非失败进程达到一致性。

因为同步通信中的一致性被证明是可以达到的，因此在之前一直有人尝试各种算法解决异步环境的一致性问题，有了 FLP 定理，这样的尝试终于有了答案。FLP 基于如下几点假设。

- **仅可修改一次**：每个进程初始时都记录一个值(0 或 1)。进程可以接收消息、改动该值并发送消息。
- **异步通信**：与同步通信的最大区别是没有时钟、不能时间同步、不能使用超时、不能探测失败、消息可任意延迟、消息可乱序。
- **通信健壮**：只要进程非失败，消息虽会被无限延迟，但最终会被送达；并且消息仅会被送达一次(无重复)。
- **Fail-Stop 模型**：进程失败如同宕机，不再处理任何消息。相对拜占庭模型，不会产生错误消息。
- **失败进程数量**：最多一个进程失败。

在现实中，常常使用 TCP(保证了消息健壮、不重复、不乱序)，每个节点都有时钟同步，纯异步场景相对比较少。但随着智能终端的发展，每个手机会为省电而关机，也会因为不在服务区而离线，这样的适用场景逐渐增多。

1. 理解 FLP 理论

FLP 是分布式计算中最基本的不可解结果之一。FLP 从理论上证明了在纯异步环境下不可能存在一种确定性的共识协议。后续的研究者们为了绕过这个定理，不得不在两个方向上进行妥协：要么加强对网络的假设，要么引入随机源。

FLP 不可能性结果表明，在一个消息传递的异步系统中，至多允许一个进程因崩溃而失败，确定性地解决共识问题是不可能的。换句话说，在由 n 个节点组成、具有无限延迟的系统

中，没有算法可以解决共识问题。要么会有一些执行过程中未能达成一致，要么会有一个执行过程没有终止（无限执行）。

FLP 不可能性结果的关键问题在于，在异步系统中，不可能区分崩溃的进程和只是运行缓慢或在慢速链路上发送消息的进程（只是需要时间到达接收者）。

2. 理解异步网络

在比特币白皮书中，中本聪的协议被明确设计为在有消息延迟的网络中运行，并且确实是在这样的网络（即互联网）上执行的，如上面提到的 FLP 假设。因此，需要分析区块链协议在异步网络这一更实际的环境中达成一致性的程度。

在完全异步的环境中，敌手可以任意延迟信息，一致性就无法得到满足。在这样的环境下，控制了一小部分计算能力的敌手可以简单地延迟来自诚实方的信息，通过足够长的延迟时间，以确保敌手可以提出自己的链（包含它想要的任何交易集），该链比所有诚实参与者持有的链都长，因此它可以让诚实参与者在任何时候切换（即放弃先前挖矿的链）到敌手的链。这样的攻击即使在部分同步的情况下也有效。

▶ 3.3.2 CAP 理论

CAP（Consistency-Availability-Partition）理论由 Eric Brewer 于 2000 年提出，并由 Seth Gilbert 和 Nancy Lynch 在 2002 年进行了形式化描述。CAP 理论进一步具体化了上述针对 FLP 不可能性结果的权衡，在实际系统设计中，帮助研究者理解在特定场景下如何在一致性、可用性和分区容错性之间进行取舍。



图 3.1 CAP 理论的关系

定义 3.4 的 CAP 理论指出，任何一个分布式数据系统在同一时间只能满足以下三个特性中的两个，即一致性、可用性和分区容忍性。接下来，首先定义这些术语，然后探讨它与共识协议的关联。图 3.1 展示了分布式系统的 CAP 定理。

定义 3.4（CAP 定理） 任何一个分布式数据系统在同一时间只能满足以下三个特性中的两个，即一致性、可用性和分区容忍性。

CAP 定理的假设如下。

- **一致性（Consistency）**：一致性意味着数据在分布式系统的所有节点之间应该保持一致，同时连接到分布式系统的客户端应该看到相同的一致数据。这通常通过复制实现。
- **可用性（Availability）**：可用性意味着即使在出现故障的情况下，分布式系统也能响应客户端请求。这是通过使用容错技术来实现的。
- **分区容忍性（Partition Tolerance）**：分区是指两个或多个节点之间的通信链路断开的情况。分布式系统应该能够容忍这种情况并继续正确运行。

网络中的分区几乎不可避免，通信中断迟早会发生。因此，在面对网络分区的不可避免性时，关键的决策点在于在可用性和一致性之间做出选择。这个选择完全取决于具体的应用场景。例如，在金融应用中，为了保持数据的一致性，牺牲一定的可用性可能是更好的选择。相反，在如网页搜索结果等场景中，为了保持服务的可用性，可以牺牲一定程度的数据一致性。值得注意的是，在没有网络分区的情况下，可以同时提供一致性和可用性。但在网络分区发生时，决策的关键就变成了选择优先保证可用性还是一致性。

▶ 3.3.3 蒙特卡洛共识

为了规避确定性异步网络的著名 FLP 不可能性结果，许多共识协议使用随机化。FLP 从理论上证明了在纯异步环境下不可能存在一种确定性的共识协议。后续的研究者们为了绕过这个定理，不得不在两个方向上进行妥协：要么加强对网络的假设，要么引入随机源。因此，共识问题的非确定性重述（或实现）通常涉及拉斯维加斯规则（Las Vegas Rules）：尽管所需的时间可能是无限的，网络必须最终达成共识（概率为 1）。这里，拉斯维加斯规则在随机化算法中的应用，通常称为拉斯维加斯（Las Vegas）算法。具体地，拉斯维加斯算法具有以下特点。

- **正确性保证**：拉斯维加斯算法保证在每次运行时总能给出正确的结果，即使其运行时间可能有所不同。
- **随机化运行时间**：由于斯维加斯算法的随机化性质，运行时间可能会有所变化，但最终的结果一定是正确的。

与斯维加斯算法相对的是蒙特卡洛（Monte Carlo）算法。蒙特卡洛算法使用随机化方法，但它只提供近似正确的答案，而不是保证每次都正确。

然而，对于大规模网络来说，这一要求（执行时间可能无限）过于严格。由于确定性共识至少需要 $O(n^2)$ 消息比特，因此任何使用少于指数为二次消息的随机协议都必须是以非零概率失败的蒙特卡洛协议。

因此，需要引入一个全局安全参数 k，并且只要求共识协议只能以可忽略的概率成功。换言之，该共识协议成功的概率小于 $1/\text{poly}(k)$，即接近零的概率比任何关于 k 的多项式函数的倒数更快。特别地，对任何可信（通信）信道的实例化，即使使用的是理想化的加密原语，也会引入一个与 k 等效的参数。

由于比特币协议使用的信息数量与网络的规模无关，满足了上述要求。如果不存在故障（或错误），那么不同信息的预期数量仅为 $O(k)$。在最糟糕的情况下，当拜占庭故障的数量为 $f=\left\lfloor\frac{n-1}{2}\right\rfloor$（不大于 $\frac{n-1}{2}$ 的最大整数）时，预期的消息复杂度为 $O(kn^2)$。

▶ 3.3.4 中等难度谜题

Aspnes 等建议在初始匿名网络中使用难度适中的谜题来分配（或建立）身份。谜题的难度可以防止计算能力有限的敌手获得较多身份。这与比特币采用的方法基本相同。

一个中等难度的计算谜题是可以被解开的，但必须付出相当大的努力、时间和/或成本，而难度参数是可以调整的。Merkle 在开发公钥密码学时首次使用了谜题。后来，谜题被用于减缓电子邮件垃圾邮件发送者的速度。随后，谜题又被用于多种场合，以挫败假冒者的攻击。其他用途还包括时间延迟加密和微（小额）支付。

计算谜题通常使用随机预言机（即理想的散列函数）来构建，将输入映射到随机输出。随机预言机是单向函数，无法反转。谜题的设计使寻找解的最有效方法是蛮力猜测和检查法。因此，敌手破解谜题的速度受到哈希值计算速度的限制。Aspnes 等所描述的协议依赖进程之间成对认证的通信渠道。比特币协议通过使用非交互式的公开可验证谜题消除了这一要求。进程之间只使用未经验证的广播信息进行通信。

▶ 3.3.5 状态机复制

状态机复制（State Machine Replication，SMR）最初由 Leslie Lamport 等在 1978 年提出，

是一种广泛应用于分布式系统中以实现容错能力的关键技术。状态机通过执行一系列命令来转换系统的当前状态，其中，系统状态的变化是通过状态转换函数来实现的，该函数根据执行的命令结果将系统从当前状态转移到下一个状态。这些命令是确定性的，意味着机器的下一个状态和输出完全由它接收到的输入命令决议。在状态机复制中，每个副本（节点）都是一个状态机，它们通过以确定性的方式执行输入命令来实现状态的转移。副本上的状态以键值对的形式维护，其中命令的执行负责将系统从一个状态转移到下一个状态。下面通过代码 3.1 来描述一个节点上的状态机。

代码 3.1　节点上的状态机算法

```
State: = initial                                    //设置系统的初始状态
Log: = lastcommand                                  //使用最后执行的命令初始化日志
while(true){                                        //循环,持续检查并处理命令
oneventreceivecommand()                             //处理接收命令的事件的函数
   {
   appendtolog(command)                             //将接收到的命令添加到日志中
   output: = statetransition(command,state)         //处理命令并相应更新状态
   sendtoclient(output)                             //将处理过的输出返回给客户端
   }
}
```

在代码 3.1 中，每个副本上的状态机从初始状态开始。当从用户接收到命令时，它将其添加到日志中。之后，通过转换函数执行命令并更新状态并产生输出。这个输出作为响应发送给用户。图 3.2 展示了这个概念。

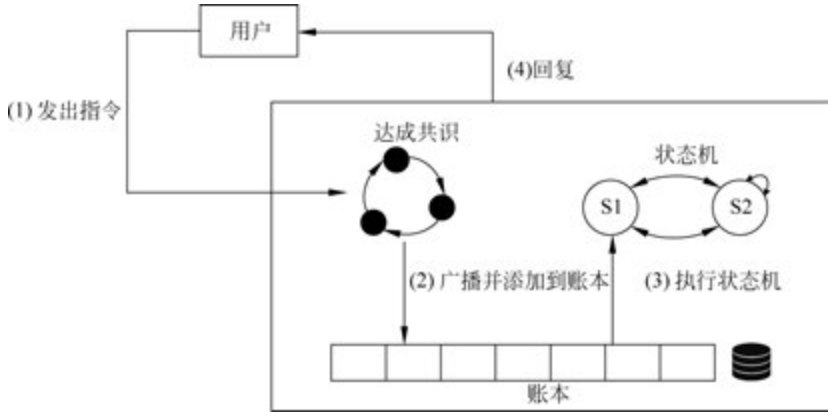

图 3.2　状态机复制

状态机复制的关键思想是，如果系统被建模为一个状态机，那么通过对操作顺序达成一致就能实现副本一致性。如果相同的命令以相同的顺序应用于所有节点，那么就能采用一种通用的方法使所有副本保持一致。然而，挑战在于如何实现命令的全局统一顺序。

为了对操作顺序达成一致，可以使用协议，如拜占庭协议或可靠广播协议（如全序广播）；也可以使用共识算法，如 Paxos 或 PBFT。注意，在全序广播中，每个进程以相同的顺序传递相同的消息。这一特性立即解决了实现操作顺序一致的问题，这是状态机复制背后的核心见解。全序广播确保来自不同客户端的命令以相同的顺序传递。如果命令以相同的顺序传递，它们将以相同的顺序执行，由于状态机是确定性的，所有副本最终会处于相同的状态。

每个副本都是一个状态机，通过执行输入命令确定性地将其状态转换到下一个状态。每个副本上的状态维护为一组(键，值)对。命令的输出从当前状态转换到下一个状态。确定性很重要，因为这确保每个命令执行产生相同的输出。每个副本从相同的初始状态开始。全序广播将相同的命令以全局顺序传递给每个副本，这使得每个副本执行相同的命令序列并转换到相同的状态，从而在每个副本上实现相同的状态。

这一原则也用于区块链，通过某种共识机制实现交易和区块序列的全序，每个节点按照其他副本的顺序执行和存储这些交易和区块序列，如工作量证明的获胜者或领导者所提议的那样。传统协议如实用拜占庭容错(PBFT)和 RAFT 是状态机复制协议。

状态机复制通常用于提高系统性能和容错能力。系统性能可以提高，因为多个副本托管数据副本，并且由于有多个副本可用，因此有更多资源可用。容错能力提高的原因很简单，因为数据在每个副本上都有副本，即使某些副本不可用，系统仍能继续运行并响应客户端查询和更新。

下面描述 SMR 具备的属性。

- **相同的初始状态(Same Initial State)**：副本总是从相同的初始状态开始。它可能只是一个空的数据库。
- **确定性操作(Deterministic Operations)**：所有正确的副本对于相同的输入和状态，确定性地产生相同的输出和状态。
- **协调(Coordination)**：所有诚实的副本以相同的顺序处理相同的命令。协调属性可以通过共识协议和全序广播(Total Order Broadcast)来实现，其中，全序广播是分布式计算中的概念，感兴趣的读者可以阅览相关资料。状态机复制还满足安全性与活性。
- **安全性(Safety)**：所有正确的副本执行相同的命令，又称为**一致性(Agreement)属性**。实现一致性有两种一般方法。可以使用全序广播或共识协议。其中，全序广播适用于命令序列的全局一次性操作，而共识协议适用于持续的、周期性的命令序列
- **活性(Liveness)**：所有正确的命令最终由正确的副本执行。这也称为完成属性。

如图 3.3 所示，可以看到这个过程是如何工作的。

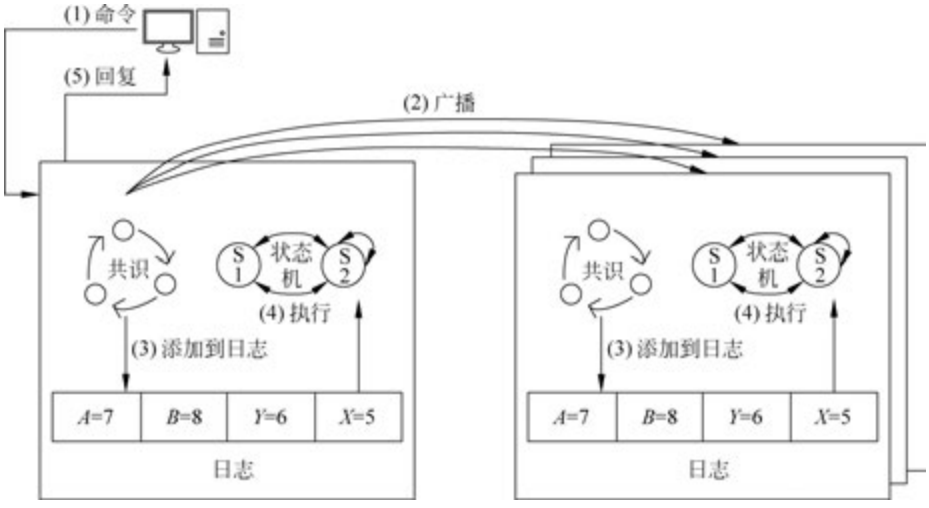

图 3.3 状态机复制构造流程

(1) 客户端向副本 1 提交命令 $x=5$。

(2) 副本 1 将该命令发送给副本 2 和副本 3。

(3) 所有副本将命令附加到它们的日志中。

（4）每个副本上的状态机执行该命令。

（5）副本 1 向客户端回复结果。

副本上的复制日志确保状态机在每个副本上以相同的顺序执行命令。共识机制（图 3.3 左上角）确保对命令顺序达成一致，并因此将其写入日志中。这涉及与其他副本就命令序列达成一致。如果大多数副本正常运行，该复制系统将继续进展。

共识和状态机复制的关系在于分布式共识建立了状态机命令的全局共同顺序，而状态机根据共识（协议）算法确定的全局顺序执行这些命令，从而每个节点（状态机）达到相同的状态。一个容错崩溃的 SMR 需要至少 $2f+1$ 个副本，而一个拜占庭容错的 SMR 需要 $3f+1$ 个副本，其中，f 是故障副本的数量。


视频讲解

3.4 比特币共识

比特币的关键在于多个匿名进程使用单向公共广播进行通信，并且达成共识。接下来，将通过分析证明：比特币协议在这种模型中达成共识的概率几乎为 100%，当拜占庭故障占网络的一半以下时。该协议具有可扩展性，因为其运行时间和消息复杂度均与网络规模无关，而仅取决于故障进程的相对计算能力。然而，这里引入了一个要求，即协议必须容忍任意数量的被动客户端，这些客户端只能接收广播而不能发送消息。

具体地，基于前文提到的蒙特卡洛规则来说明共识的要求：允许有正但可以忽略的概率的分歧，并且运行时间必须是有界的，并基于此开展分析和推导。

▶ 3.4.1 背景与问题

考虑一个大型公共对等网络中的拜占庭共识问题：尽管存在少数故障进程任意偏离协议，网络中的所有进程必须就某个值达成一致。共识是一个强大的工具，使得分布式应用如动态名称分析、证书吊销、可变文件系统和加密货币成为可能。不幸的是，没有准入控制的网络本质上容易受到 Sybil 攻击（女巫攻击）的影响，在这种攻击中，自私或恶意的进程会声称多个虚假身份。这对共识协议是有害的，因为共识协议通常涉及收集多数票。是什么阻止攻击者多次投票？

2008 年，中本聪提出了比特币，核心是一种新颖的拜占庭共识协议，其中，中等难度谜题阻止了计算能力有限的对手获得过多影响力。接下来，在模型中，假设同步处理器和通信的存在，以及一个随机预言机（即理想的哈希函数）的存在。使用蒙特卡洛规则来说明共识的要求：允许有正但可以忽略的概率的分歧，并且运行时间必须是有界的。

▶ 3.4.2 计算模型

考虑网络是由 n 个配置相同的进程 p_i 组成的集合，即 $\{p_1, p_2, \cdots, p_n\}$。一个执行是交替的状态和转换的序列，$\pi_0, s_0, \pi_1, s_1, \cdots, \pi_i, s_i, \cdots$，其中从状态到可用转换集的映射定义了可允许的执行。该映射通常由以下 5 类规则描述。

1. 同步处理器

每个转换都与单调（递增）的时间戳相关联，$r_i \leqslant r_{i+1}$，每个时间戳都是实数时间增量 dr 的整数倍（即 $r = n \cdot \mathrm{dr}, n \in \mathbf{R}$）。每个进程在每轮中都要执行一次 compute 进程，而且每个进程都必须在进入下一轮之前完成该轮动作。假设各进程具有相同的与随机预言机有关的计算能力：每个进程每轮最多可查询一次随机预言机（即调用 oracle）。此外，假设在协议开始时公布一个通用

随机字符串 CRS。这一假设排除了“抢先行动”的敌手。考虑该模型的连续极限为 dr→0。

2. 同步通信

匿名进程使用 broadcast(m)指令进行通信，该指令将消息 m 放入与每个进程相关联的无序邮箱中。当信息被重新接收时，接收进程无法知道信息来自何处。邮箱中的信息可以按任何顺序、在任何时间发送，但有一个最大延迟 Δ。如果 m 在 r 时被广播，那么每个进程都会在 $r+\Delta$ 时或之前执行 receive(m)。不将 receive 进程的访问次数计入预言机查询的总数。

3. 非确定性

特定的执行顺序由一个明确的敌手来决议，即从部分执行到可用转换的映射。进展可以执行指令 coinflip 来随机产生$[0,1)\subset\mathbf{R}$。假设 $H:[0,1)\to[0,1)$是一个随机预言机函数，它将输入均匀地映射到输出。尽管敌手可能不依赖于未来随机事件的结果，敌手拥有全部信息，这可能来自于整个执行历史，包括每个进程的内部状态(进程没有秘密)。因此，敌手定义了可接受执行的概率分布。

4. 拜占庭故障

敌手被赋予对 f 个进程的控制权，这些进程被称为故障(腐败)进程或棋子。敌手是非适应性的，必须在一开始就指定它的棋子。相对于随机预言机，敌手在计算上是受限的；它只能通过它的棋子查询随机预言机，每个故障进程每轮查询一次。

5. 被动用户

除了 n 个进程外，还模拟了一组任意的被动用户，它们代表网络用户。除了不能发送任何信息(它们的 broadcast 指令被替换为无操作)外，它们与进程完全相同。用户指的是被动用户和正确进程的集合(但不包括错误进程)。用户使用一条不可撤销、一次性使用的指令，即 decide(v)，将 v 作为最终值输出。

▶ 3.4.3 蒙特卡洛共识问题陈述

每个正确进程 p_i 最初都有一个输入值 $v_i\in\{0,1\}$。蒙特卡洛共识协议必须满足以下三个要求。

- 终止性：所有用户必须在规定时间内做出决议。
- 一致性：用户必须在相同的值上做出决议(可忽略的概率除外)。
- 全体一致性：决议值必须是输入值之一(概率不可忽略)。可以防止敌手对网络产生过多影响。排除了让每个进程决议一个固定值的“琐碎”协议。

▶ 3.4.4 2f+1 容忍度的比特币协议

$2f+1$ 容忍度的比特币协议指的是在拜占庭容错模型下，系统能够容忍最多 f 个拜占庭故障节点，只要系统中有至少 $2f+1$ 个节点参与，系统就能正确运行并达成共识。

在比特币中，所有权通过公钥密码学确定：网络一致同意公钥与加密货币之间的关联关系。加密货币的交易信息可以通过数字签名的认证从一个公钥转移到另一个公钥对应的地址。因此，比特币容易被理解为一个分布式状态机，其中任何给定时间的状态对应于一个账户余额组成的公开账本。这里主要关注的是网络建立对某一时刻状态的共识的机制。

1. 比特币共识协议

算法 3.3 描述了比特币共识协议，其中，进程通过尝试解决一个中等难度的谜题来为它们选择的值投票。如算法 3.4 所述，这个特定的谜题基于哈希碰撞。当一个正确的进程找到谜

题的解时，它会将其广播到网络。每个谜题的解被计为单个值的一票，每个进程选择那个得到最多票的值（即在该值上进行挖矿）。

算法 3.3　针对进程 p_i 的比特币共识协议

```
1.  初始执行：
2.    Votes ← ∅                                   //初始化投票集为空集
3.    Prefer ←Proposed_i                          //设置此进程首选提议为 Proposed_i
4.  在接收到（Votes',Prefer'）时执行：              //当接收到其他节点的投票和提议时
5.    如果对所有 v∈Votes'验证(v,Prefer') 都成立且满足 //验证收到投票和提议是否有效
6.      |Votes| > |Votes'|，那么                    //如果新的投票集比当前的投票集要大
7.      Votes ← Votes'                            //更新当前的投票集为新的投票集
8.      Prefer ← Prefer'                          //更新当前的首选提议为新的提议
9.  在计算(r)时执行：
10.   nonce ←coinflip                  //通过抛硬币来生成一个随机值 nonce∈[0,1)⊂R
11. 如果验证（nonce,Prefer)成功，那么               //如果验证 nonce 和当前首选提议成功
12.   广播（Votes∪{nonce},Prefer)
                  //将 nonce 添加到投票集中，并广播更新后的投票集和首选提议
13. 如果 r≥r̂，则决议 Prefer                        //最终决议采用当前的首选提议
```

2. 比特币的中等难度谜题

算法 3.4　比特币的中等难度谜题

```
1.  给定参数 μ,n 和 dr。令 p := dr/(nμ)。   //假设 CRS 是一个在协议开始时公布的公共随机字符串。最
                                            //后，假设 H:[0,1)→[0,1)是一个随机预言机函数，其输出
                                            //均匀分布
2.  进程 oracle(x)                          //定义进程 oracle 的输入与输出
3.    返回 H(x)                             //oracle 函数，返回输入 x 的哈希值
4.  进程 verify(Vote,Prefer)                //定义进程 verify 的输入与输出
5.    hash ← oracle (CRS||Prefer||Vote)
                                            //通过 oracle 函数计算 CRS、首选项和投票和的哈希值
6.    如果 hash/{0,1}^κ ≤ p，则返回 True
                        //如果计算得到的哈希值小于或等于挖矿难度值 p，则验证成功
```

解决谜题的策略几乎等同于随机猜测，敌手被迫参与一场它无法赢得的竞赛。接下来，基于对泊松过程的分析，对中本聪提出的共识协议进行形式化的分析。

假设单位时间等于消息延迟界限，因此时间单位为 $\Delta=1$。由于没有找到比随机的查询预言机更有效的谜题解决方案，则重点考虑敌手这样做（查询随机预言机）的最坏情况。因此 n 个进程中的每个都在执行一个独立同分布的伯努利实验，成功概率为 $p=\dfrac{\mathrm{dr}}{n\mu}$。随着 dr 接近零，整个网络根据泊松到达过程产生谜题解决方案，每一对连续谜题解的时间间隔是一个独立同分布的指数随机变量（如接下来要介绍的泊松分布），其尺度参数为 μ。

3. 泊松分布

定义 3.5（泊松分布）　*若一个随机变量 X 的概率分布为*

$$P\{X=k\}=\mathrm{e}^{-\lambda}\frac{\lambda^k}{k!},\quad k=0,1,2,\cdots \tag{3.1}$$

则称 X 服从参数为 λ 的泊松分布，记为

$$X \sim P(\lambda) \quad 或 \quad X \sim \pi(\lambda) \tag{3.2}$$

其中，k 代表的是变量的值，且是自然数。

泊松分布的图形特征如图 3.4 所示。另外，泊松分布与二项分布（在后文介绍）近似。简单来说，泊松分布用于描述在固定时间段内或空间范围内某个事件发生的次数，通常适用于事件发生的次数具有稀疏性和独立性的情况。

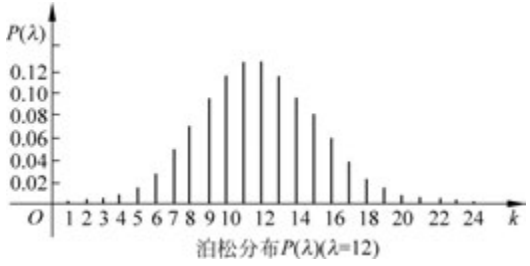

图 3.4 泊松分布示意图

进一步地，令 A_x 表示找到第 x 个谜题的解时间。令随机变量 $M_{r,i}$ 代表进程 p_i 在时间 r 关联到其选择值的投票数（即 $|\mathrm{Votes}|$）。令 M_r 代表这些变量中正确进程的最小值。如果 $M_{\hat{r}} \geqslant x$，则算法 3.3 的第 13 行意味着所有用户决议了与 x 或更多票关联的值。如果 $A_{2x} > \hat{r}$，则少于 $2x$ 票已经被投出，因此所有用户必须已经决议了一个唯一的值。

直接描述 M_r（变量中正确进程的最小值）是困难的，因为它取决于特定的敌手。相反，通过耦合可以进行论证。构建一个与 M 相关的随机过程 B，以便精确描述 B。设 B 是 $n-f$ 个正确进程的一个子集。由于泊松过程是无限可分的，子集 B 可以根据一个泊松到达过程找到谜题解决方案，其预期到达时间间隔 $\mu_B = \mu\delta/2$，其中，δ 满足 $\delta = 2(n-f)/n > 1$（快于总速率的一半）。

定义 B 为交替阶段的序列：①挖矿和②延迟。在挖矿阶段，等待 B 中的一个过程找到谜题的解。当找到解决方案时，切换到延迟阶段并等待一个恒定时间 $\Delta = 1$。在延迟期间找到谜题的被忽略不计。让 B_x 表示 B 完成每个阶段 x 轮周期的时间。现在有了 B 和 M 之间的一个有用的关系，如引理 3.3 所示。

引理 3.3 已知 B_x 和 M_r，那么对于任意 $r \geqslant 0$ 和 $x \geqslant 0$，则 $B_x = r$ 意味着 $M_r \geqslant x$。其中，随机过程 B_x 表示 B 完成每个阶段 x 轮周期的时间。M_r 代表多个进程变量 p_i 在时间 r 关联到其选择值的投票数，其中正确进程的最小值。

证明：假设协议开始于时间 $r = 0$，因此 $B_0 = 0$，$M_0 = 0$。为了便于归纳，假设 $B_x = r$，$M_r \geqslant x$。现在，假设 $r' \geqslant r$ 是正确进程找到下一个谜题答案的时间，因此有 $B_{x+1} = r' + \Delta$。由于 Δ 是最大的信息延迟，那么从算法 3.3 的第 10、11 行和第 5～7 行可知，到 $r' + \Delta$ 时，每个正确的进程都会将某个值与至少 $M_{r'} + 1$ 张票联系起来。因此，$M_{r'+\Delta} \geqslant M_{r'} + 1 \geqslant x + 1$，根据归纳法，该定理对所有 r 和 x 都成立。

现在，构建参数 μ，p，x 和 $\hat{r}$ 的值，从而可以以极大概率得出 $A_{2x} > r > B_x$。正确的进程比一半的网络具有 δ-优势。将这一优势分成三份：两份三分之一，分别用于 A_{2x} 和 B_x 的偏差边界，另外三分之一用于考虑 B 所承受的延迟阶段。目标是使谜题足够困难，以至于平均而言，正确的进程会在找到两个解之前找到并传播一个解。

首先，用 δ 和安全参数 k 来定义 x：

$$x := k / D_{\mathrm{KL}}(\delta^{(\frac{1}{3})} \parallel 1) \tag{3.3}$$

其中，$D_{\mathrm{KL}}(\mu_P \parallel \mu_Q)$是带有标度参数 μ_P 和 μ_Q 的指数分布 P 和 Q 之间的相对熵（Relative Entropy，又称为 KL-发散或 KL-分歧）。KL-发散与区分两个概率分布所需的比特数有关。稍后需要的两个实例公式如下。

$$D_{\mathrm{KL}}(\delta^{(\frac{1}{3})} \parallel 1) = \delta^{(\frac{1}{3})} - 1 - \log\delta^{(\frac{1}{3})} \tag{3.4}$$

$$D_{\mathrm{KL}}(\delta^{-(\frac{1}{3})} \parallel 1) = \delta^{-(\frac{1}{3})} - 1 - \log\delta^{-(\frac{1}{3})} \tag{3.5}$$

接着，求解 $\hat{r}$：

$$\frac{(\hat{r} - x)}{x\mu_B} = \frac{2x\mu}{\hat{r}} = \delta^{(\frac{1}{3})} \tag{3.6}$$

使其结果是一个简化后的二次方程：

$$\hat{r} := \frac{4x}{4 - \delta^{(\frac{5}{3})}} \tag{3.7}$$

接下来，将给出一个关于这些参数达标的推理。

4. 参数达标引理

引理 3.4（参数达标引理） 设 $k>0$ 为全局安全参数，参数 μ、$\hat{r}$ 和 x 的定义如式(3.3)、式(3.6)和式(3.7)所示。如上所述，设 A 为规模参数为 μ 的泊松到达过程；A_{2x} 为 $2x^{\mathrm{th}}$ 到达的时间。假设 B 是一个带有交替阶段的过程：①泊松到达（规模参数 $\frac{\delta}{2}\mu$）；②恒定延迟时间（规模参数 $\Delta=1$）；B_x 是 B 通过两个交替阶段并完成 x 轮的时间点。那么，$A_{2x} > \hat{r} > B_x$ 成立（可忽略概率除外）。

证明：通过基于 Chernoff（切尔诺夫）界限，可推导出想要的结果。基于切尔诺夫界限（Chernoff bounds）的详细证明留给读者自行完成或参阅文献[9]的附录。

定义 3.6 在一个有匿名同步进程和拜占庭最小容错中，比特币的共识协议（算法 3.3）能够达成共识（除非以可忽略的概率），网络中，如果拜占庭故障进程占到网络的一半（即 $n \leqslant 2f$），那么没有任何协议能以高概率达成共识。

证明：三个要求都得到了满足。

- **终止性**：所有用户必须在有限时间内做出决议。根据算法 3.3 的第 13 行，用户在时间 $\hat{r}$ 做出决议。
- **一致性**：用户必须决议相同的值（除非以极小概率）。根据引理 3.3 和引理 3.4，至少有 $1-2e^{-k}$ 的概率，所有用户都会决议得到 x 票的值，尽管网络总共产生的谜题解少于 $2x$ 个。因此，获得 x 票的值是唯一的。
- **全体一致性**：决议的值必须是输入值之一（以不可忽略的概率）。

假设在协议开始时将计数器初始化为 0。如果一个正确的进程在总共找到两个解之前找到并传播了一个解（即 $B_1 < A_2$），那么在时间 B_1 增加计数器；否则，在时间 A_2 减少计数器。在任一情况下，在修改计数器后，重复实验，共计 $2x+1$ 次。计数器的值是一个简单的随机游走。根据式(3.6)和 A 和 B 的定义，有

$$E[B_1] = \mu_B + 1 < 2\mu = E[A_2] \tag{3.8}$$

因此，随机游走是正偏的。由于进行了奇数次步骤，最终值为正的概率大于 1/2。如果最终值为正，那么根据选票定理（用于计算在选举过程中一个候选人始终领先的概率）知道，计数

器从未回到零的概率为$\frac{1}{2x+1}$。当这种情况发生时，正确的进程会在第一步就收敛到它们的其中一个输入，并最终决议那个值。

定义 3.7（选票定理） 如果候选人 A 比候选人 B 获得更多票，可以预测 A 在整个计票过程中始终保持领先的概率。这种概率计算涉及两个候选人的票数差异，并通过数学公式$(p-q)/(p+q)$来表达，其中，p 和 q 分别代表两个候选人的票数。

▶ 3.4.5 $2f+1$ 容忍度下限

该部分通过分析证明，$2f+1$ 是最优的拜占庭容忍度。在模型中，假设用户是被动的，这导致了一个基于不可区分性的简单不可能性结论。对于使用随机预言机的协议，无论谜题的性质如何，$2f+1$ 是最优的容忍度。如果敌手控制了一半的网络，则可以通过查询相同次数的随机预言机，完美模拟正确的进程。因此，被动用户必须区分正确的过程和冒名顶替者。

定义 3.8 在一个有被动用户的匿名网络中，如果拜占庭故障过程占了网络的一半（$n \leqslant 2f$），则没有协议能够以高概率达成共识。

证明：假设相反，存在这样一个协议。将进程分为两组 P 和 Q。P 和 Q 之间不传递任何消息，但被动用户接收到两组的所有广播。考虑以下两种情况。

- **情况 1**：正确的进程在 P 中，而腐败进程在 Q 中。所有进程接收输入 0。腐败进程模拟正确的协议，但输入为 1。由于全体一致性，被动用户决议为 0 的概率不可忽视。
- **情况 2**：正确的过程在 Q 中，而腐败进程在 P 中。所有过程接收输入 1。腐败进程模拟正确的协议，但输入为 0。全体一致性，正确的过程共识为 0。

在任何一种情况下，腐败进程都仿佛被给予了相反的输入一样模拟正确的协议。然而，由于两种情景无法区分，用户最好的做法只能是随机选择一个比特，并以 1/2 的概率反对。

▶ 3.4.6 运行时间与消息复杂度

分析比特币协议的渐近复杂性。根据引理 3.4，正确进程发送的不同消息数量被参数 x 界定，这个参数在尺寸的优势参数 δ 下定义（见式(3.3)）。每条消息包含一整个投票集合，但大多数消息是重复的；这里只计算不同的元素。这一界限与网络的总大小 n 和消息延迟界限无关。因为 n 是离散过程的数量，假设敌手控制的过程少于一半，最差可容忍的参数是 $f=\left\lfloor \frac{n-1}{2} \right\rfloor$。因为参数 $\delta=\frac{n-f}{n}$，可以计算当 $n \to \infty$（n 趋于无限大）时，消息的渐近数量是 $x=O(n^2k)$。同样，由于运行时间 $\hat{r}$ 以时间单位 Δ 计算，可以发现运行时间 $\hat{r}$ 是关于 $O(\Delta n^2 k)$ 渐近的。

前文解释了模型要求被动用户仅通过观察广播消息来区分正确的网络和计算能力受限的冒名顶替者（敌手）。因此，可以将拜占庭共识问题与统计测试相关联。在每个瞬间，每个进程都更倾向于选择最有可能被大多数进程选择的值。对于定义的特定谜题，唯一有区分度的信息是发现的不同谜题的解的数量。考虑每条消息获得的确定性位数$\left(\text{即}\frac{k}{x}\right)$。根据式(3.3)，这等于两个指数分布之间的 KL-分歧，其中，参数 δ 是各自尺度参数的比率。由于 KL-分歧是从两个分布中的一个样本获得的信息量的度量，共识协议至少对于基于伯努利实验的谜题来说是理论最优的。

小结

本章对共识有关的基础与理论知识进行了系统梳理。通过经典的拜占庭将军问题，结合Dolev-Strong协议，逐步过渡到分布式公式理论，以及本章核心内容：比特币共识协议。通过上述内容安排，为读者呈现了一个直观和清晰的共识协议分析理念，为读者开展进一步的区块链分析打下基础。本章的分析结论是：中本聪共识协议至少对于基于伯努利实验的谜题来说是理论最优的，即 $2f+1$ 是最优的拜占庭容忍度。

习题

1. 简述拜占庭将军问题的规则和胜利条件，不要遗漏任何因素。

2. 同步规则是本章讨论共识协议的基础，分析同步网络，并通过逆向思维，尝试解释半同步网络以及异步网络。

3. Dolev-Strong协议的设计目的是什么？Dolev-Strong协议有什么缺陷？

4. 试分析Dolev-Strong协议所满足的共识性质。首先简述Dolev-Strong协议，然后对共识协议要求的一致性和有效性进行分析。不要求一步分析到位，但需要读者通过分析来锻炼安全思维和分析问题的能力。

5. 蒙特卡洛共识的特点是什么？

6. 使用状态机复制的概念来描述比特币区块链达成共识的过程。

7. 试解释 $2f+1$ 容忍度的概念，并分析比特币协议是如何实现 $2f+1$ 容忍度的？

8. 试通过蒙特卡洛共识对比特币共识进行建模和分析，不要求一步分析到位，但需要读者通过分析来锻炼安全思维和分析问题的能力。