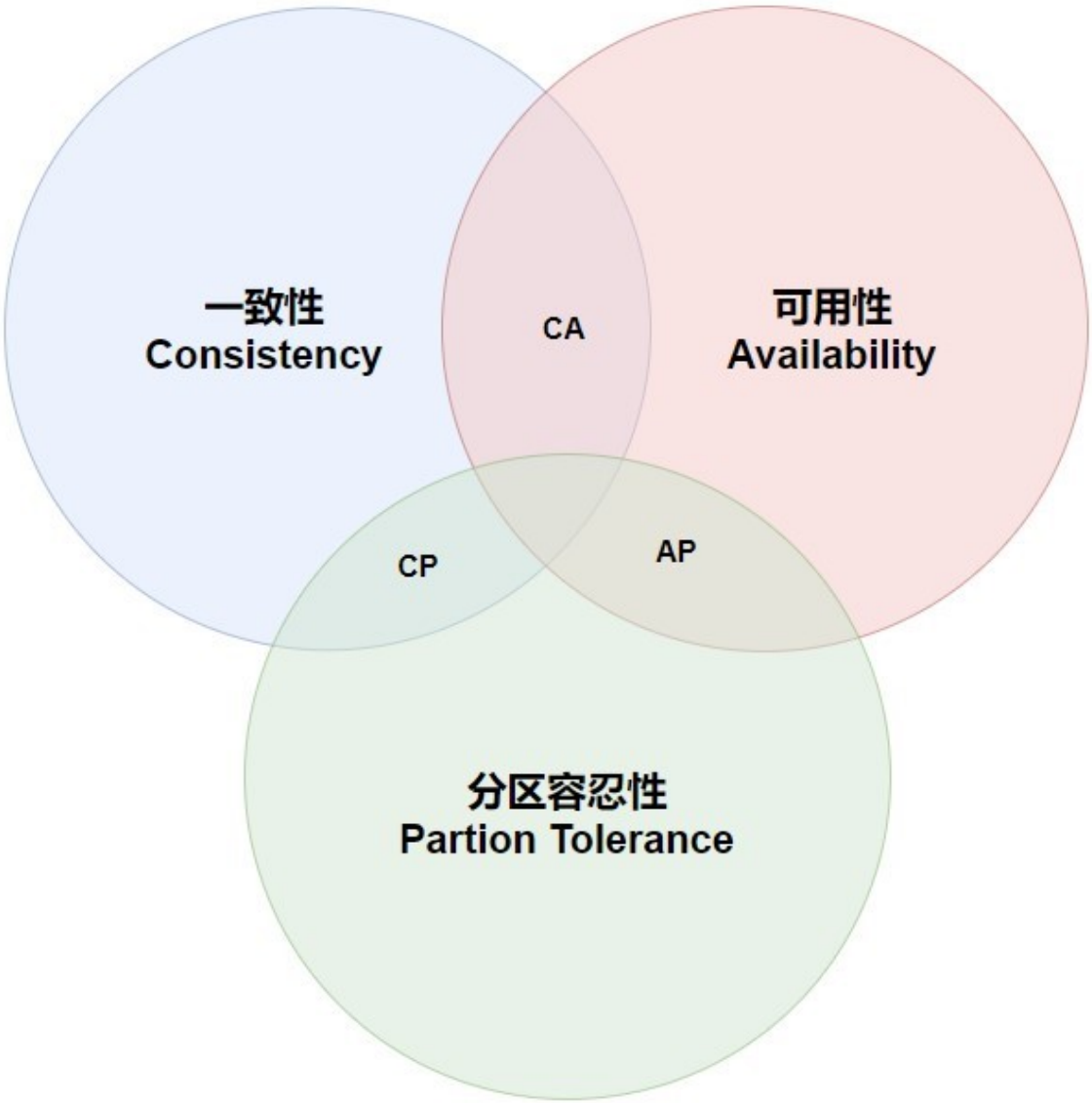
一致性

原作者github: https://github.com/sjsdfg/Interview-Notebook-PDF

PDF制作github: https://github.com/sjsdfg/Interview-Notebook-PDF

一、CAP

分布式系统不可能同时满足一致性（C：Consistency）、可用性（A：Availability）和分区容忍性（P：Partition Tolerance），最多只能同时满足其中两项。



一致性

一致性指的是多个数据副本是否能保持一致的特性。

在一致性的条件下，系统在执行数据更新操作之后能够从一致性状态转移到另一个一致性状态。

对系统的一个数据更新成功之后，如果所有用户都能够读取到最新的值，该系统就被认为具有强一致性。

可用性

可用性指分布式系统在面对各种异常时可以提供正常服务的能力，可以用系统可用时间占总时间的比值来衡量，4 个 9 的可用性表示系统 99.99% 的时间是可用的。

在可用性条件下，系统提供的服务一直处于可用的状态，对于用户的每一个操作请求总是能够在有限的时间内返回结果。

分区容忍性

网络分区指分布式系统中的节点被划分为多个区域，每个区域内部可以通信，但是区域之间无法通信。

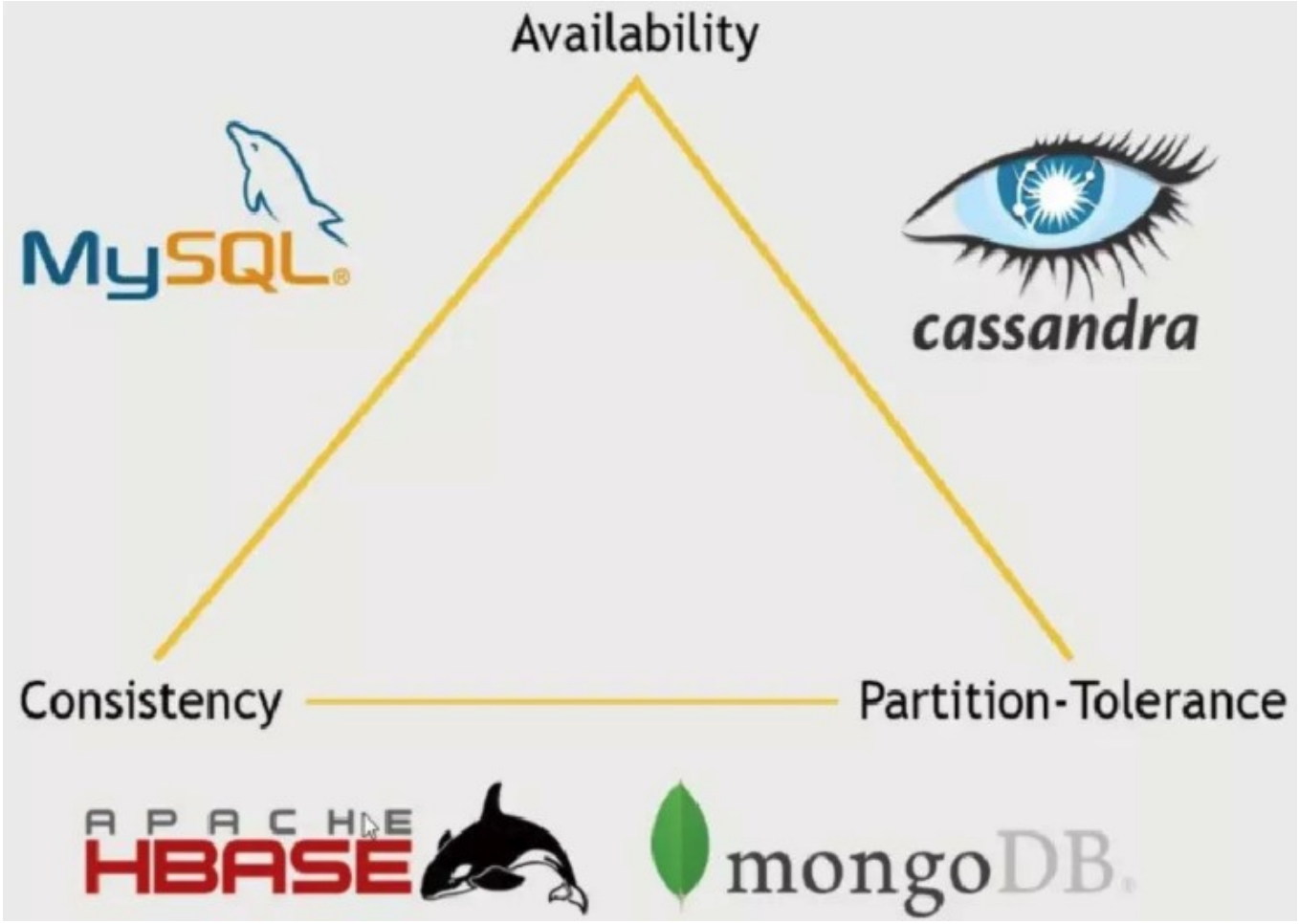
在分区容忍性条件下，分布式系统在遇到任何网络分区故障的时候，仍然需要能对外提供一致性和可用性的服务，除非是整个网络环境都发生了故障。

权衡

在分布式系统中，分区容忍性必不可少，因为需要总是假设网络是不可靠的。因此，CAP 理论实际在是要在可用性和一致性之间做权衡。

可用性和一致性往往是冲突的，很难都使它们同时满足。在多个节点之间进行数据同步时，

- 为了保证一致性（CP），就需要让所有节点下线成为不可用的状态，等待同步完成；
- 为了保证可用性（AP），在同步过程中允许读取所有节点的数据，但是数据可能不一致。



二、BASE

BASE 是基本可用（Basically Available）、软状态（Soft State）和最终一致性（Eventually Consistent）三个短语的缩写。

BASE 理论是对 CAP 中一致性和可用性权衡的结果，它的理论的核心思想是：即使无法做到强一致性，但每个应用都可以根据自身业务特点，采用适当的方式来使系统达到最终一致性。

基本可用

指分布式系统在出现故障的时候，保证核心可用，允许损失部分可用性。

例如，电商在做促销时，为了保证购物系统的稳定性，部分消费者可能会被引导到一个降级的页面。

软状态

指允许系统中的数据存在中间状态，并认为该中间状态不会影响系统整体可用性，即允许系统不同节点的数据副本之间进行同步的过程存在延时。

最终一致性

最终一致性强调的是系统中所有的数据副本，在经过一段时间的同步后，最终能达到一致的状态。

ACID 要求强一致性，通常运用在传统的数据库系统上。而 BASE 要求最终一致性，通过牺牲强一致性来达到可用性，通常运用在大型分布式系统中。

在实际的分布式场景中，不同业务单元和组件对一致性的要求是不同的，因此 ACID 和 BASE 往往会结合在一起使用。

三、2PC

两阶段提交（Two-phase Commit，2PC）

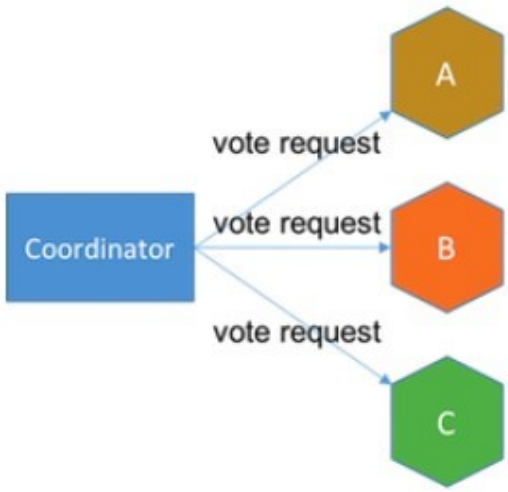
主要用于实现分布式事务，分布式事务指的是事务操作跨越多个节点，并且要求满足事务的 ACID 特性。

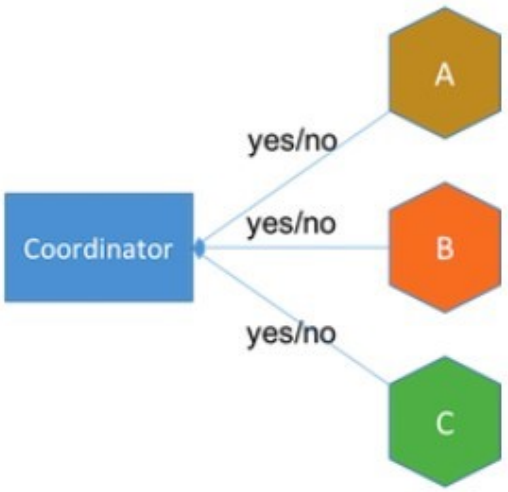
通过引入协调者（Coordinator）来调度参与者的行为，，并最终决定这些参与者是否要真正执行事务。

运行过程

1. 准备阶段

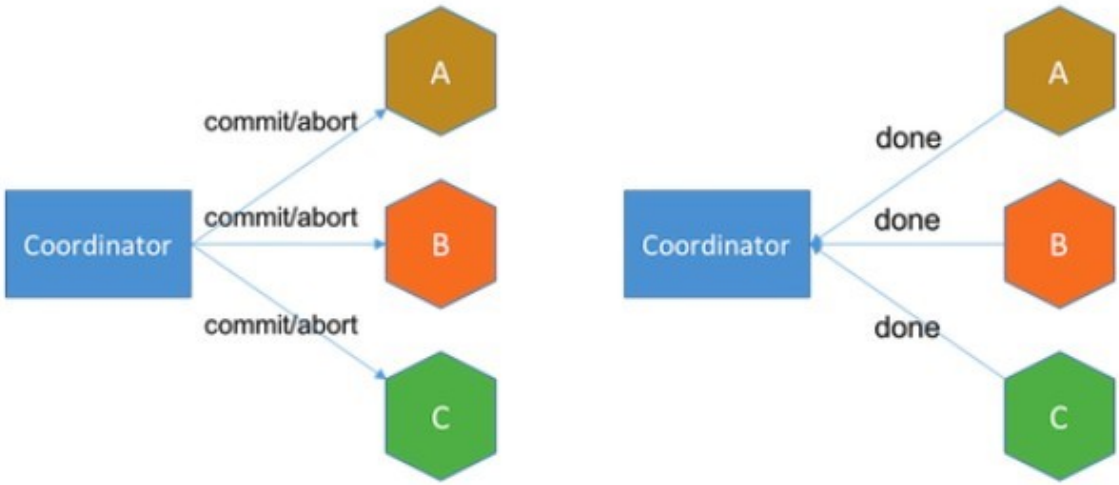
协调者询问参与者事务是否执行成功，参与者发回事务执行结果。





2. 提交阶段

如果事务在每个参与者上都执行成功，事务协调者发送通知让参与者提交事务；否则，协调者发送通知让参与者回滚事务。



存在的问题

1. 同步阻塞

所有事务参与者在等待其它参与者响应的时候都处于同步阻塞状态，无法进行其它操作。

2. 单点问题

协调者在 2PC 中起到非常大的作用，发生故障将会造成很大影响，特别是在阶段二发生故障，所有参与者会一直等待状态，无法完成其它操作。

3. 数据不一致

在阶段二，如果协调者只发送了部分 Commit 消息，此时网络发生异常，那么只有部分参与者接收到 Commit 消息，也就是说只有部分参与者提交了事务，使得系统数据不一致。

4. 太过保守

任意一个节点失败就会导致整个事务失败，没有完善的容错机制。

四、Paxos

用于达成共识性问题，即对多个节点产生的值，该算法能保证只选出唯一一个值。

主要有三类节点：

- 提议者（Proposer）：提议一个值；
- 接受者（Acceptor）：对每个提议进行投票；
- 告知者（Learner）：被告知投票的结果，不参与投票过程。

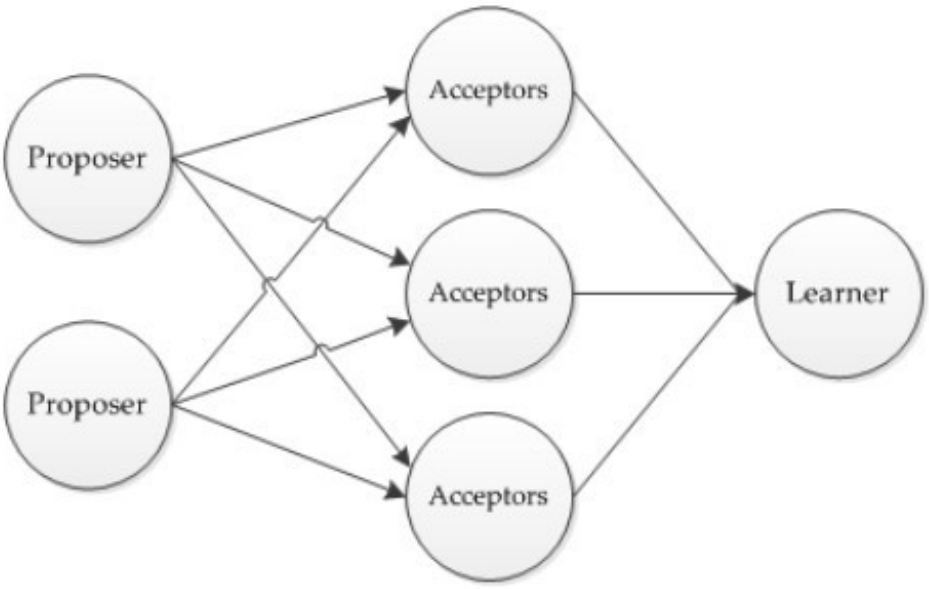


Figure 1: Basic Paxos architecture. A number of proposers make proposals to acceptors. When an acceptor accepts a value it sends the result to learner nodes.

执行过程

规定一个提议包含两个字段：[n, v]，其中 n 为序号（具有唯一性），v 为提议值。

下图演示了两个 Proposer 和三个 Acceptor 的系统中运行该算法的初始过程，每个 Proposer 都会向所有 Acceptor 发送提议请求。

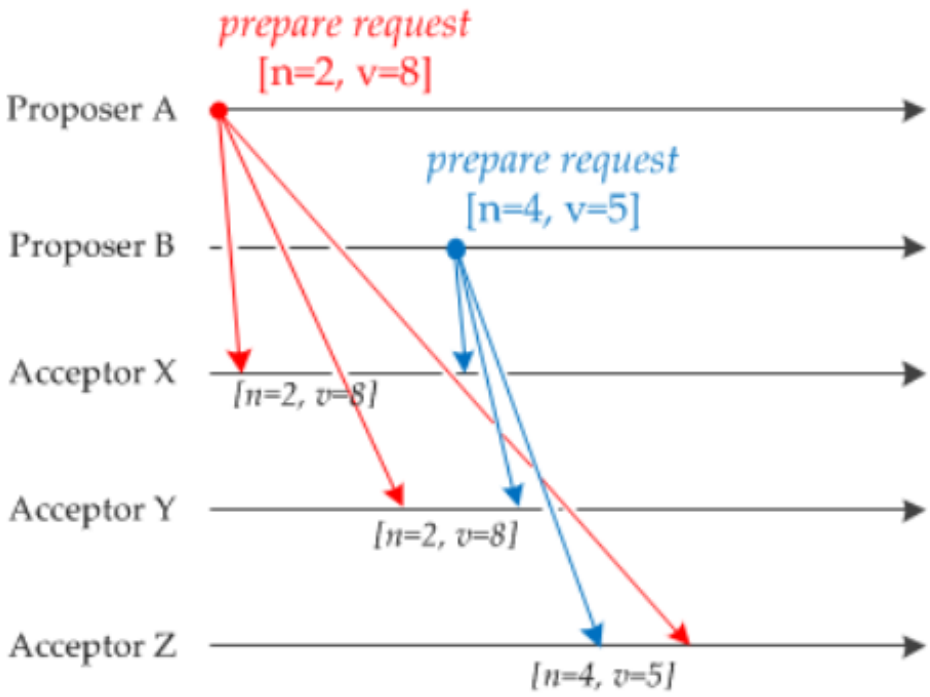


Figure 2: Paxos. Proposers A and B each send prepare requests to every acceptor. In this example proposer A's request reaches acceptors X and Y first, and proposer B's request reaches acceptor Z first.

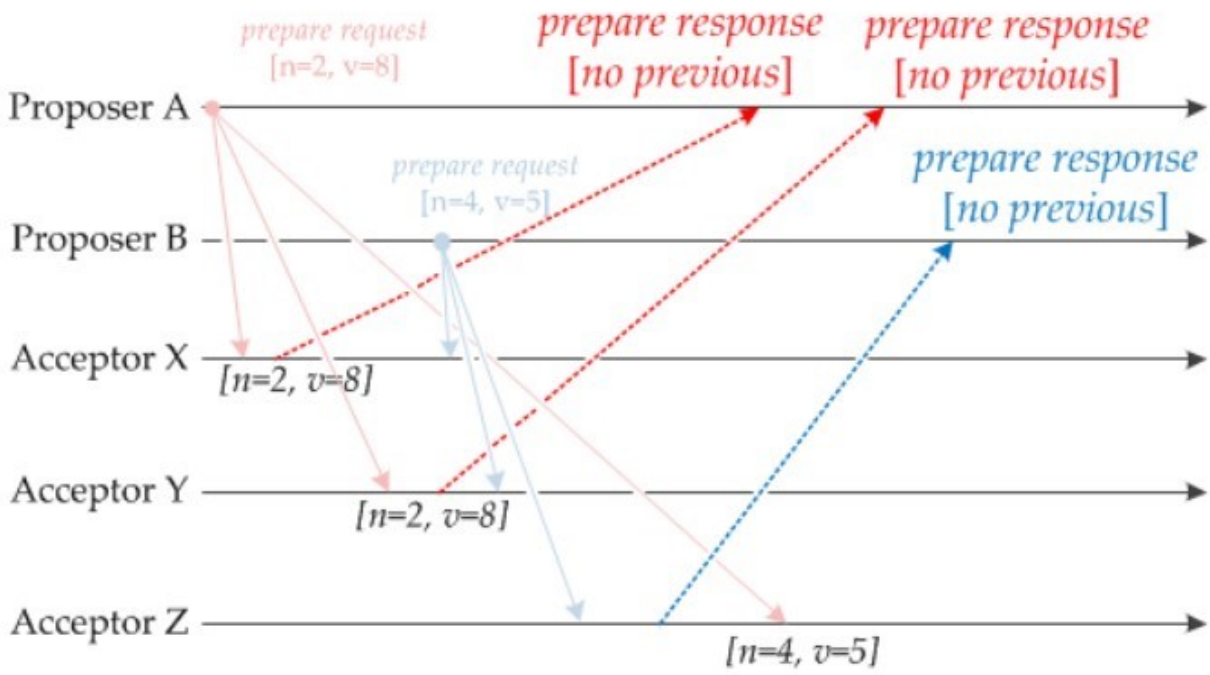


Figure 3: Paxos. Each acceptor responds to the first prepare request message that it receives.

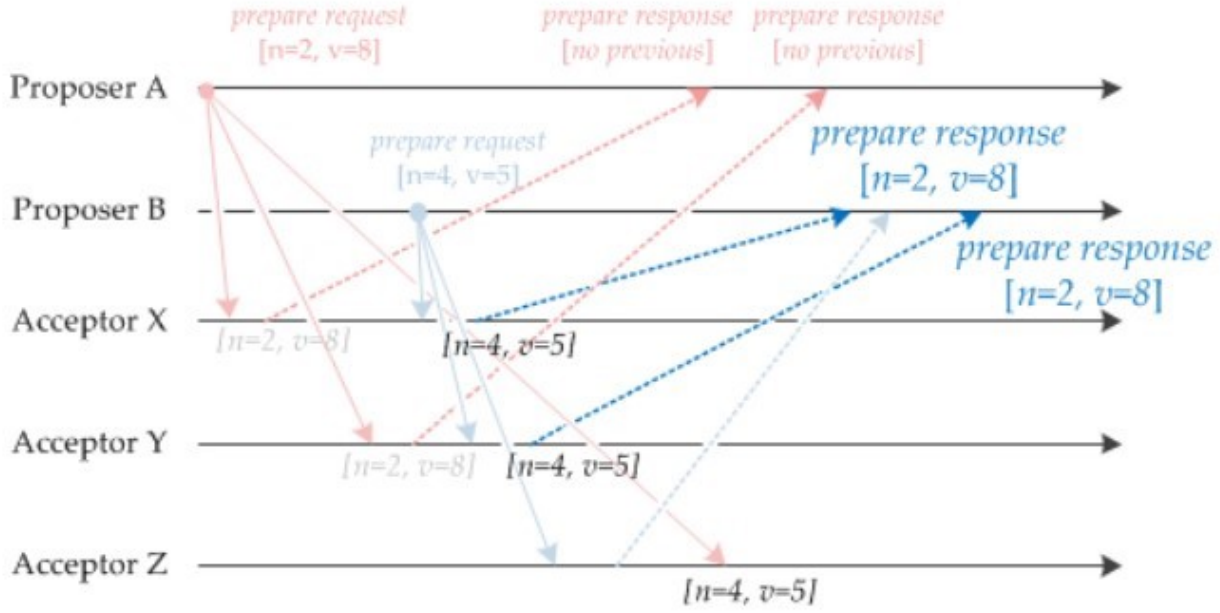


Figure 4: Paxos. Acceptor Z ignores proposer A's request because it has already seen a higher numbered proposal (4> 2). Acceptors X and Y respond to proposer B's request with the previous highest request that they acknowledged, and a promise to ignore any lower numbered proposals.

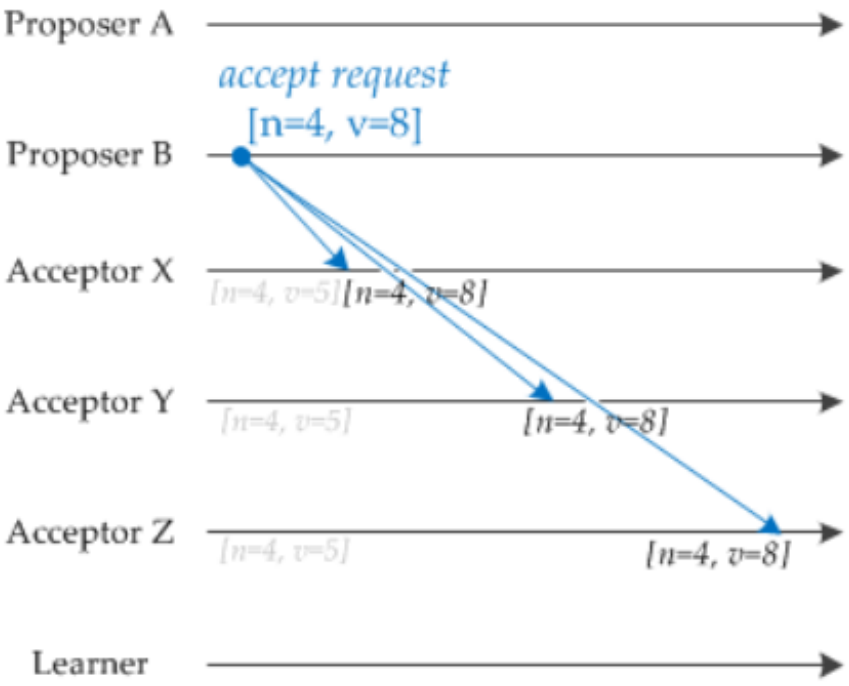
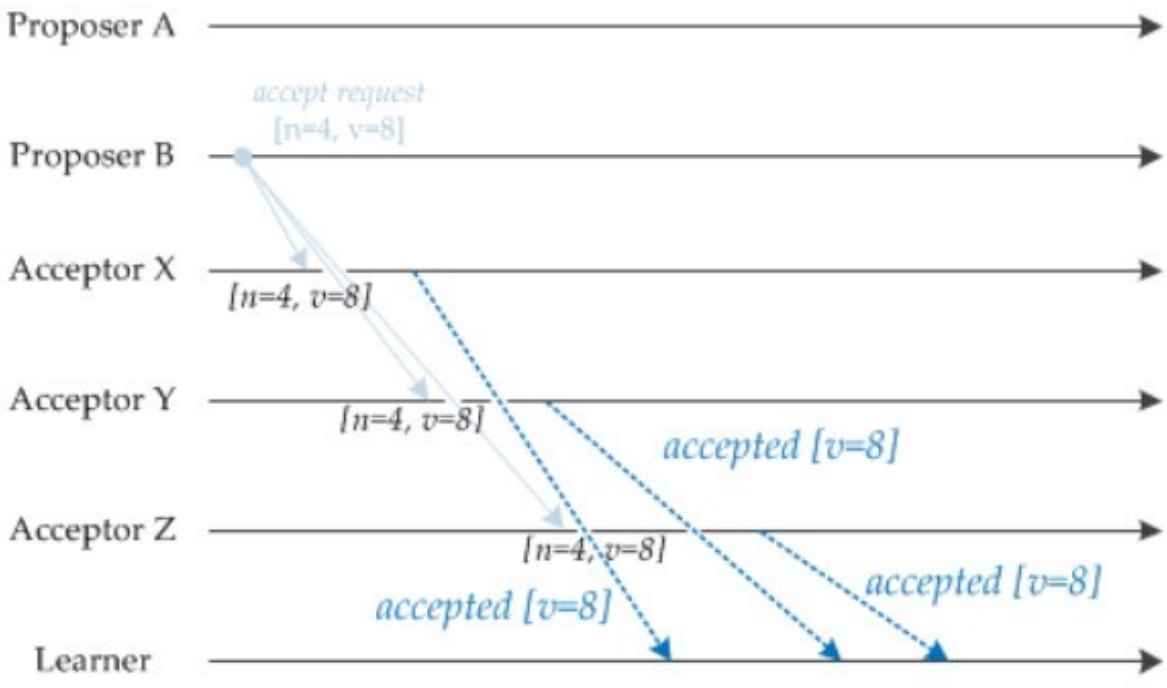


Figure 5: Paxos. Proposer B sends an accept request to each acceptor, with its previous proposal number (4), and the value of the highest numbered proposal it has seen (8, from [n=2, v=8



约束条件

1. 正确性

指只有一个提议值会生效。

因为 Paxos 协议要求每个生效的提议被多数 Acceptor 接收，并且 Acceptor 不会接受两个不同的提议，因此可以保证正确性。

2. 可终止性

指最后总会有一个提议生效。

Paxos 协议能够让 Proposer 发送的提议朝着能被大多数 Acceptor 接受的那个提议靠拢，因此能够保证可终止性。

五、Raft

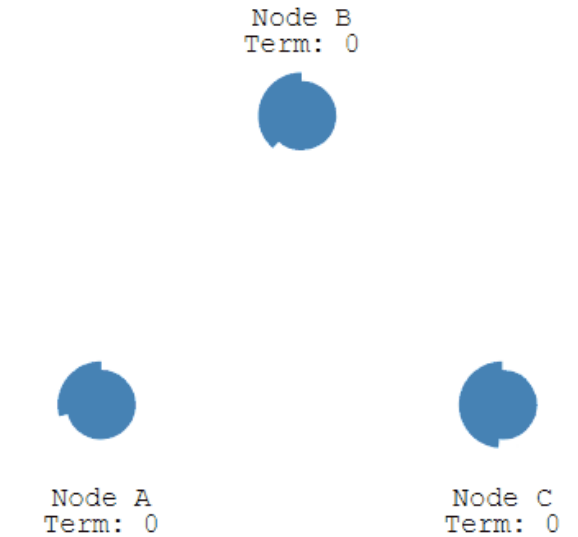
Raft 和 Paxos 类似，但是更容易理解，也更容易实现。

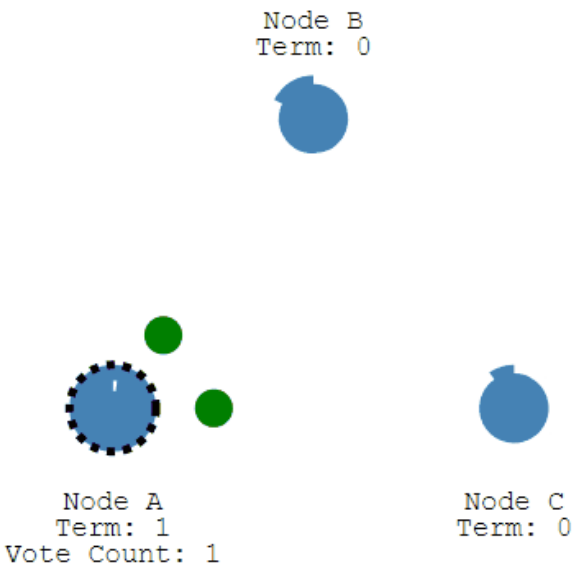
Raft 主要是用来竞选主节点。

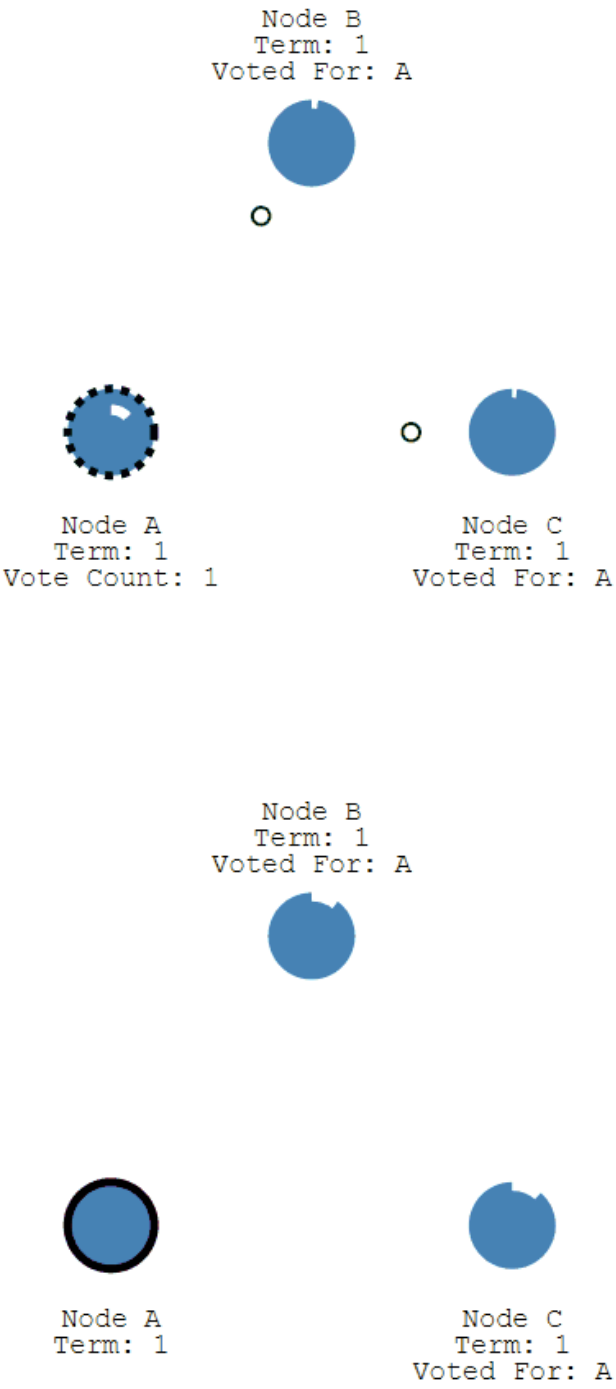
单个 Candidate 的竞选

有三种节点：Follower、Candidate 和 Leader。Leader 会周期性的发送心跳包给 Follower。每个 Follower 都设置了一个随机的竞选超时时间，一般为 150ms\~300ms，如果在这个时间内没有收到 Leader 的心跳包，就会变成 Candidate，进入竞选阶段。

- 下图表示一个分布式系统的最初阶段，此时只有 Follower，没有 Leader。Follower A 等待一个随机的竞选超时时间之后，没收到 Leader 发来的心跳包，因此进入竞选阶段。

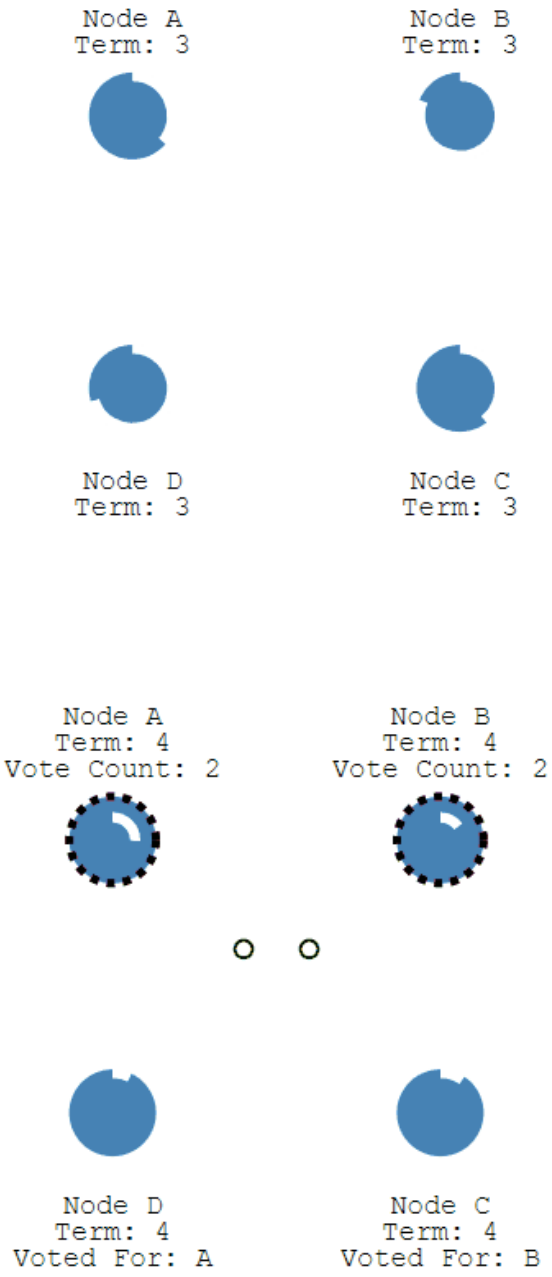

Node B
Term: 0
Node A
Term: 0
Node C
Term: 0


Node B
Term: 0
Node A
Term: 1
Vote Count: 1
Node C
Term: 0



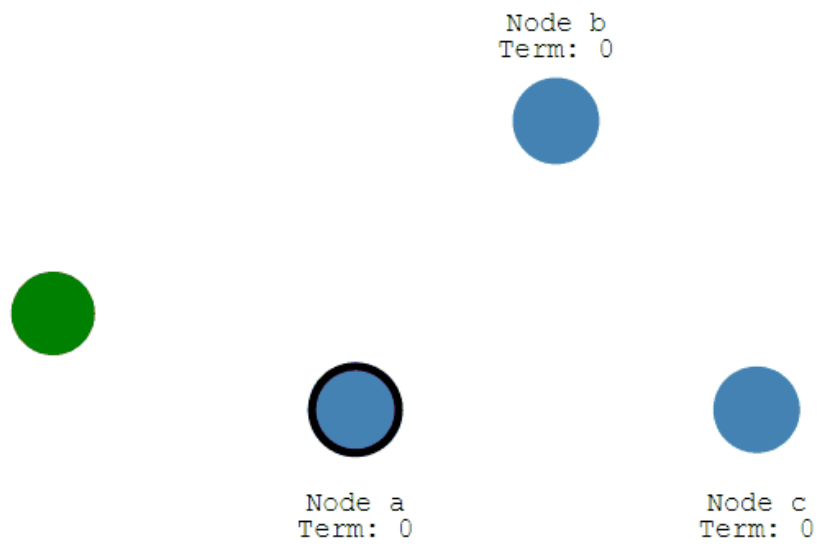
多个 Candidate 竞选

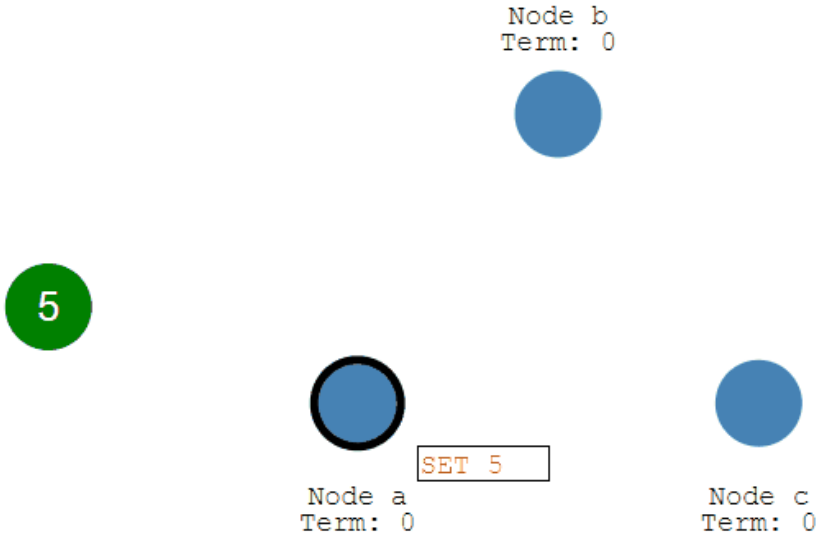
- 如果有多个 Follower 成为 Candidate，并且所获得票数相同，那么就需要重新开始投票，例如下图中 Candidate B 和 Candidate D 都获得两票，因此需要重新开始投票。

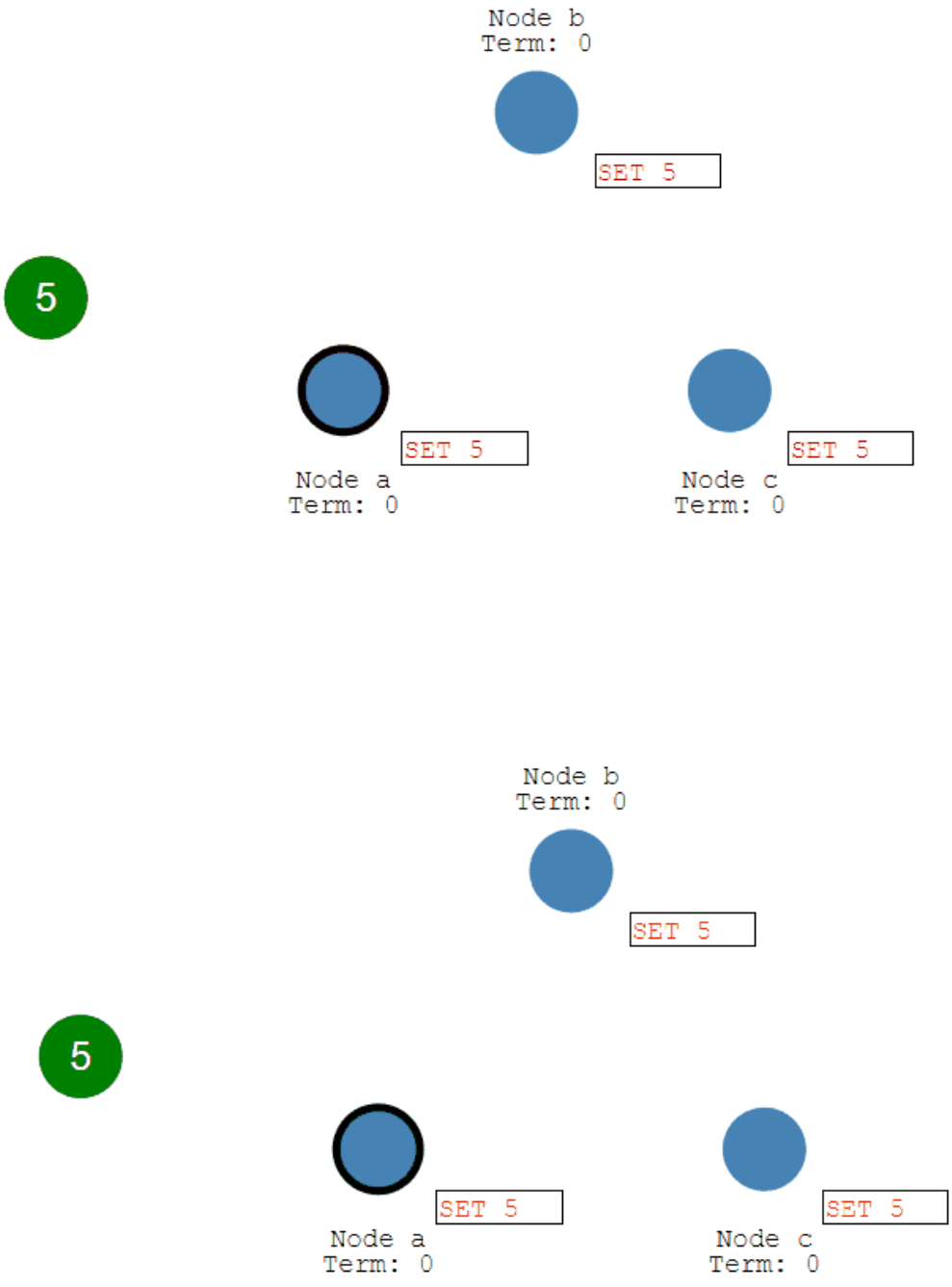


日志复制

- 来自客户端的修改都会被传入 Leader。注意该修改还未被提交，只是写入日志中。


Node b
Term: 0
Node a
Term: 0
Node c
Term: 0


Node b
Term: 0
5
SET 5
Node a
Term: 0
Node c
Term: 0



参考资料

- 倪超. 从 Paxos 到 ZooKeeper : 分布式一致性原理与实践 [M]. 电子工业出版社, 2015.

- What is CAP theorem in distributed database system?
- NEAT ALGORITHMS - PAXOS
- Raft: Understandable Distributed Consensus
- Paxos By Example

---

github: https://github.com/sjsdfg/Interview-Notebook-PDF