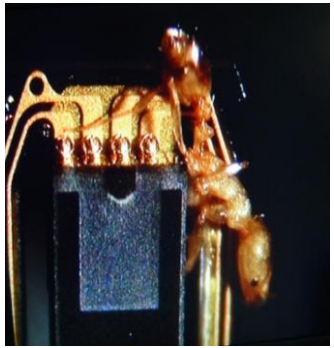
联盟链共识算法简介

范磊

上海交通大学

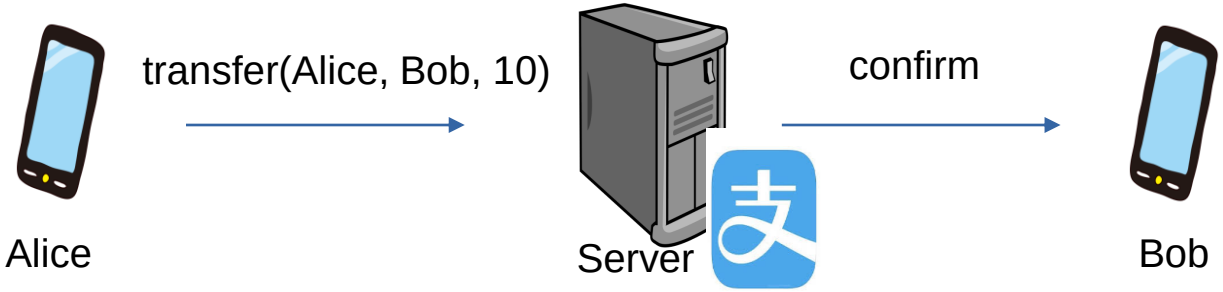
计算机并不（足够）可靠


:(
你的电脑遇到问题，需要重新启动。 我们将为你重新启动。
有关此问题的详细信息和可能的解决方法，请访问 https://www.windows.com/stopcode
MACHINE CHECK EXCEPTION
https://blog.csdn.net/qq_40907977



考虑关键应用

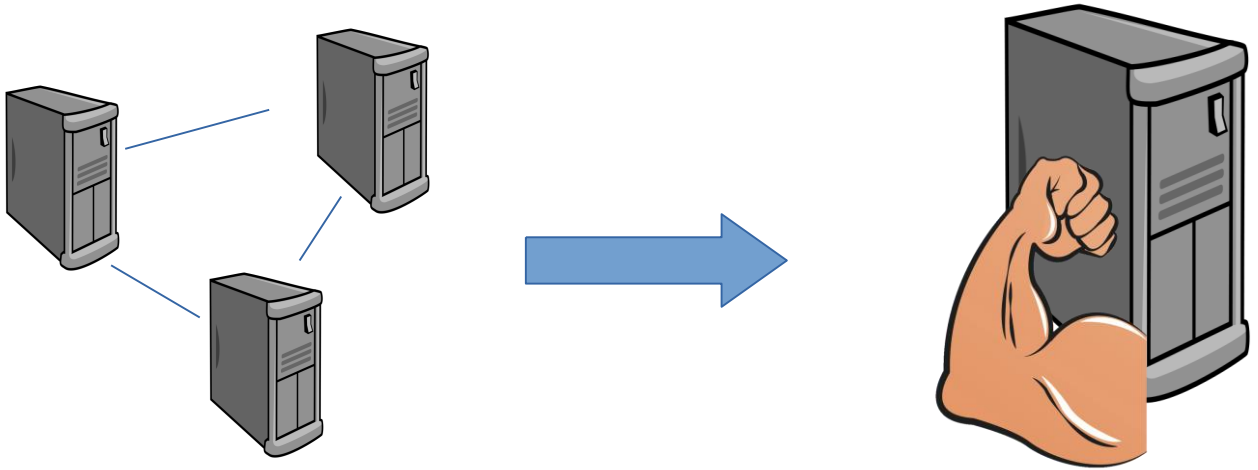
- Alice给Bob转10块钱
  - Alice: 100 - 10 = 90　　Bob: 0 + 10 = 10



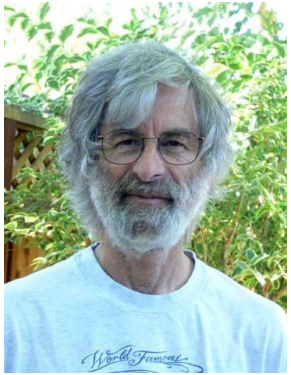
- 如果服务器出现故障 ...

如何应对?

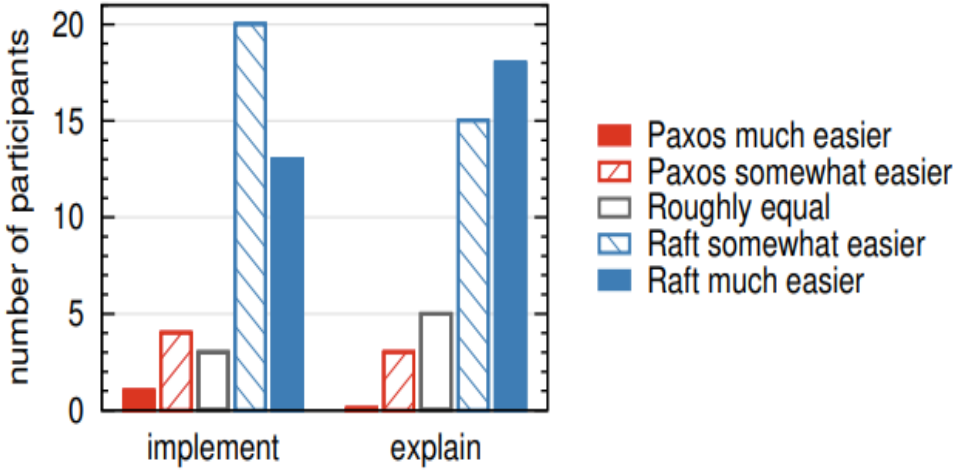
- 冗余策略
- 通过协同一组物理服务器，在逻辑上模拟一个不会出错的服务器
- 服务器 = 结点 或者 process



典型的共识协议



Leslie Lamport
2013 Turing award winner



Paxos → **Raft**

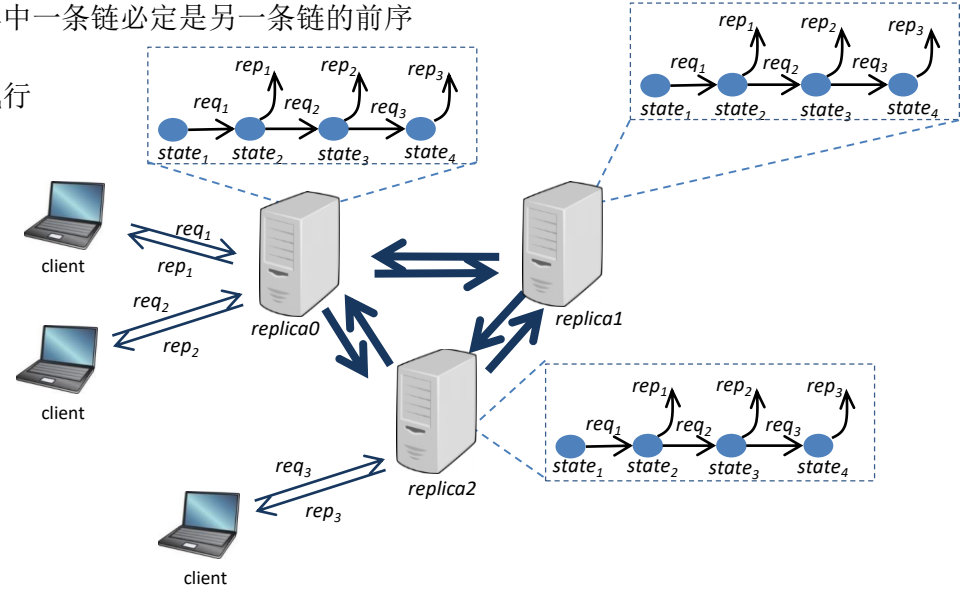
Leslie Lamport.
The Part-Time Parliament. ACM TOCS'98.

Diego Ongaro and John Ousterhout.
In Search of an Understandable Consensus Algorithm. Usenix ATC'14.

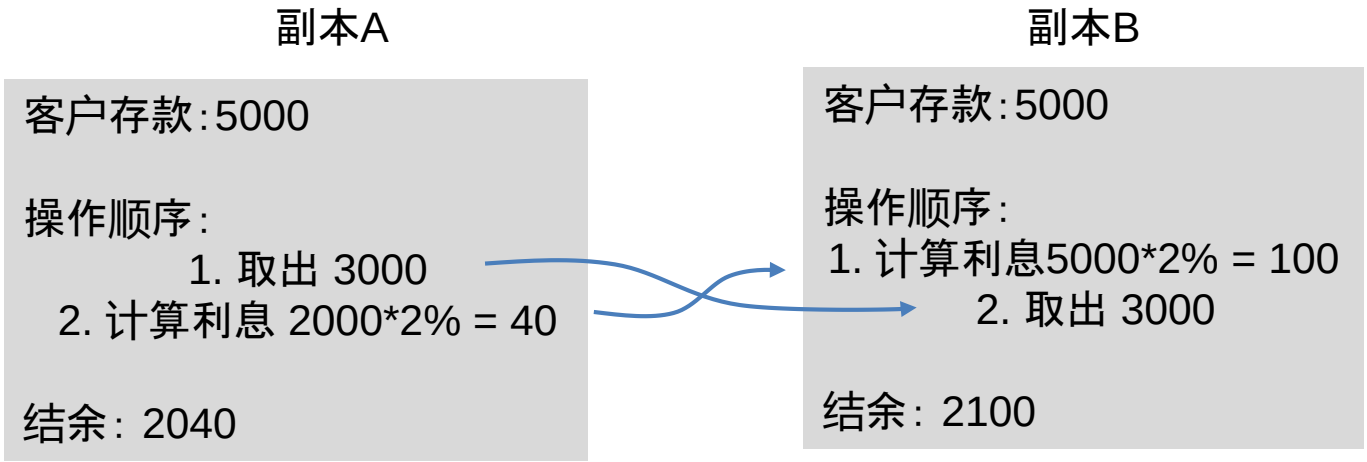
区块链 ≈ 状态机复制协议

- Request: <Request, req>
- Indication: <Response, res>

- 状态机复制协议模拟一台“永不犯错”的中心服务器，满足：
  - ✓ (安全性, safety):
    - ➢ 所有状态机按照相同顺序执行请求，并得到相同的输出和新的状态
    - ➢ 区块链：任意两个正确节点维护的链，其中一条链必定是另一条链的前序
  - ✓ (活性, liveness)
    - ➢ 所有请求最终都会被所有正确的状态机执行
    - ➢ 区块链：所有合法请求最终都能够上链



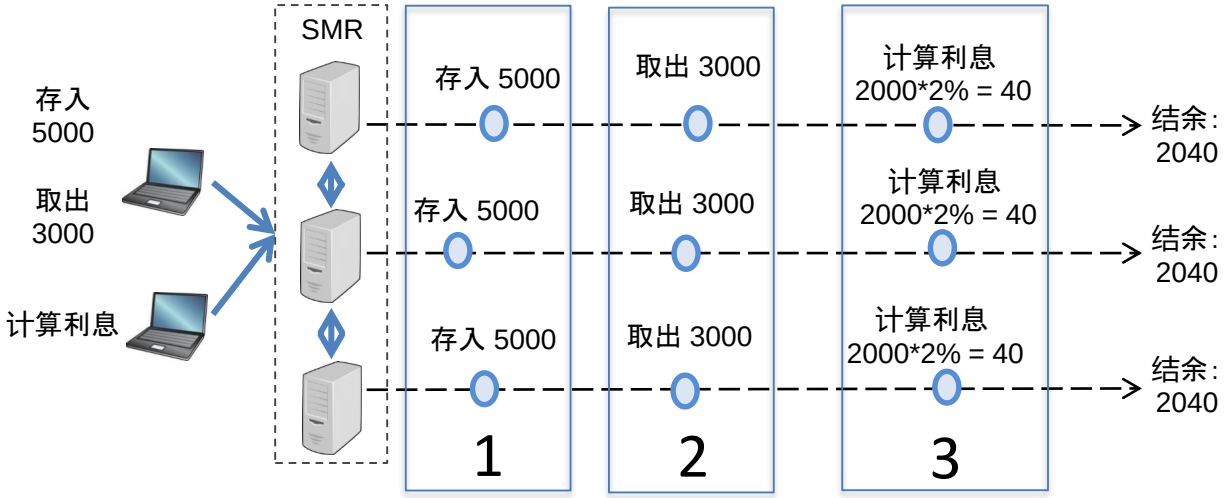
多副本的一致性问题



副本间状态不一致！

状态机复制协议的优势

- 一种通用的解决方案
  - ✓ 或者说，实现一个正确、高可用的日志系统



共识问题

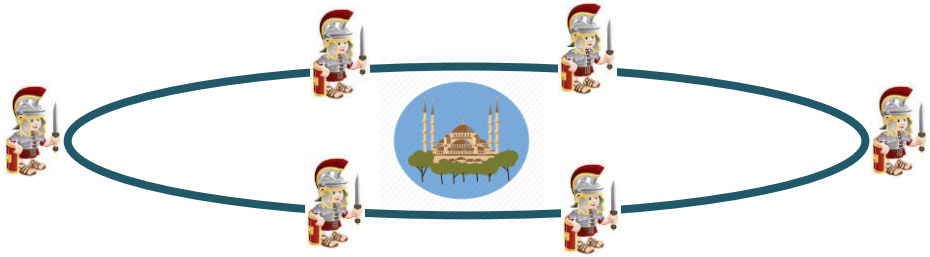
- 对一项提议达成一致：所有结点都可提议，最终会决定一个值
- Request: <propose, v>
- Indication: <decide, v' >

安全性
Agreement：所有（正确）节点决定同一个值
Integrity：达成共识的值必定是由某个节点提议的；如果所有（正确）节点都提议某个值，共识结果为该值

活性
Termination：　所有正确结点最终都决定一个值

最简单的共识问题：01共识

- 拜占庭将军问题：明天是否进攻拜占庭？（0：不进攻　1：进攻）



状态机复制协议的安全假设

- 参与状态维护的节点集合已知
- 假设最多有t个结点可能出错，总共需要部署多少结点？
  - ✓ N = F(t)
  - ✓ N ≥ t+1

- (安全性，safety):所有状态机按照相同顺序执行请求，得到相同的输出和新的状态
- (活性，liveness):所有请求最终都会被所有正确的状态机执行（上链）

结点错误

- **简单错误**
  - ✓ Crash Fault-Tolerance (CFT)
  - ✓ 结点停止运行
  - ✓ 丢失本地数据和状态

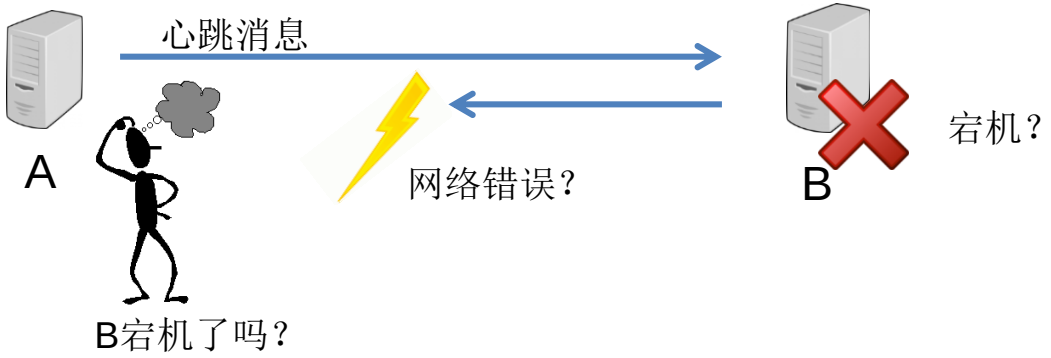
- **拜占庭错误**
  - ✓ Byzantine Fault-Tolerance
  - ✓ 结点可产生任意错误，错误行为不可控
    - ➢ 位跳变，数据崩溃，软硬件错误，配置错误和操作错误等
  - ✓ 极端情况：错误结点协同发起攻击
  - ✓ 一般假设拜占庭节点计算资源有限，不违反密码学工具的安全假设

X

网络错误

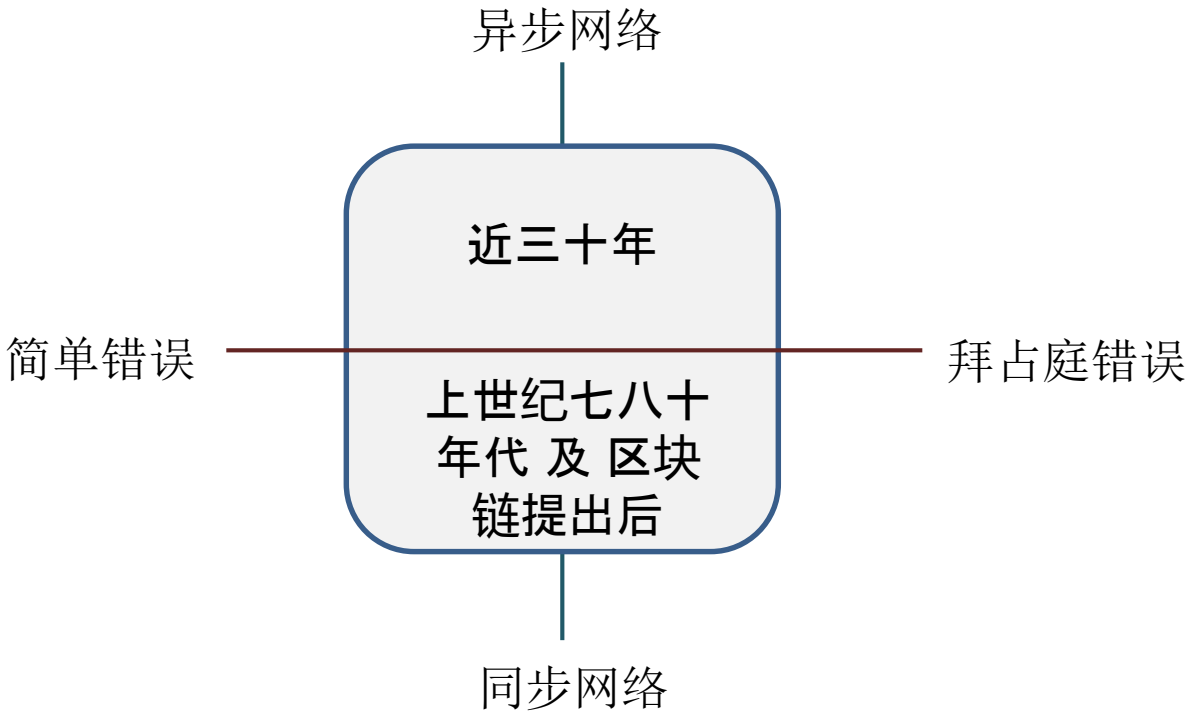
- 网络是结点间的通信通道
- 网络拥塞、网络设备故障导致的丢包和延迟异常
- 同步网络：网络最大延迟Δ已知
- 异步网络：无法给定最大延迟Δ ，保证正确结点在Δ内完成通信

- 分布式系统：结点之间的心跳机制不可靠



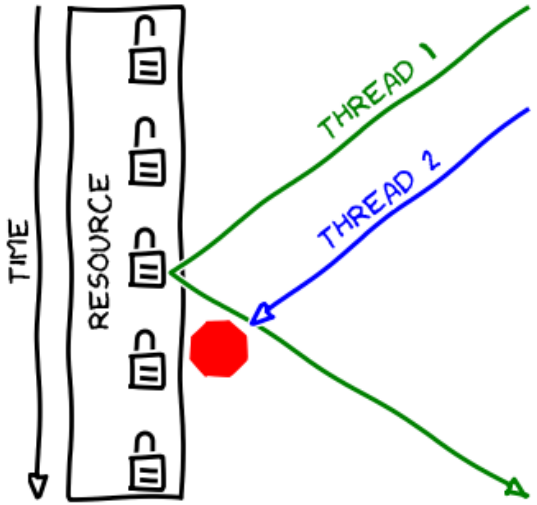
Y

共识算法错误模型的四象限



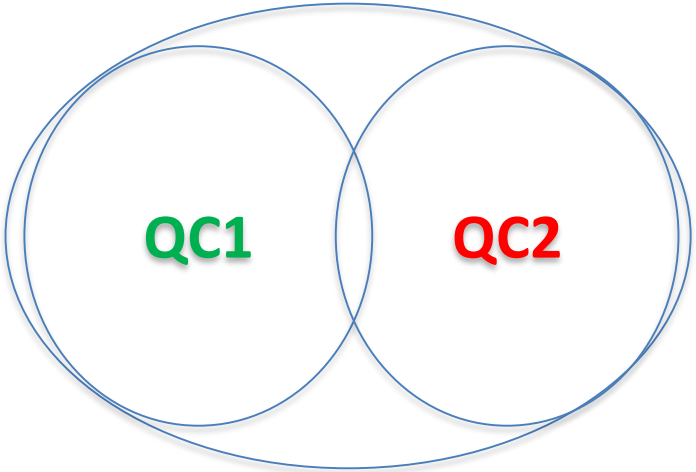
如何保证数据的安全性和活性?

- 传统的中心化系统
- 关键资源加锁
  - ✓ 多线程程序
  - ✓ 数据库并发访问

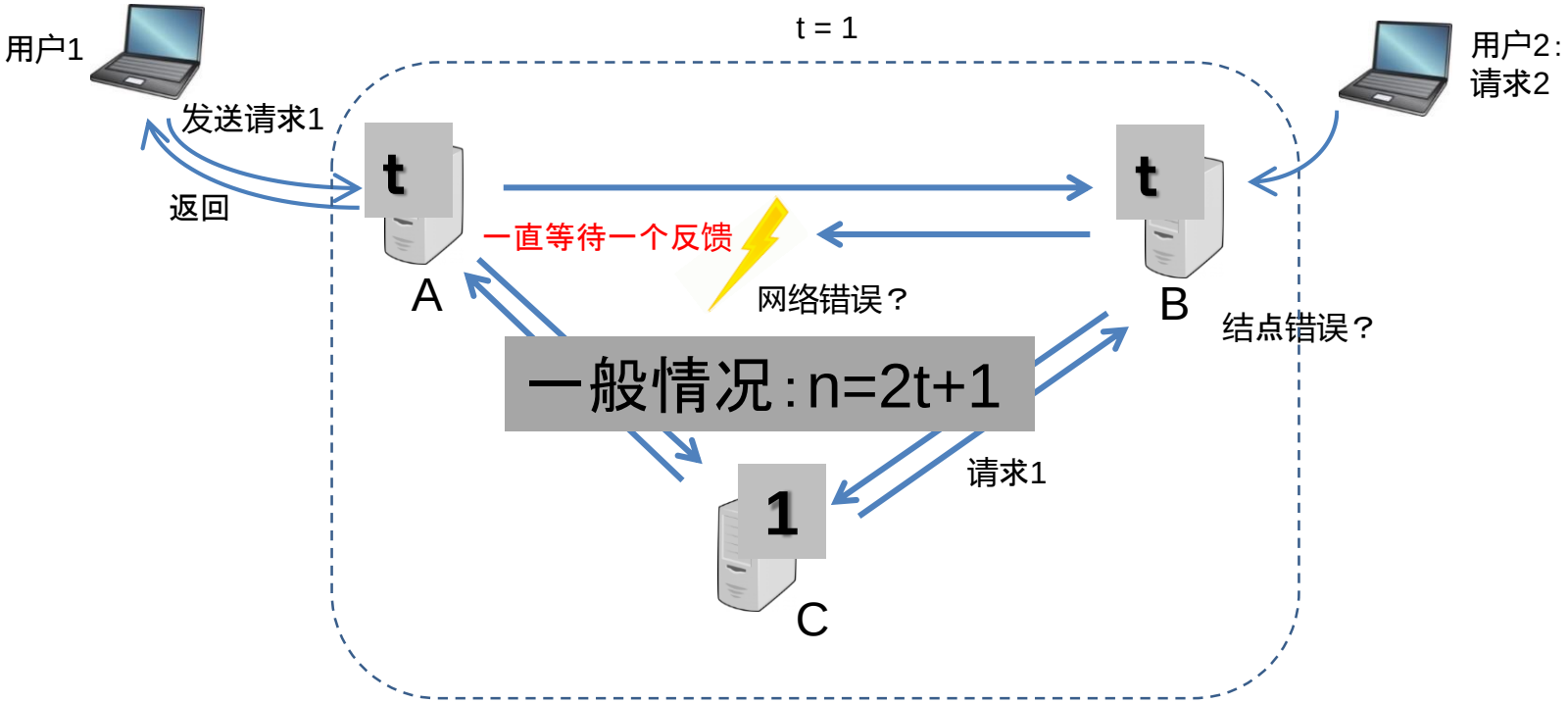


如何保证数据的安全性和活性?

- 去中心化系统
- 参与者投票机制
  - ✓ 包含了对一个提议超过(N+f)/2的投票集合称为法定人数证明(Quorum Certificate，QC)。
  - ✓ 其中N是参与者总数量，f为拜占庭节点数量

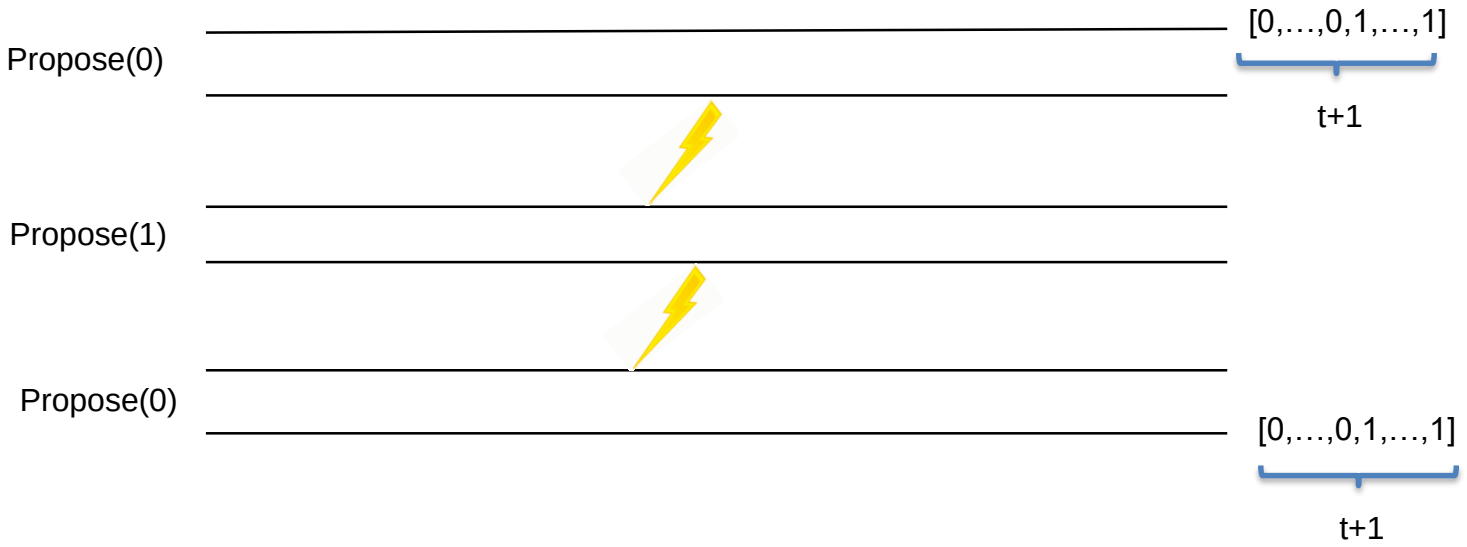


保证安全性与活性：Quorum机制，少数服从多数



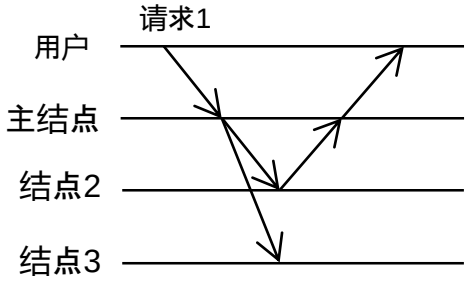
问题还没结束

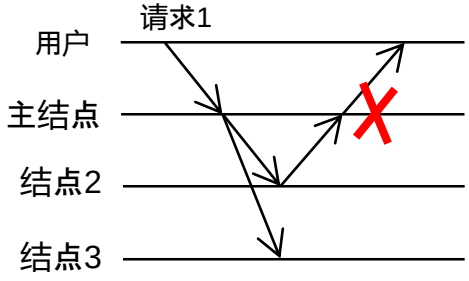
- 如果多个结点提议不同的请求，最终应该听谁的？



主流共识算法的思路：以Paxos/Synod为例

- 选出一个临时的主节点（leader, primary）
- 主节点负责提议请求，并发送到其他结点
- 在收到t个确认后，可确定共识的结果（Quorum机制）



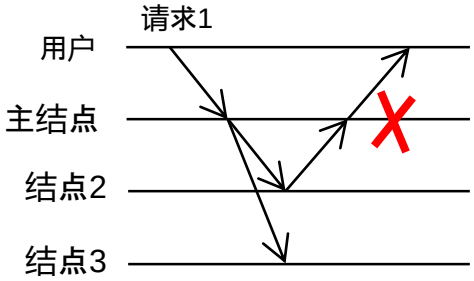


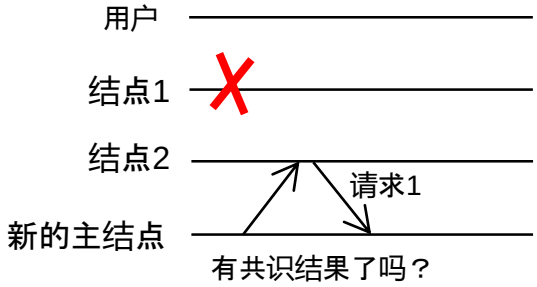
中心结点选举

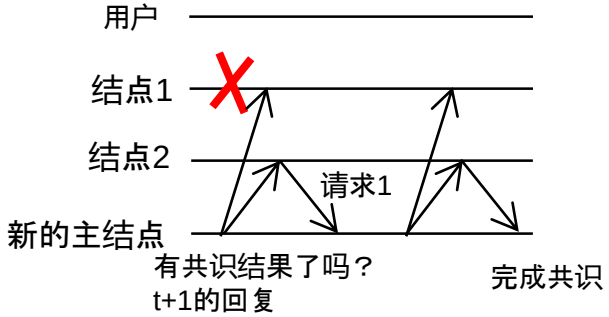
- 当主节点错误时，
  - ✓ 选出新的主结点
  - ✓ 新的主节点需要确定系统当前的状态（是否有正在进行的共识）
  - ✓ 主结点从新的状态开始处理请求

共识的关键问题

- 避免新的主节点遗忘过去的共识结果







Quorum机制

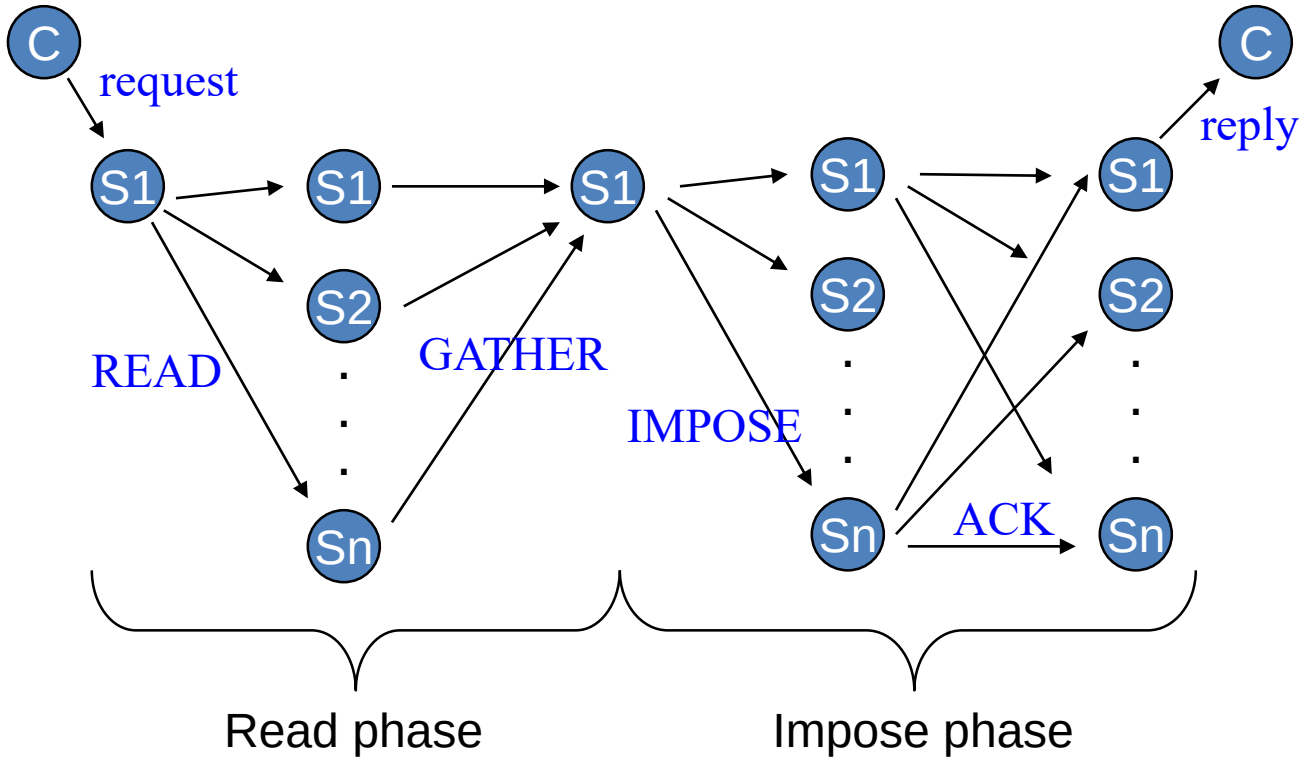
Paxos/Synod

- 基于轮次的算法（round-based）
  - ✓ 在每一轮r，r mod n作为主节点
  - ✓ r = 0，leader = 0；r = 1，leader = 1

- 主节点首先通过一次交互确定以前轮次的共识结果

- 基于以前轮次的共识结果，开始新一轮的共识

消息序

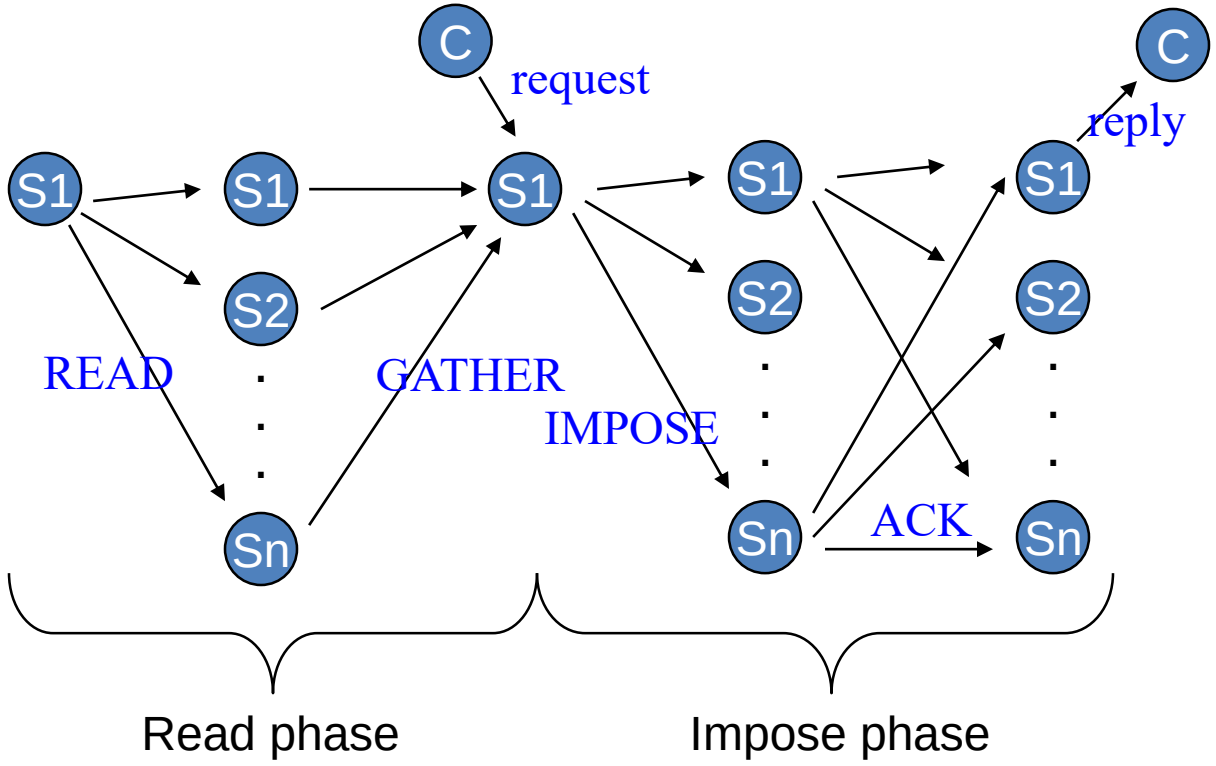

C
request
S1
S1
S1
S2
Sn
READ
GATHER
S1
S2
Sn
IMPOSE
ACK
S1
S2
Sn
C
reply
Read phase
Impose phase

复杂度

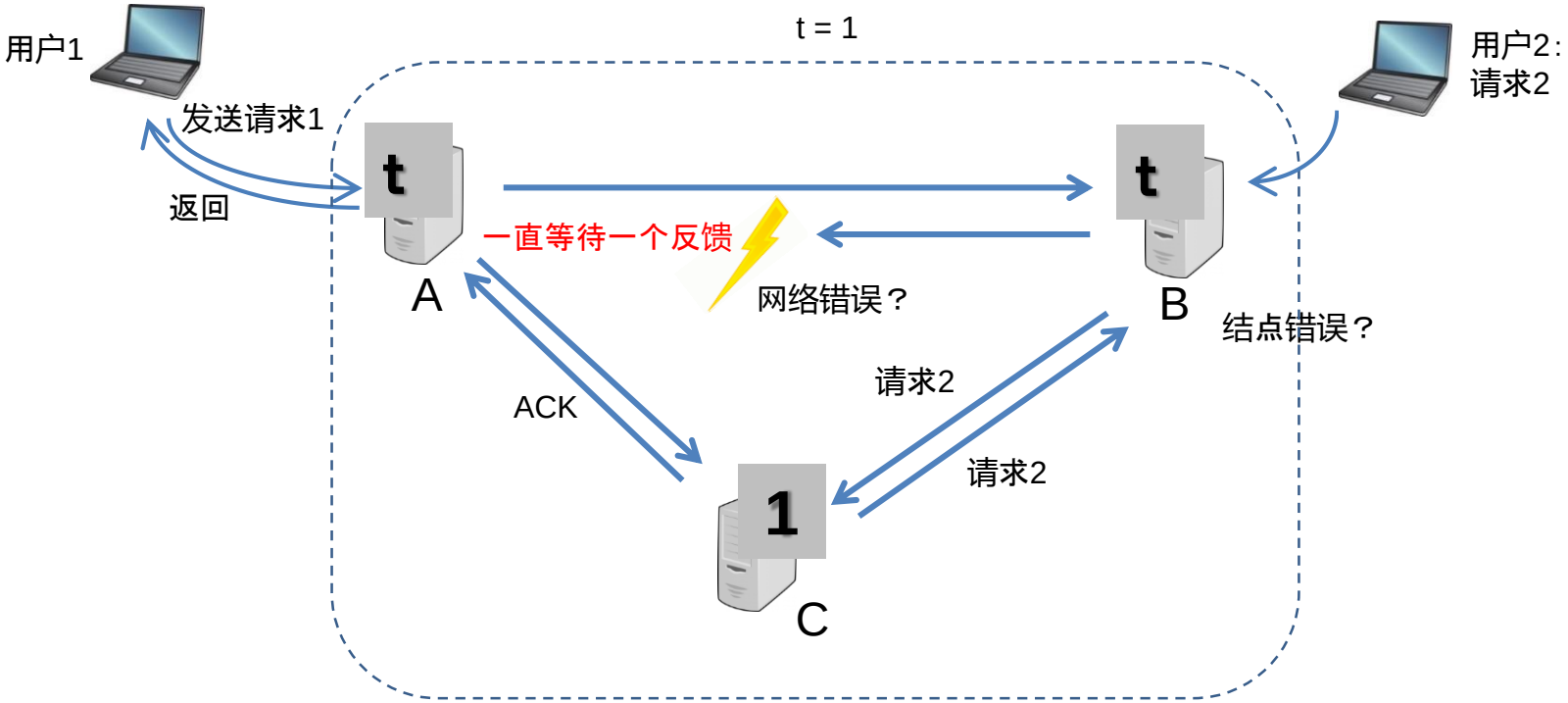
- 消息轮次：4轮交互
- 消息复杂度$O(n^2)$ 或者 $O(n)$

- 优化：第一轮的共识过程可以省略READ步骤
  - ✓确定没有任何值完成共识
  - ✓消息轮次：2轮交互

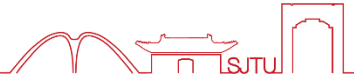
消息序： 优化的情况


C
request
C
reply
S1
S1
S1
S1
S1
S2
S2
S2
READ
GATHER
IMPOSE
Sn
Sn
ACK
Sn
Read phase
Impose phase

新问题——如果存在拜占庭节点



拜占庭敌手的数量上限



- 在分布式系统中如果有N个节点，其中存在f个拜占庭节点，则#(QC)>(N+f)/2

- 在分布式系统中如果有N个节点，其中存在f个拜占庭节点，如果共识协议具有响应性（Responsiveness），则N>=3f+1

PBFT算法

系统模型

网络环境：半同步网络
异常：丢失、延迟、重复或乱序
假设：节点的失效是独立发生的

密码技术：

公钥签名、消息验证编码、无碰撞哈希函数生成的消息摘要
防止欺骗攻击和重播攻击、检测被破坏的消息

对异常的假设：

能够操纵多个失效节点、延迟通讯、延迟正常节点
不能无限期延迟正确节点，不能破解加密算法。

PBFT算法

算法的目标

确定性的副本复制服务（n个节点）

操作：

读写、任意确定性的计算

失效节点数 f：n>3f

f个副本失效

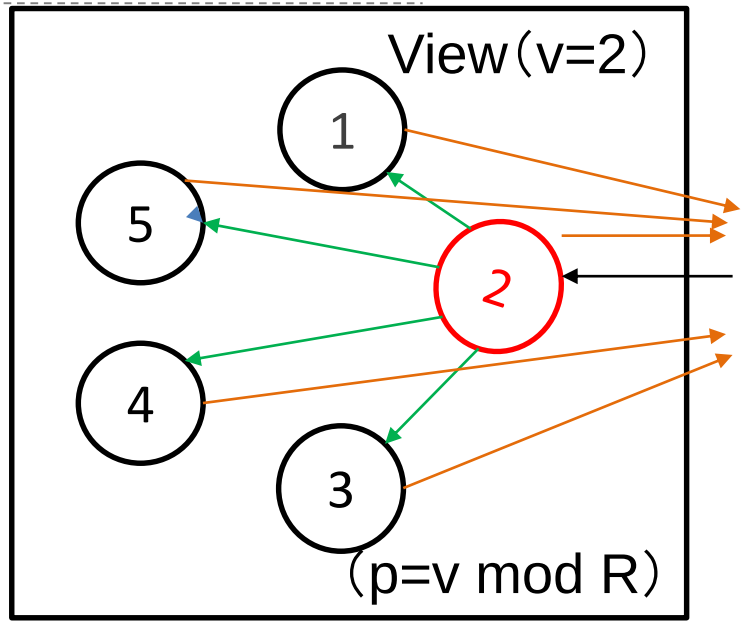
f个正常副本延迟

n-2f>f（由于f+1副本节点返回相同的结果，至少有一个正常节点参与该结果的生成）

PBFT算法

算法概述

Agreement（一致性协议）
Checkpoint（检查点协议）
View change（视图更换协议）



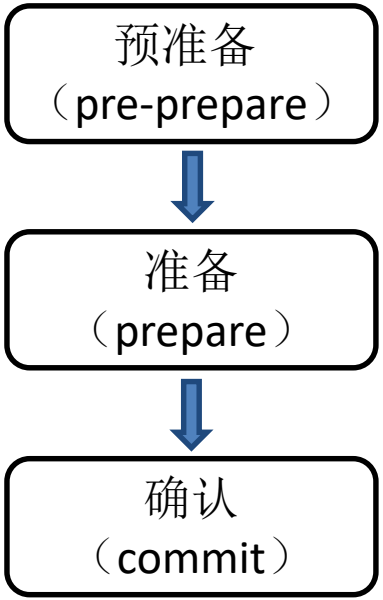
- 确定性
- 从相同的状态开始执行

client

f+1个不同副本节点发回相同的结果，作为整个操作的最终结果

PBFT算法

Agreement



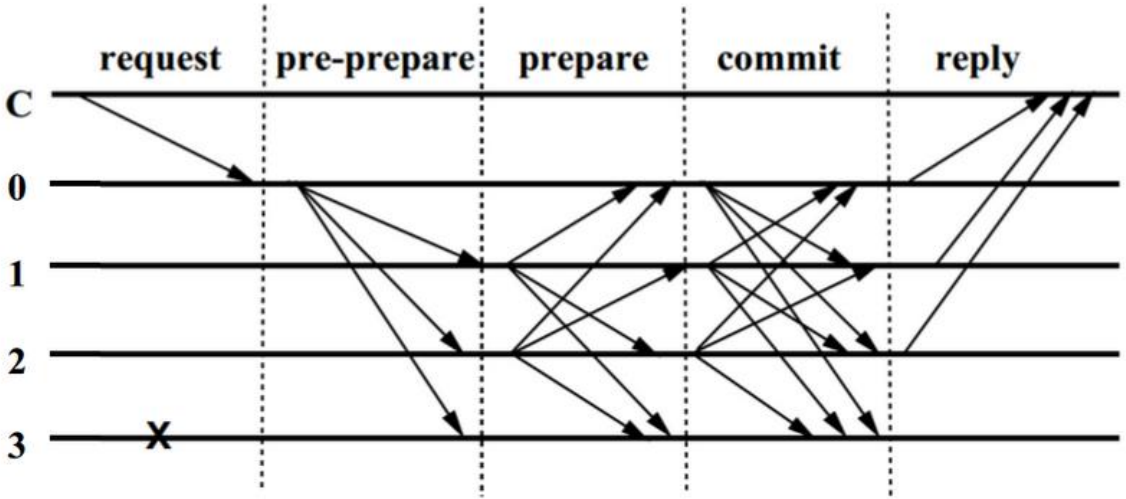
确保同一视图中请求发送的时序性

确保**不同视图之间**确认请求是严格排序的

为什么多一轮投票？

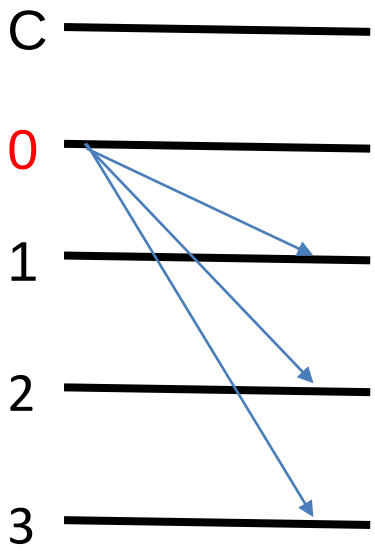
PBFT算法

Agreement



PBFT算法

Agreement——pre-prepare

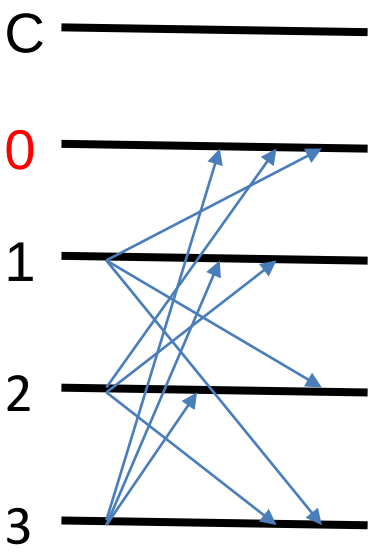


<<PRE-PREPARE,v,n,d>,m>

这里v是视图编号，n是主节点分配给该请求的序号，m是客户端发送的请求消息，d是请求消息m的摘要。

PBFT算法

Agreement——prepare



<PREPARE,v,n,d,i>

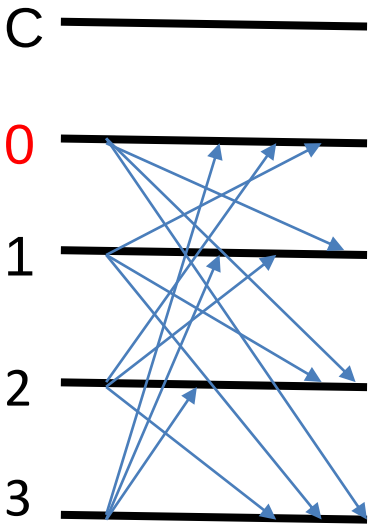
i是自身的节点序号
将预准备消息和准备消息写入自己的消息日志

接受准备消息的条件：
V一致，n在确定范围内，d正确

完成标志：
副本节点i将(m,v,n,i)以及2f个从不同副本节点收到的与预准备消息一致的准备消息（v，n，d一致）记入其消息日志

PBFT算法

agreement——commit



<COMMIT,v,n,d,i>

i是自身的节点序号

接受确认消息的条件：

V一致，n在确定范围内，d正确

PBFT算法

正确性 Safety

失效节点数 f：n>3f

f个副本失效

f个正常副本延迟

n-2f>f（由于f+1副本节点返回相同的结果，结果一定是正确的）

PBFT算法

checkpoint

证明状态的正确性，删除无异议消息记录

在请求序号可以被某个常数整除的时候周期性进行

检查点消息:<CHECKPOINT,n,d,i>

n是最近影响状态的请求序号

d是状态的摘要

i是生成该消息的副本节点序号

被证明了的检查点称为**稳定检查点**

PBFT算法

View change （Liveness）同步协议——超时机制

主节点失效 → 视图更换

由备份节点的超时机制来判断

<VIEW-CHANGE, v+1, n, C, P, i>

n是最近的稳定检查点s的序号
C是2f+1个证明s正确的检查点消息
P包含了一个Pm的集合，m是在i节点进入准备阶段的序号大于n的请求
Pm包含了一个预准备消息和2f个经验证的准备消息

PBFT算法

View change

Primary p of view v+1

收到2f个view-change message
<VIEW-CHANGE,v+1,n,C,P,i>

广播<NEW-VIEW,v+1,V,O>给其他副本节点

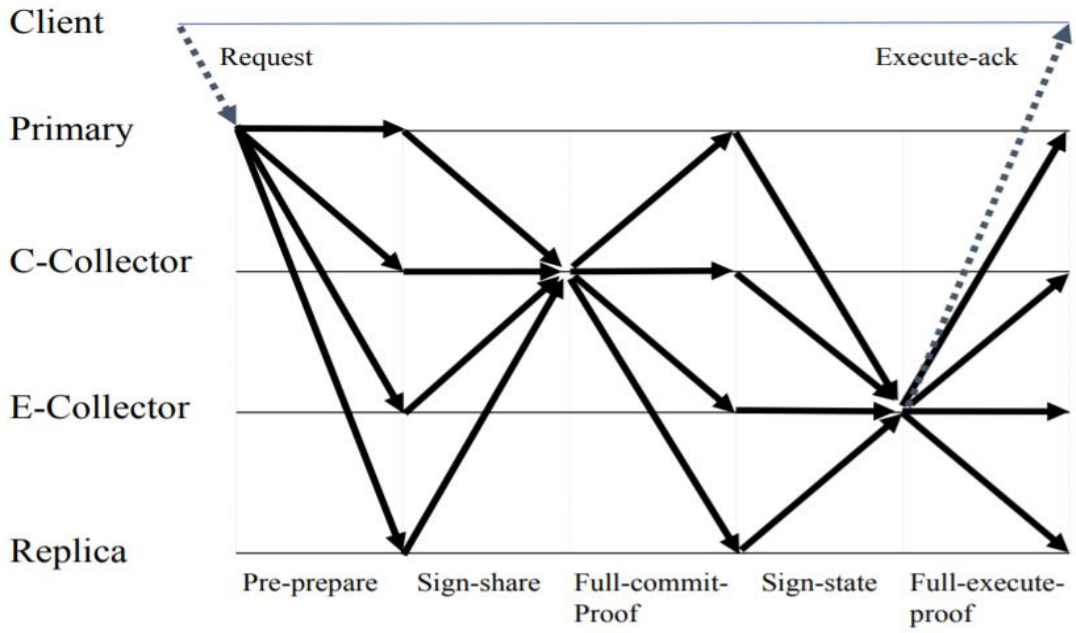
V包括接收到的view-change message和他自身的view-change message
O是预准备消息的集合

主节点p确定请求序号的范围：
低值h：V中最近的稳定检查点。
高值H：V中准备消息的最大序号。

PBFT复杂度

- **失效节点数 f：n>3f**
- Agreement 协议 Prepare与Commit阶段
- 所有节点共发出n个消息，算法复杂度$O(n^2)$
- View Change协议
- 所有节点发出n^2个消息，算法复杂度$O(n^3)$

SBFT



Collector节点采集签名并聚合验证，从而避免签名在n个节点中分发

SBFT分析

- 失效节点数 f：n>3f，门限取k=2f
- 普通节点接收常数个消息
- 聚合节点接收O(n)消息

HotStuff

- SBFT及类似协议通过门限聚合签名减少了Agreement过程的通信量
- SBFT类协议仍然没有减少View Change 过程的通信量
- 后果：

  Leader更换困难，几乎依靠指定

  集中化倾向严重，潜在单点故障问题

HotStuff

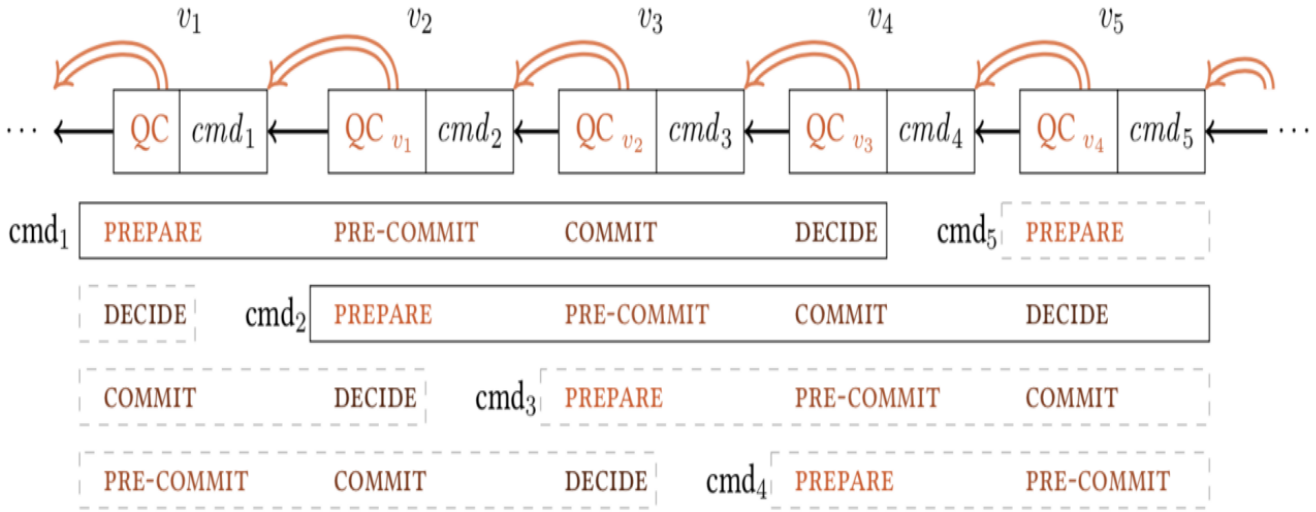
- HotStuff的设计目标：
  - 对某个提案达成共识的通信复杂度为$O(n)$，
  - 在更换视图时，其通信复杂度依然是$O(n)$
- Hot-stuff 实现方法：
  - 将PBFT的prepare，commit的两个过程扩展到了prepare，pre-commit，commit三个过程
  - 在达成共识的同时更换视图

为什么又多一轮投票？

Hot Stuff基本阶段

- PREPARE phase：新的Leader选择最优的一个节点，以及对应的投票集合highQC ，并生成新的节点，形成prepareQC
- PRE-COMMIT phase：Leader 搜集（n-f）个针对prepare的投票，并加入到prepareQC
- COMMIT phase ：Leader收到$(n-f)$个 来自replica的pre-commit 投票后，将其组装成一个precommitQC并广播

HotStuff链式结构



Hot Stuff复杂度分析

- **失效节点数 f：n>3f，门限取k=2f+1**
- 普通节点接收常数个消息
- 聚合节点接收O(n)消息

PBFT类协议的缺点

- **已知系统节点数，Permissioned**
  - 投票依赖于超过f个恶意节点，或达到n-f个投票
- **时间同步假设**
  - 依赖超时机制实现系统的存活性，否则当Leader失效后无法继续
- **压力不平衡，中性化趋势**
  - Leader节点采用门限聚合签名后可以降低其他节点压力，但是Leader节点要处理O(n)的签名消息
  - 对比Bitcoin，所有节点均只需处理O(1)消息

进一步的改进

- 能不能减少HotStuff的投票轮次？

谢谢