Basic Paxos 两阶段共识流程全解析

Grissom

2025.09

目录

CONTENTS

01

共识背景与角色

Basic Paxos 两阶段共识流程全解析

深入理解异步网络环境下，多节点如何通过精确的消息交换，对单一值达成强一致性的理论基石。

Prepare 阶段　Accept 阶段

三角色职责与核心状态

Proposer（提议者）

生成唯一递增的提案号 `n`，驱动两阶段协议，根据反馈抉择最终提案值。

Acceptor（接受者）

维护 `promised_n`，`accepted_n`，`accepted_value` 三状态，通过投票保证算法安全性。

Learner（学习者）

监听 Acceptor 的 Accepted 消息，当发现某个值被多数派接受时，学习该值并对外公布共识结果。

02

Prepare 阶段消息详解

Phase 1：Prepare 请求广播

Proposer

生成新编号 `n`

广播 Prepare(n)

抢占本轮投票权：若获多数派 Promise，则后续编号小于 `n` 的提案将被拒绝。

核心作用：此阶段不携带值，仅用于锁定“话语权”，为后续阶段形成不可逆的交集约束。

Acceptor 对 Prepare 的处理

条件满足：n > promised_n

更新本地状态

promised_n ← n

返回 Promise(含历史最高已接受值)

条件不满足：n ⩽ promised_n

拒绝旧提案

保持状态不变

返回 NACK(promised_n)

此步骤保证旧提案无法被接受，并把“历史最高值”向上传递，为后续值继承提供依据。

Proposer 收集 Promise 并抉择值

1. 收集响应
等待多数派 Promise

2. 抉择值 v
若 Promise 含已接受值，选编号最大者；否则用自选值。

3. 确定提案
最终提案确定为 (n, v)

关键规则：任何已被选定的值都会被后继更大编号提案无条件继承，确保“一旦选定，永不更改”。

03

Accept 阶段消息详解

Phase 2：Accept 请求携带最终值

Proposer

携带最终确定的 (n, v)

广播 Accept(n, v)

请求 Acceptor 接受此提案。Acceptor 本地决策，无需交互，将延迟控制在 1 轮 RTT。

Acceptor 投票与 Accepted 回应

条件满足：n ≥ promised_n

接受提案

accepted_n ← n

accepted_value ← v

返回 Accepted(n, v)

条件不满足：n < promised_n

拒绝提案

保持状态不变

返回 NACK

Accepted 消息既是对 Proposer 的确认，也是 Learner 发现共识的线索，通常采用组播或 gossip 方式扩散。

Proposer 判定“选定”并通知学习

1. 判定“选定”
收到多数派 Accepted

2. 广播 Chosen
向所有 Learner 广播

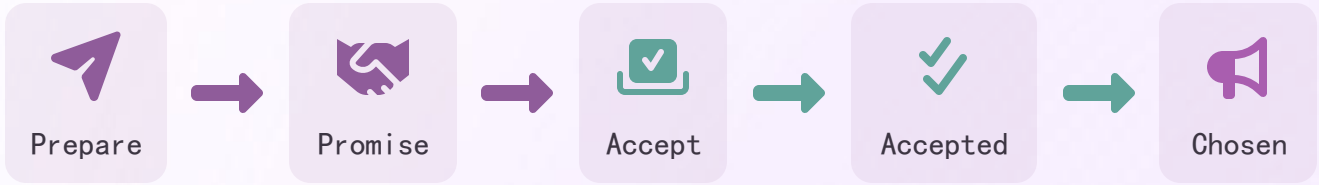
3. 共识完成
Learner 学习并公布结果

安全闭环：此时值 v 被正式选定，后续任意更大编号提案都必须继承 v，共识不可篡改。

04

消息时序与异常处理

无竞争情况下的最简消息流



时间复杂度

2 轮 RTT

消息复杂度

4n 条消息

理想环境中无重试、无冲突，是延迟上界最优路径，也是 Multi-Paxos 稳定运行时的常态。

并发冲突下的重试与活锁避免

并发冲突

多个 Proposer 同时发起不同编号，高编号 Prepare 会抬高 promised_n，导致低编号 Accept 被 NACK。

工程解决方案

通过 随机指数退避 、 Leader 租约 或批量提交，将冲突概率降到极低，保证实际可用性。

消息丢失与重复的场景容错

消息重复

Acceptor 以“最大编号”原则幂等处理。重复 Prepare 只刷新 promised_n；重复 Accept 若编号相同则再次返回 Accepted，不影响结果。

消息丢失

由 Proposer 超时重传驱动，只要最终能凑齐多数派响应，协议即可推进。仅需额外 RTT 作为代价。

该设计使 Paxos 在异步、丢包、乱序网络下仍保持安全，体现了其强大的容错能力。

05

总结与扩展思考

两阶段消息流核心要点回顾

Prepare 阶段

用“承诺”锁住历史，阻止旧提案复活，为后续阶段形成不可逆的交集约束。

Accept 阶段

用“投票”写入新值，形成新的多数派共识，完成值的选定。

整个流程仅依赖 编号大小比较 和 多数派交集 ，逻辑极简却保证安全。掌握这条消息链条，就能理解所有 Paxos 变种的演进思路。

从 Basic 流程到工程优化路径

Basic Paxos

两阶段完整执行，无优化，理论基石。

↓

Multi-Paxos

稳定 Leader 摊销 Prepare，日常仅需 Accept，高吞吐。

↓

Fast Paxos

客户端直连 Acceptor，理想情况 1 轮，低延迟。

EPaxos

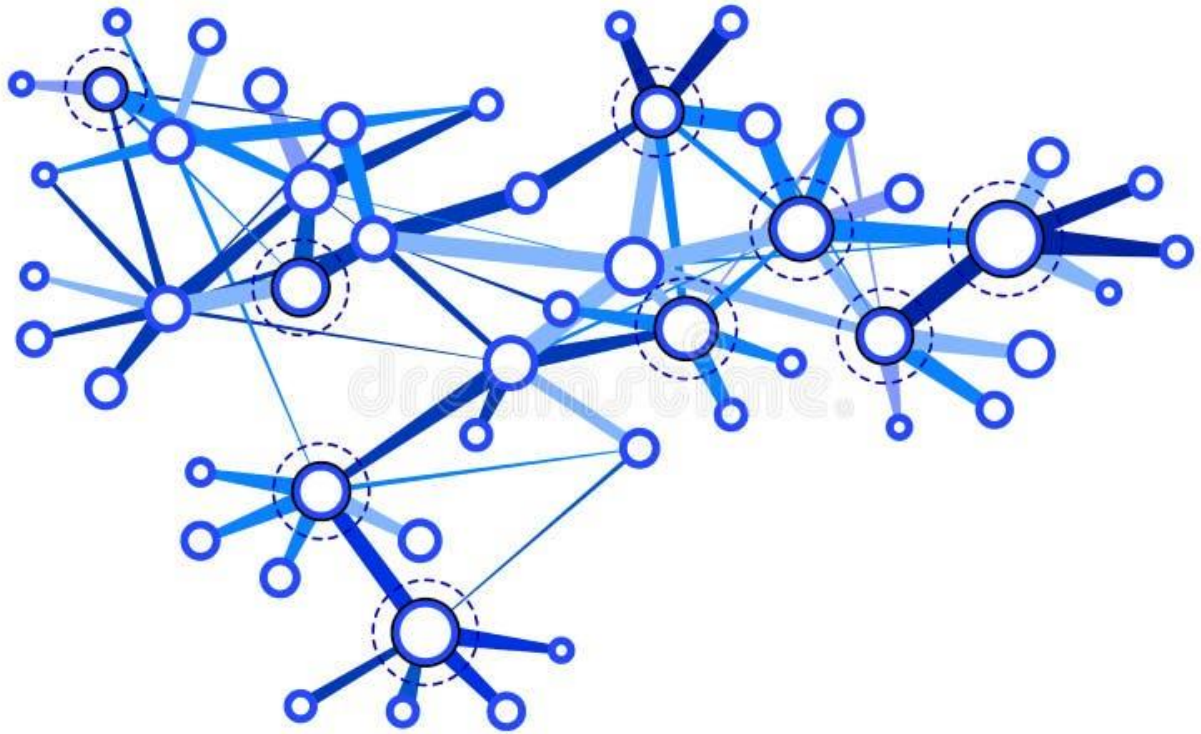
无 Leader，依赖图并行排序，高并发。

所有优化都在 Basic 框架内做减法或并行化，但核心不变式始终保留，体现理论抽象的持久指导力。

学习启示与实践建议

理解消息顺序比背诵伪代码更重要。通过亲手调通两阶段消息，才能真正体会“异步网络 + 崩溃模型”下实现强一致的精妙之处。

- 动手实践：用 UDP 实现最小 Paxos，注入丢包与重复，观察容错机制。
- 进阶实验：实现租约选主，测量退避策略对尾延迟的改善。
- 理论基础：为后续研究 Raft、PBFT、HotStuff 奠定扎实的直觉基础。



感谢观看