Paxos

余辰侃

Basic Paxos 目标

- 在多数成员可用的情况下，在多个提案（如多主环境）中确定唯一值

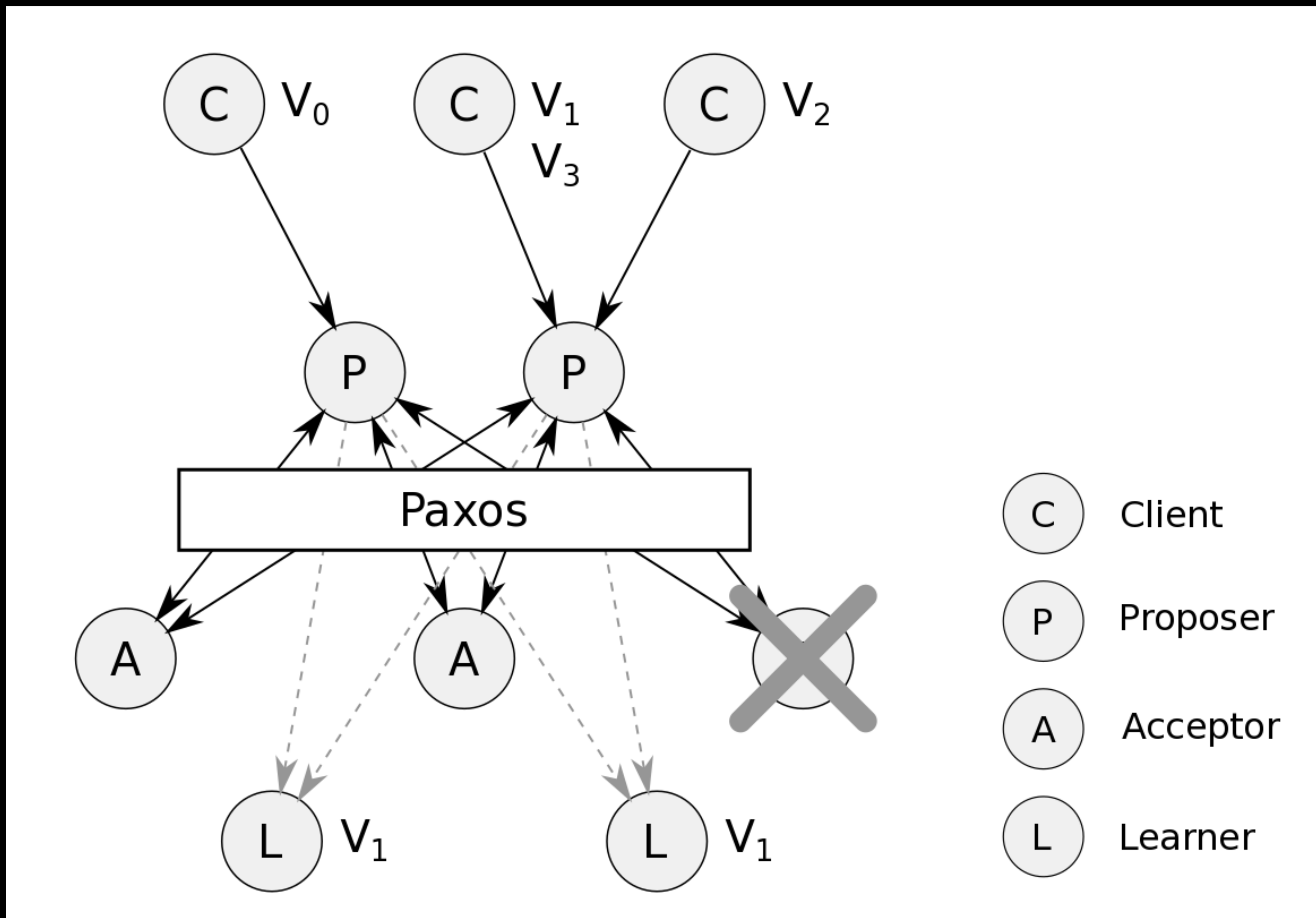
Basic Paxos 成员

- Proposer：提案发起者，可同时任意发起多个不同提案
- Acceptor：提案接受者，超过半数（多数派）确认接受某一提案后该提案被认为为最终结果
- Learner：被确认的提案被广播给学习者学习

Basic Paxos 保证

- 系统在多数成员(Acceptors)存活的情况下可用
- 支持乱序，丢包，重复，高延迟的网络环境（不支持错包)
- 保证提案从所有提议中选出
- 提案一旦被确定即保证不被更改

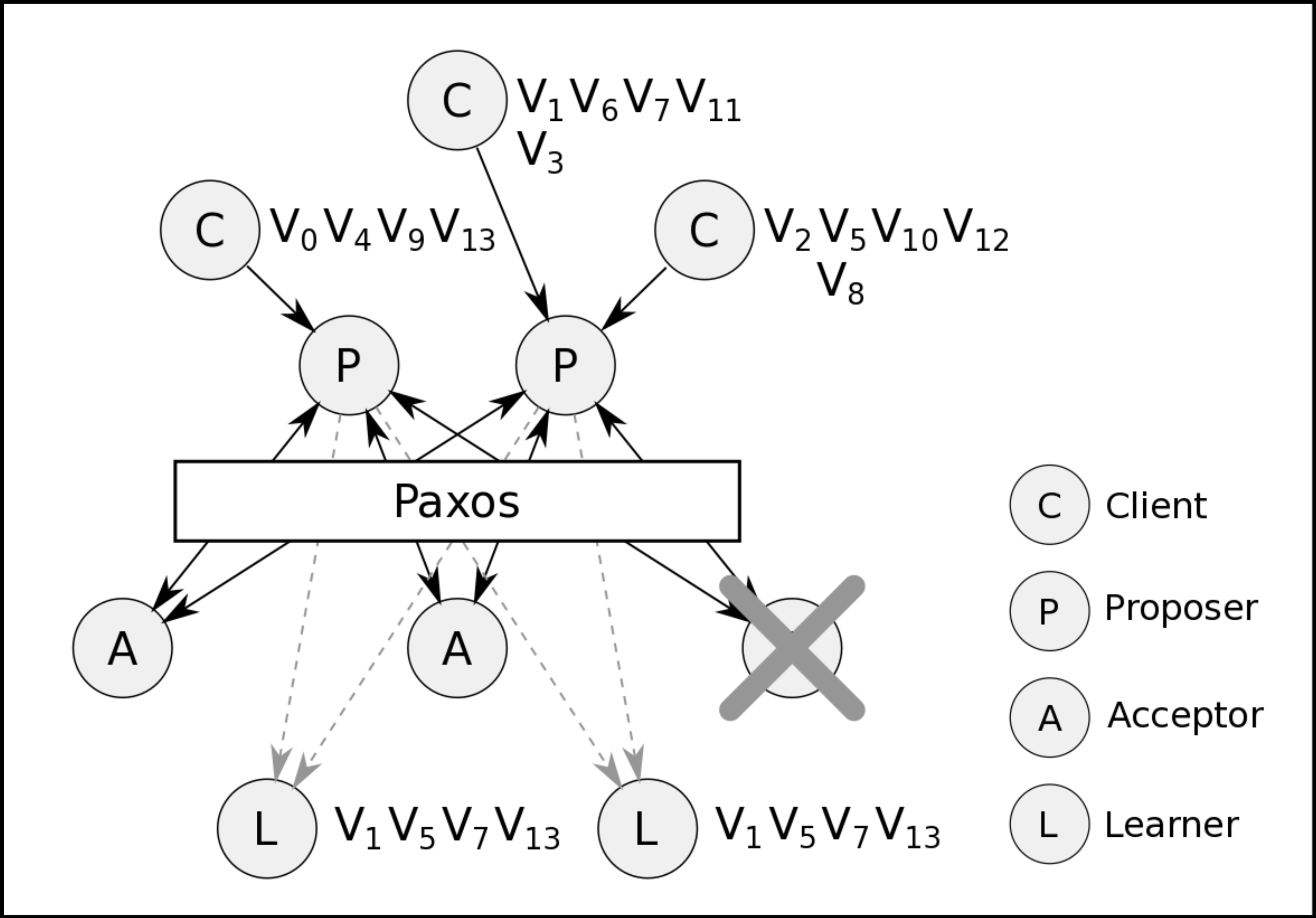
Basic Paxos 示例



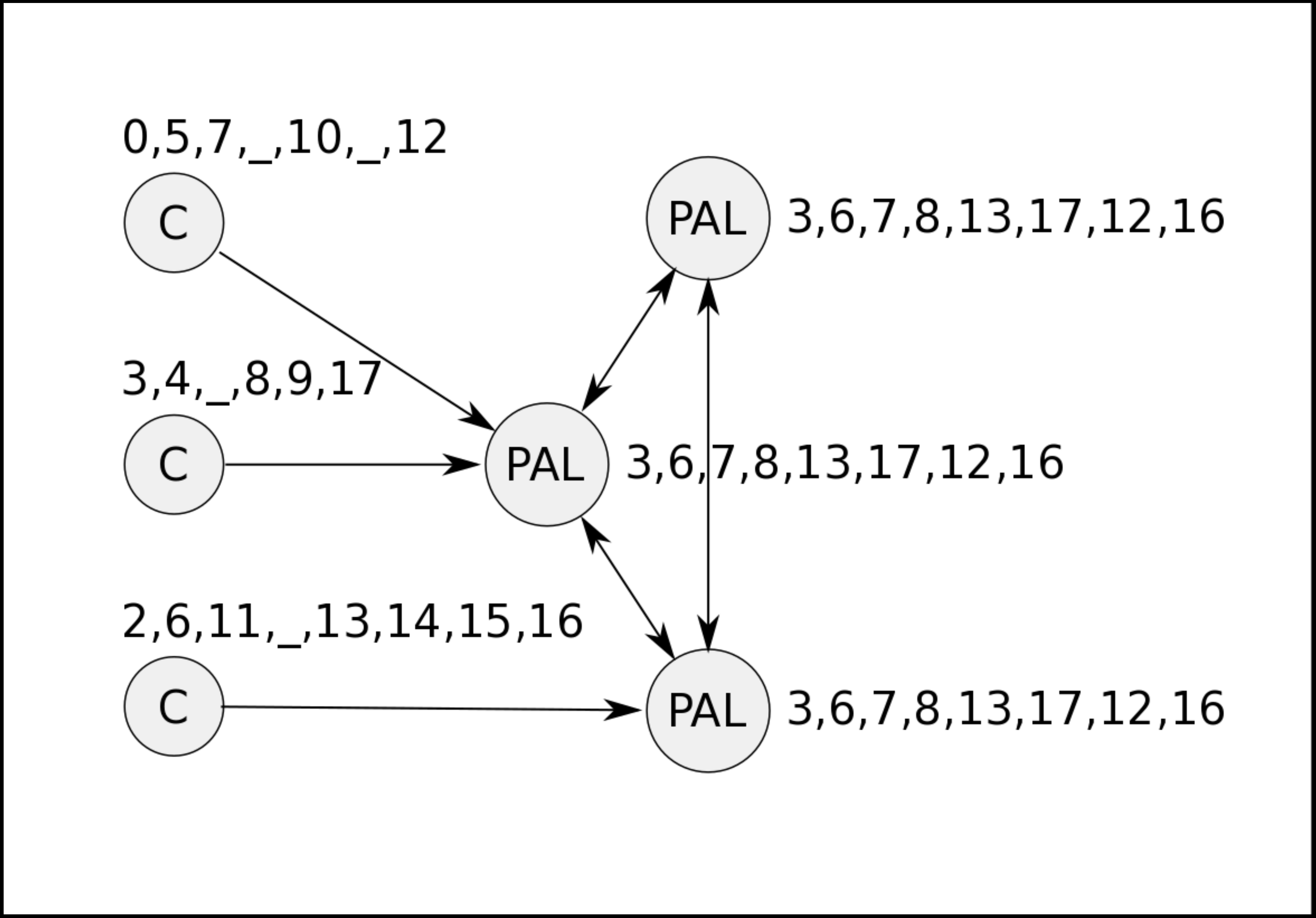
Multi Paxos 目标

- 在多数成员（Acceptors）可用的情况下，可持续从连续提议中确定一组值序列
  - 操作序列（读写） => 状态机（存储）

Multi Paxos 示例



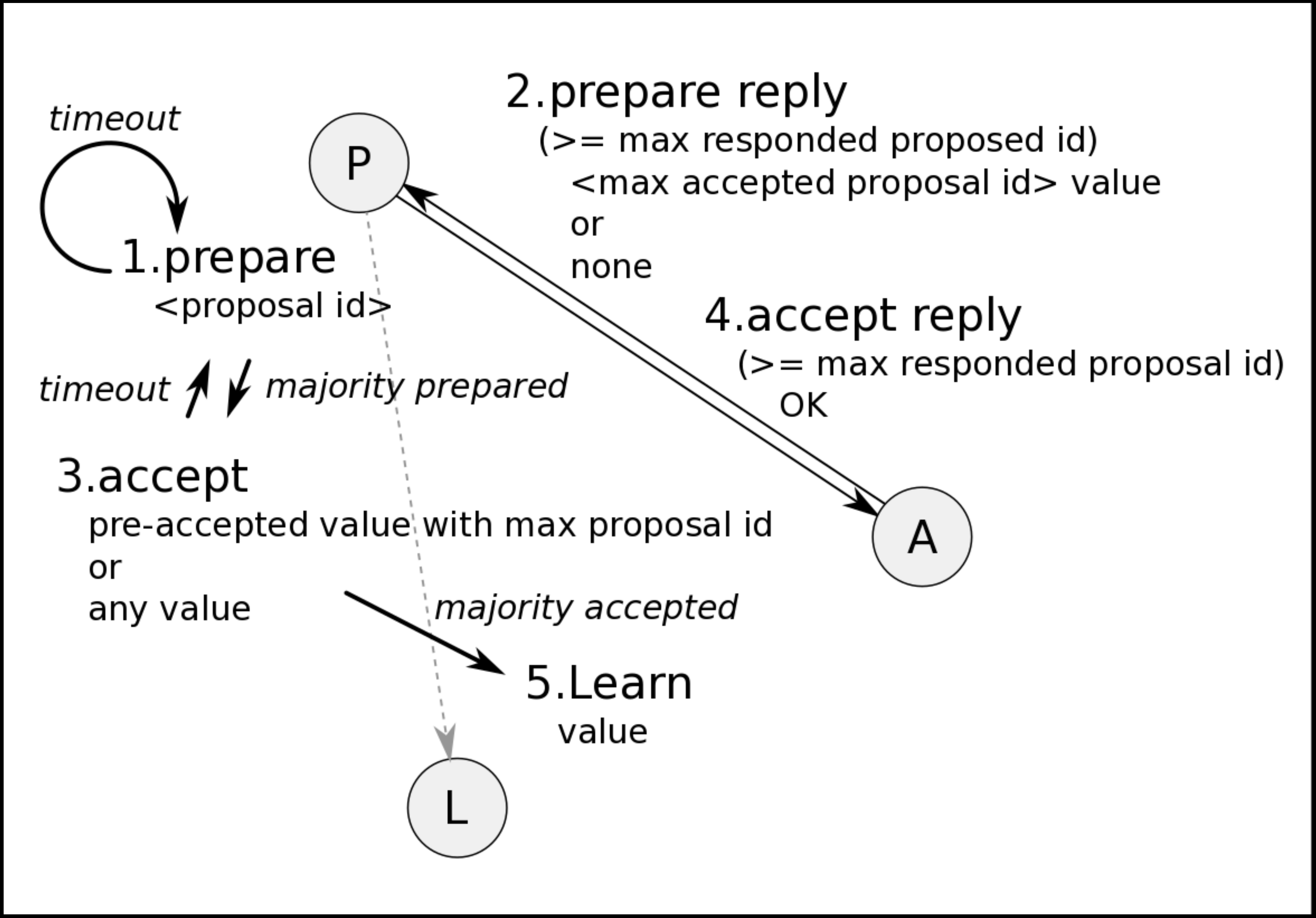
Multi Paxos 示例



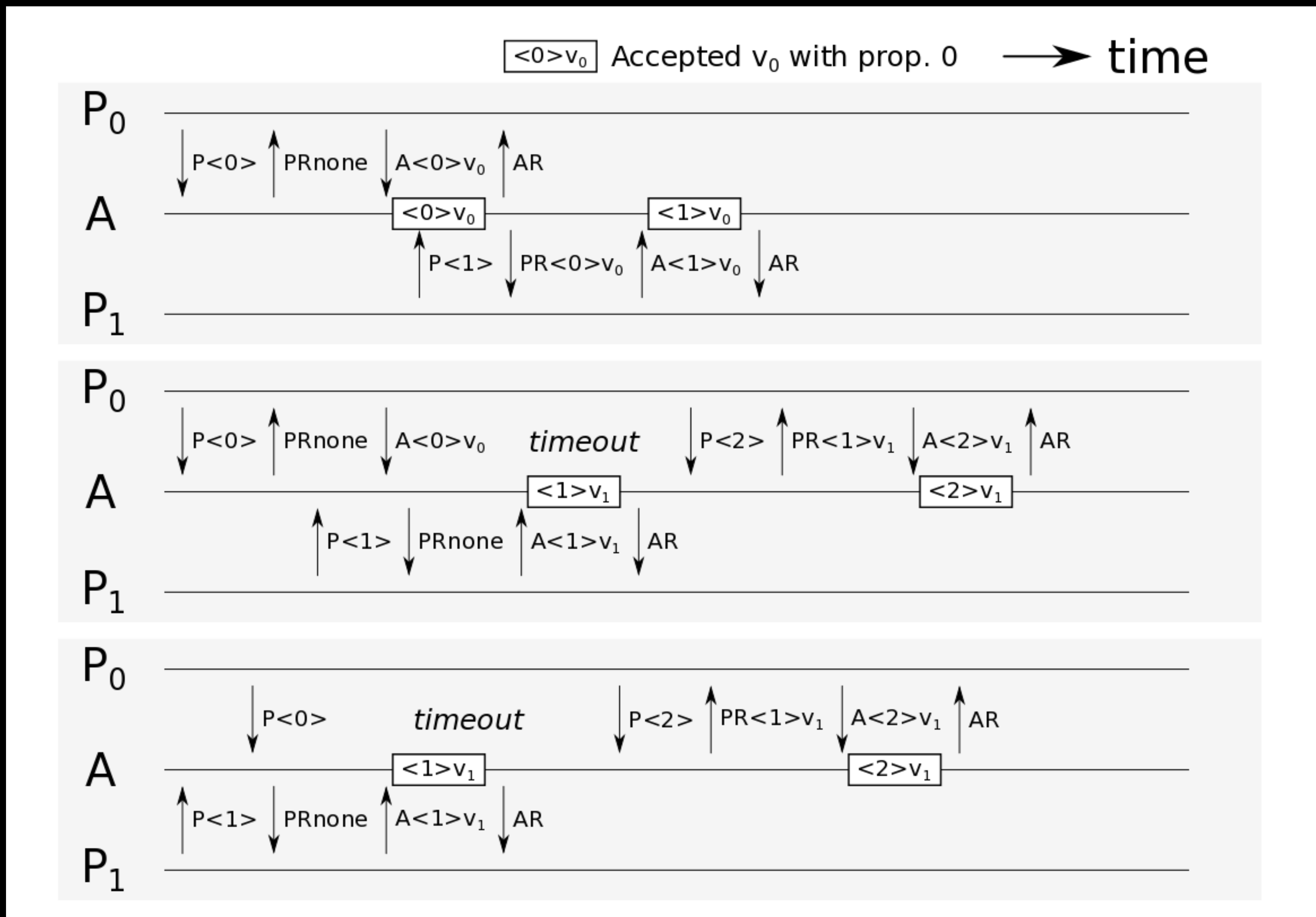
Basic Paxos 实现

- Prepare
  - Proposer：向所有 acceptor 发起含唯一递增 proposal id *n* 的 prepare 请求
  - Acceptor：如果没有响应过 proposal id 大于 *n* 的 prepare 请求，返回最近接受（最大 proposal id）的值及其 proposal id，空若无
- Accept
  - Proposer：如果多数 acceptor 响应 prepare 请求，从返回值中选择 proposal id 最大的值向所有 acceptor 发起含 proposal id *n* 的 accept 请求，空则任意
  - Acceptor：如果没有响应过 proposal id 大于 *n* 的 prepare 请求，接受值并返回
- Learn
  - Proposer：如果多数 acceptor 响应 accept 请求，确认该值，广播给所有 learner

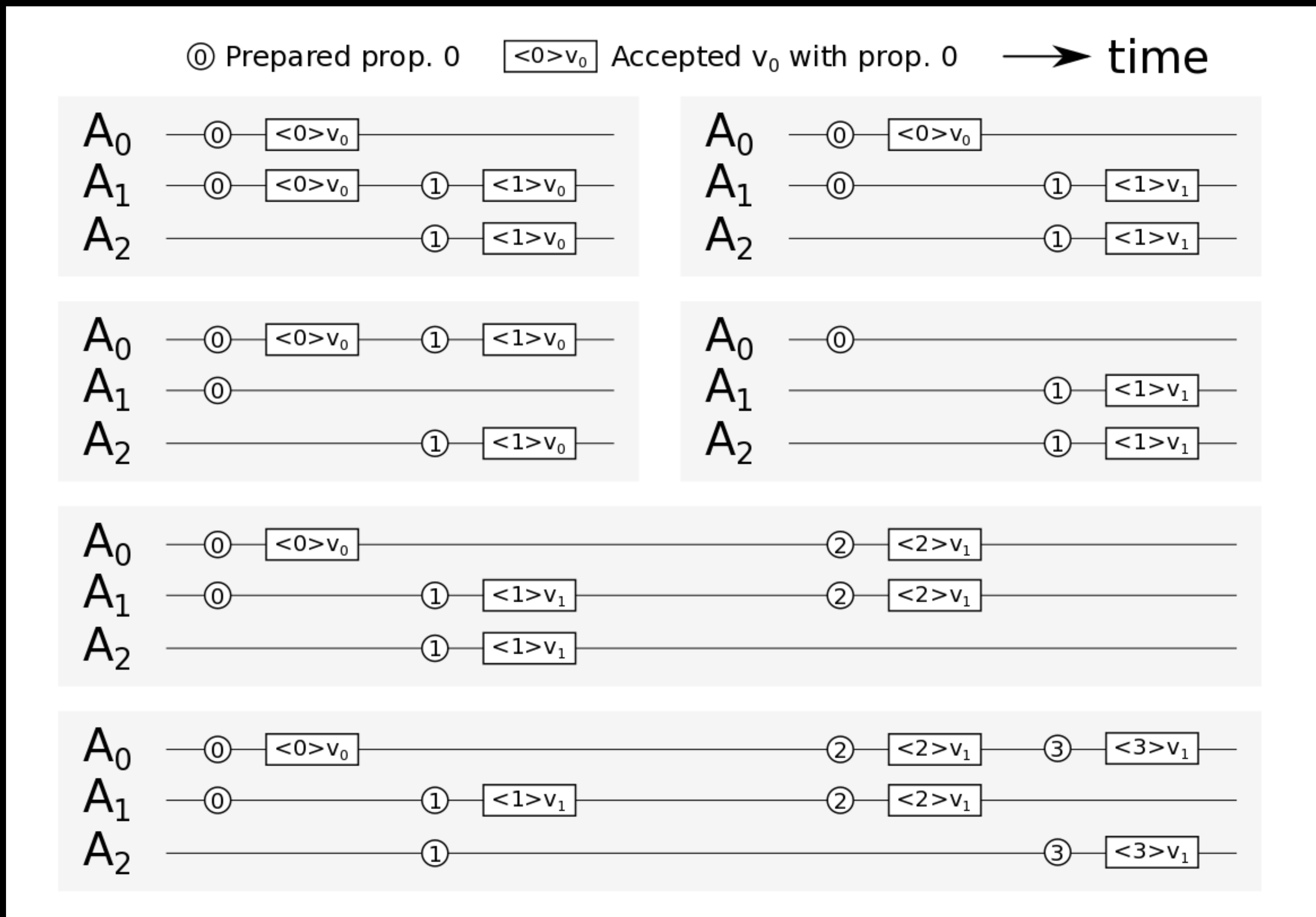
Basic Paxos 实现



Basic Paxos 示例



Basic Paxos 示例



Basic Paxos 证明

For a system with $n_{acceptors} = 2q$ or $2q + 1$ and $n_{majority} = q+1$,
to proof: if $<p_i>v_i$ is accepted by $n_{majority}$ acceptors A during time t_s to t_e, no $<p_j>v_j$ where $v_j \;!= v_i$ can be accepted by maj. after t_e.

=> max resp. prop. id of a_i in $A >= p_i$ at t_e
  => no $p_j < p_i$ can be accepted by maj. after t_e

=> no $p_j > p_i$ can be accepted by a_i in A before a_i accepted $<p_i>v_i$
  => $<p_i>v_i$ is in the set of pre-accepted values P for $p_j > p_i$

to show for $<p_k>v_k$ in P where $p_k > p_i$ and $p_k == $ max prop. id. in P,
$v_k == v_i$
  Proof by contridication:

  Suppose $<p_k>v_k$ is the value accepted by any acceptor where
  $v_k \;!= v_i$ and $p_k > p_i$ and $p_k ==$ min prop.id. in $<p_x>v_x$ satisfies
  $v_x \;!= v_i$ and $p_x > p_i$

  => exists $<p_l>v_l$ where $v_l \;!= v_i$ and $p_l > p_i$ and $p_l < p_k$ accepted
    by some acceptors

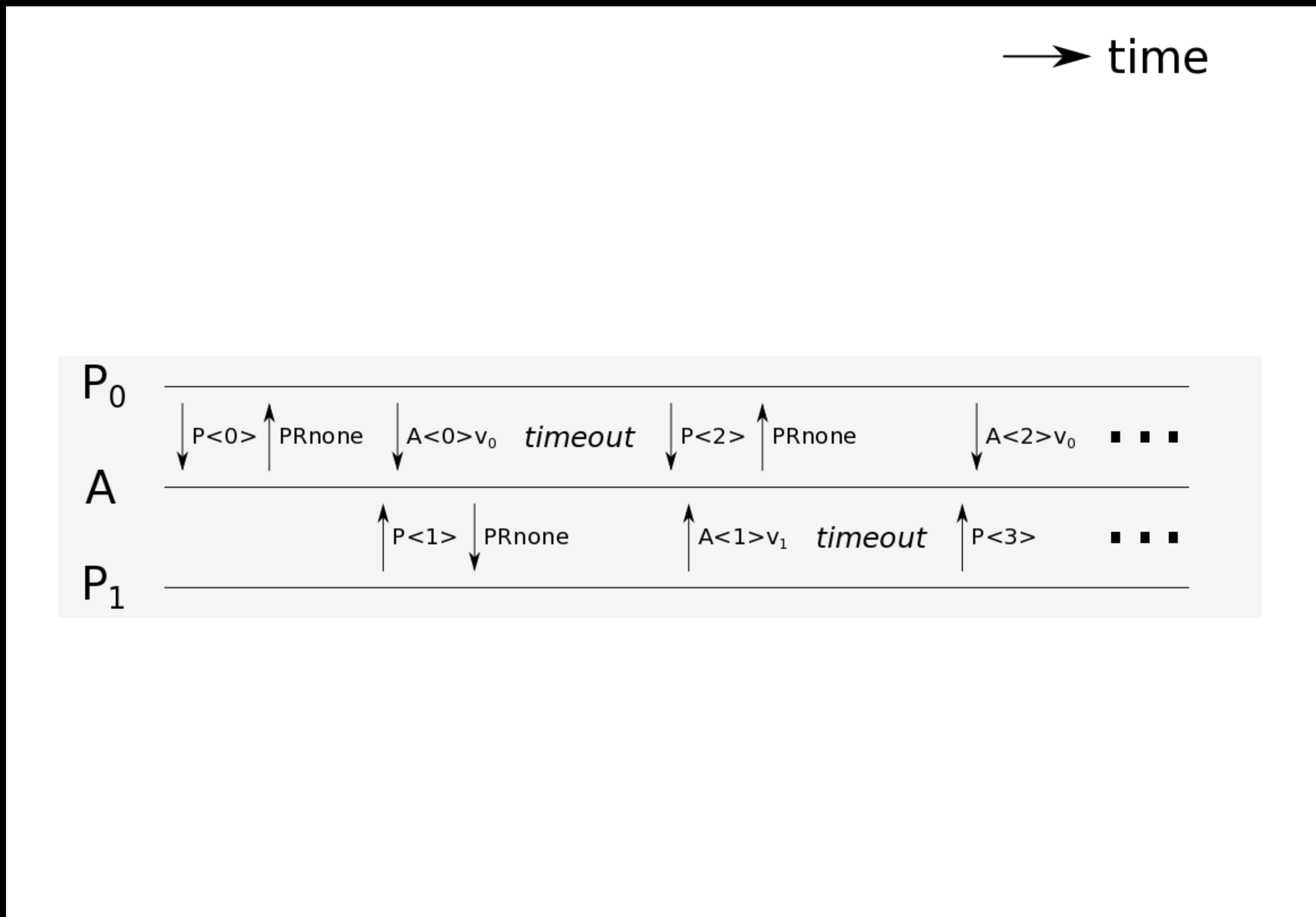
多数派

- 对含 $2q + 1$ acceptors 的系统，多数派为 $q + 1$
  - Prepared by $q + 1$ acceptors
  - Accepted by $q + 1$ acceptors
  - # prepared + # accepted = $2q + 2 > 2q + 1$
- 对含 n acceptors 的系统
  - Prepared by m acceptors
  - Accepted by $n - m + 1$ acceptors
  - # prepared + # accepted = $n + 1 > n$

多数派

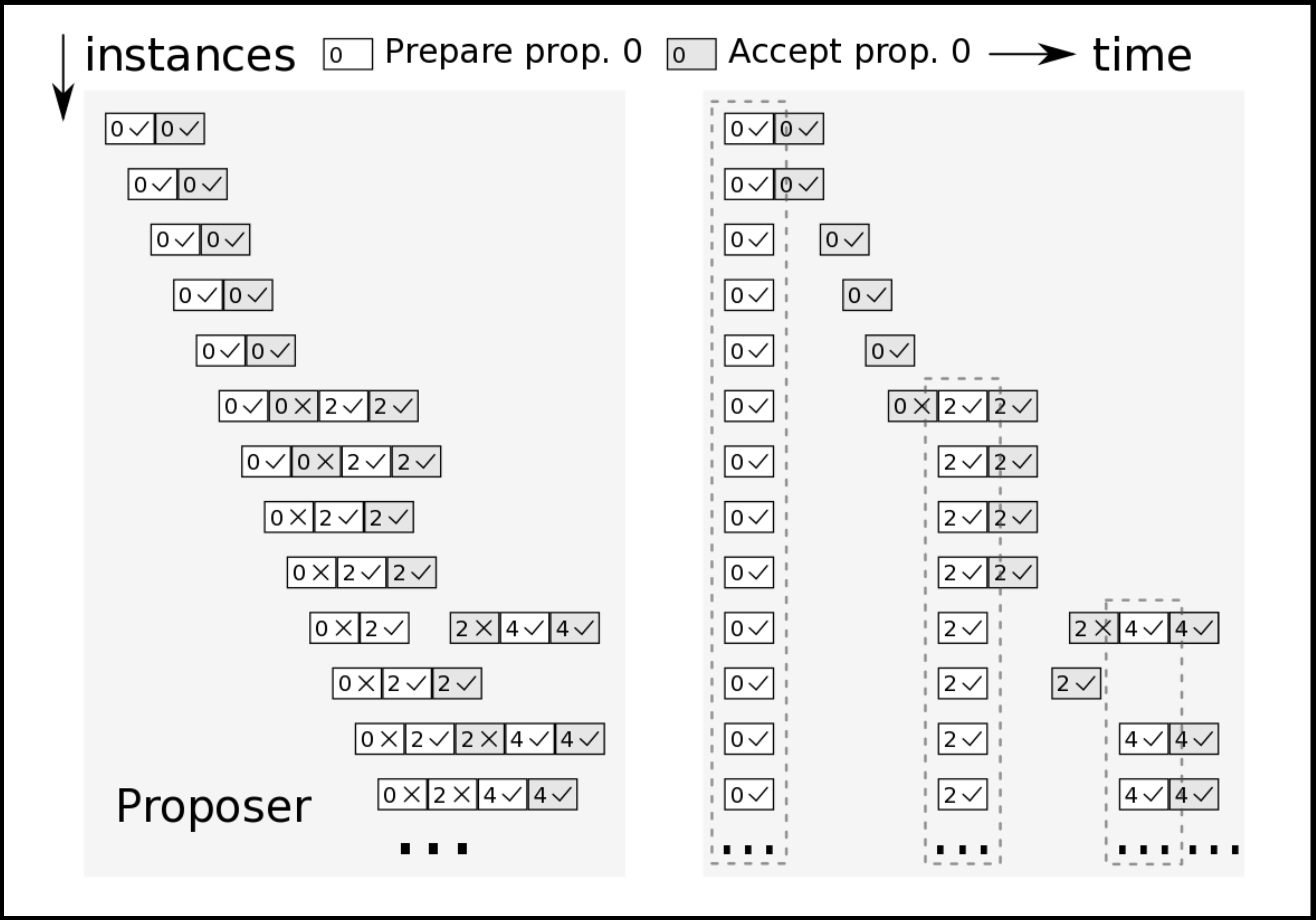
- For n acceptors, prepared by m, accepted by $n - m + 1$
- $1 <= m <= n$
- $m = 1$（强同步数据）
- $m = n$（强同步选主）
- 可用性
  - prepare 需 m 台存活，accept 需 $n - m + 1$ 台存活

活锁



time
P0
A
P1
P<0>
PRnone
A<0>v0
timeout
P<2>
PRnone
A<2>v0
P<1>
PRnone
A<1>v1
timeout
P<3>

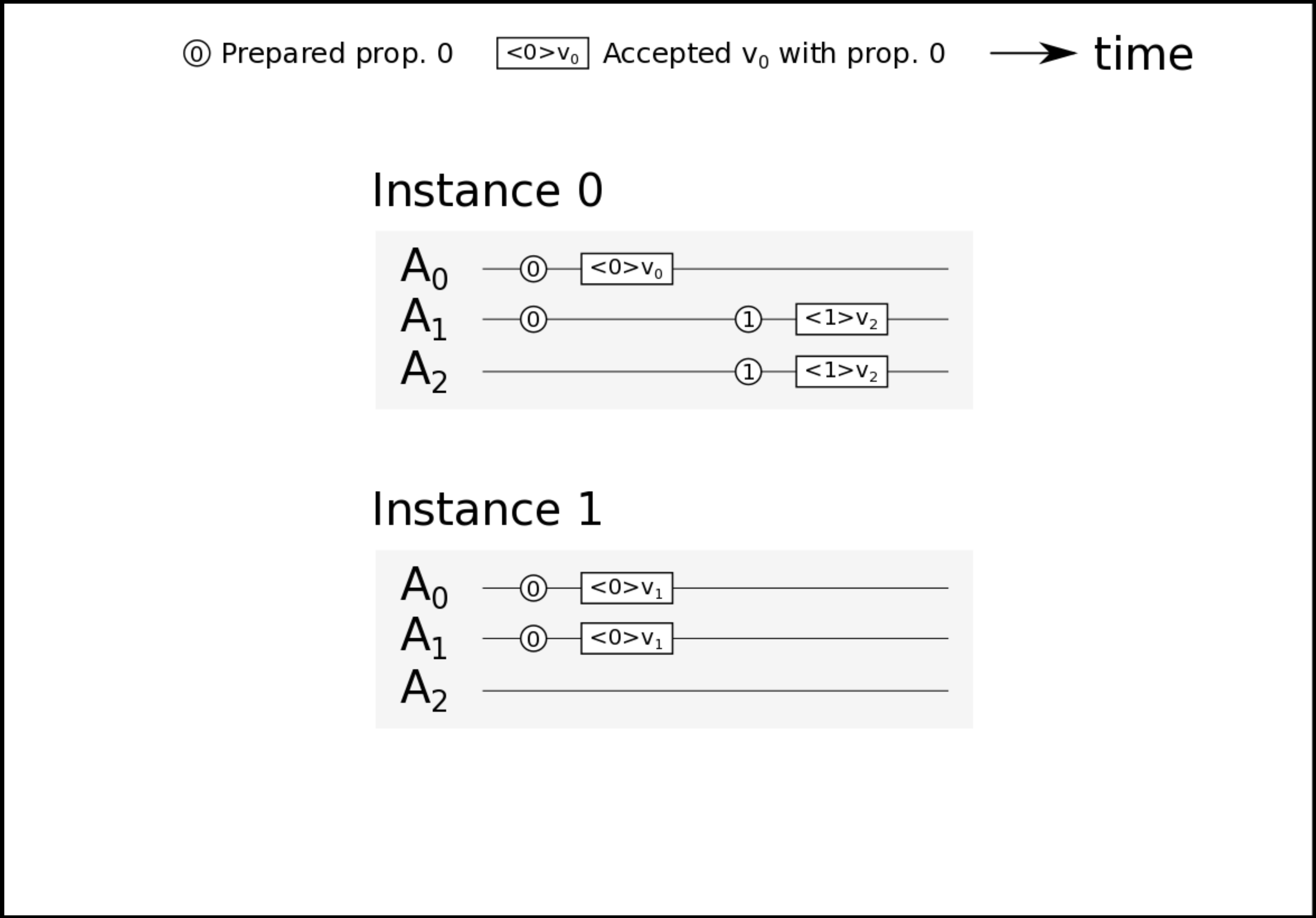
Basic => Multi Paxos



Multi Paxos 时序

- 如果提议 A 被确认后提议 B 才被提出，保证 B 在 A 之后被执行
  - 空洞
  - 空操作

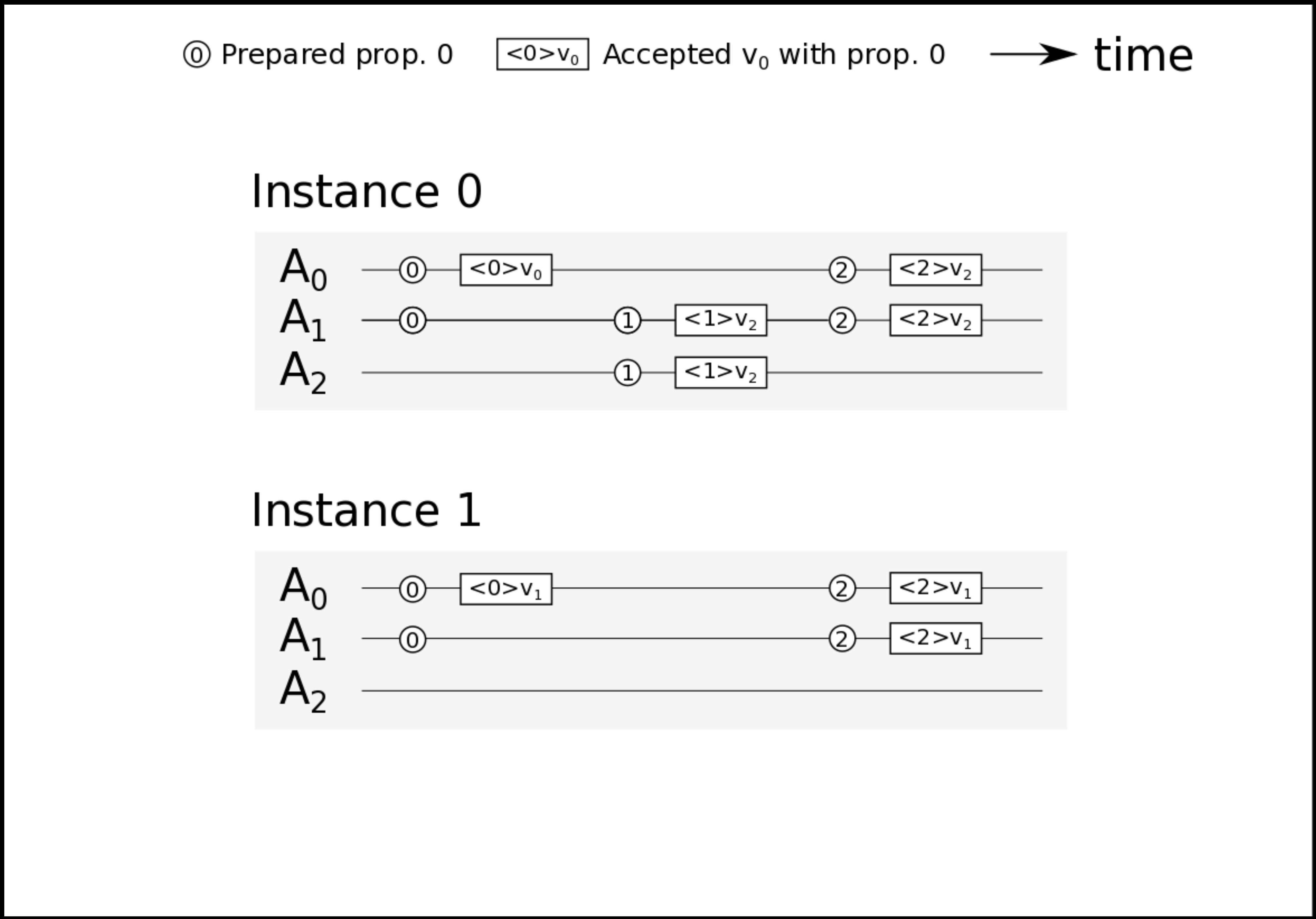
Multi Paxos 时序



更严格的时序

- 请求顺序 vs 执行顺序
  - 超时（不确定的未来状态）
- 单 Proposer Instance ID 分配和冲突（先发先至）
- 多 Proposer 乱序
  - 编码序号（严格增序执行）
  - 客户端 vs 服务端（多节点间后发先至网络）

更严格的时序



0 Prepared prop. 0
<0>v0 Accepted v0 with prop. 0
time
Instance 0
A0
A1
A2
0
<0>v0
2
<2>v2
0
1
<1>v2
2
<2>v2
1
<1>v2
Instance 1
A0
A1
A2
0
<0>v1
2
<2>v1
0
2
<2>v1

成员变更

- 一阶成员（Acceptor）变更
  - 每次只变更一个 Acceptor，任意新旧多数派交集不为空
  - # acceptors $2q$ <=> $2q + 1$
  - # old/new majority = $q + 1$
  - # old/new prepared + # old/new accepted = $2q + 2 > 2q + 1 > 2q$

成员变更的多数派

- 一阶成员（Acceptor）变更
  - # acceptors n => $n + 1$
  - Let prepared by m
    - # new prepared = m
    - # old accepted = $n - m + 1$
    - # new prepared + # old accepted = $n + 1$ == # new acceptors
  - For $n = 2q$, prepared, accepted by m, $n - m + 2$

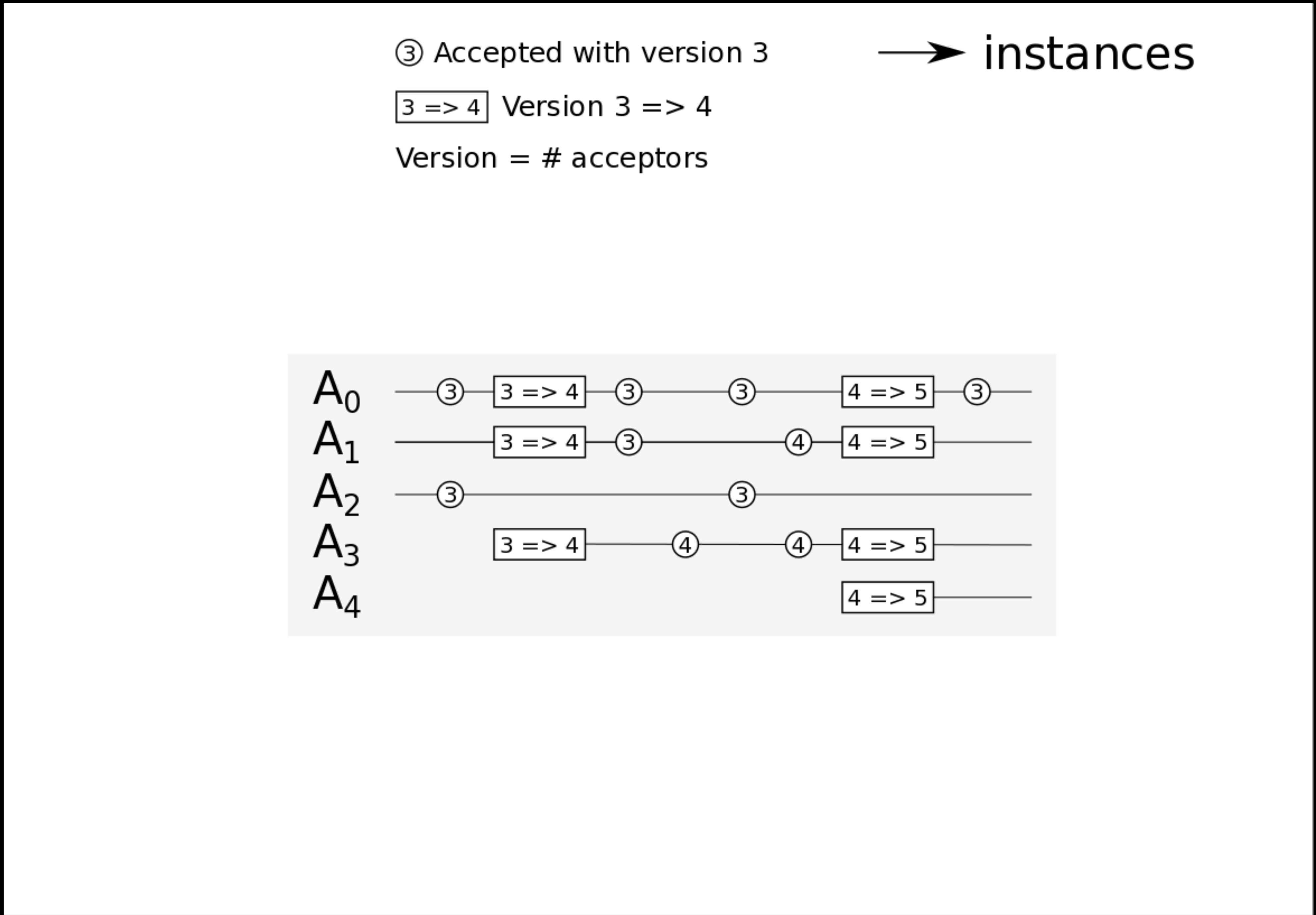
连续成员变更

- 不能同时存在 n, $n + 1$, $n + 2$ 的 acceptor 组
- $n + 1$ => $n + 2$ 前提
  - 多数派接收 # acceptors = $n + 1$
  - 多数派不响应 # acceptors = n（版本）

连续成员变更

- *n* 下确定的值在 *n* + 2 下有效
  - 成员变更并入其它数据（操作）队列
  - 对已执行（已确定）的数据不执行 Paxos

连续成员变更示例



③ Accepted with version 3
3 => 4 Version 3 => 4
Version = # acceptors
instances
A0
A1
A2
A3
A4
3 => 4
4 => 5

Failover

- 多数派存活
- 删除一个成员，添加一个成员

选主

- 在系统稳定情况下，有唯一主
- 性能优化
- 耦合

检查点

- 无限长操作序列
- 状态机一致，检查点一致

参考

- The part-time parliament
- Paxos made simple
- https://github.com/tencent-wechat/phxpaxos
- http://oceanbase.org.cn/?p=90
- http://oceanbase.org.cn/?p=111
- http://oceanbase.org.cn/?p=160

Q&A

- 谢谢！