

# 讲师介绍



Hash    QQ: 805921455

从事Java软件研发近十年。

前新浪支付核心成员、

咪咕视讯(中国移动)项目经理、

对分布式架构、高性能编程有深入的研究。

明天，你一定会感谢今天奋力拼搏的你

# Zookeeper 集群

## 分布式系统开发技术

# 目录

## 课程安排



01

ZK集群简介

集群搭建、部署、监控



02

ZK集群的灵魂-Leader

Leader选举



03

ZK集群工作原理

协议核心、崩溃恢复、  
数据同步



04

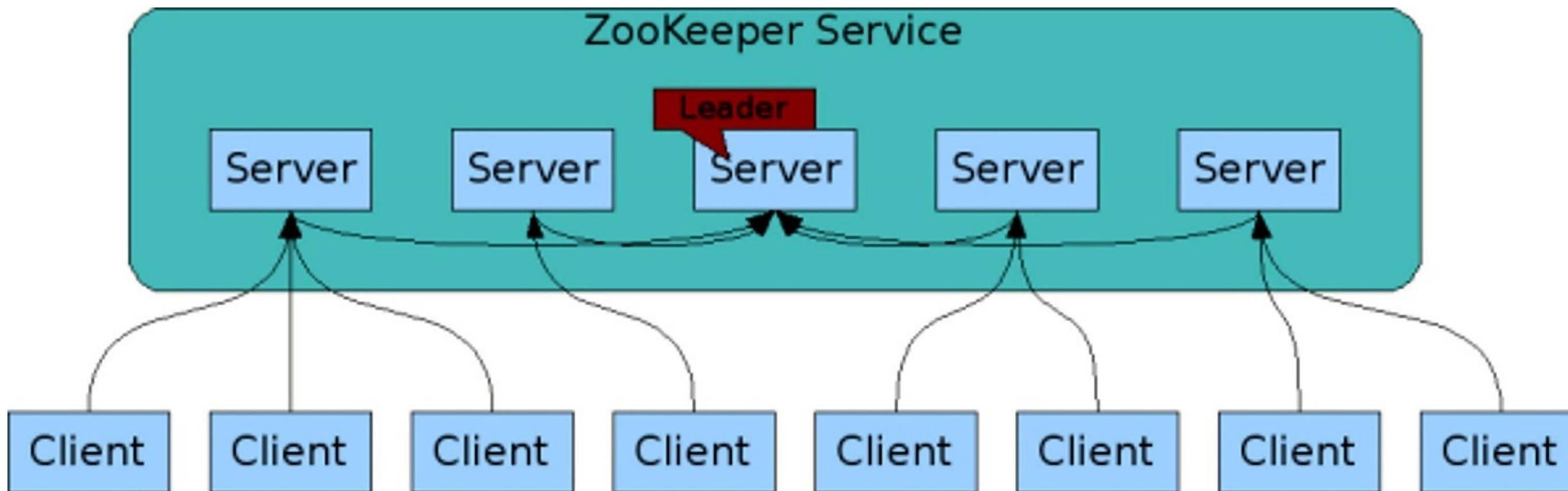
总结知识点

本堂课知识总结



## ZK集群简介

# ZooKeeper集群



- 可靠的ZooKeeper服务
- 只要集群的大多数都准备好了，就可以使用这项服务
- 容错集群设置至少需要三个服务器，强烈建议使用奇数个服务器
- 建议每个服务运行在单独的机器上

# ZooKeeper集群搭建

## 配置

```
tickTime=2000
dataDir=/var/lib/zookeeper/
clientPort=2181
initLimit=5
syncLimit=2
server.1=zoo1:2888:3888
server.2=zoo2:2888:3888
server.3=zoo3:2888:3888
```

### initLimit

集群中的follower服务器(F)与leader服务器(L)之间完成初始化同步连接时能容忍的最多心跳数（tickTime的数量）。如果zk集群环境数量确实很大，同步数据的时间会变长，因此这种情况下可以适当调大该参数。

### syncLimit

集群中的follower服务器与leader服务器之间请求和应答之间能容忍的最多心跳数（tickTime的数量）。

## 集群节点

server.id=host:port:port

**id**，通过在各自己的dataDir目录下创建一个名为myid的文件来为每台机器赋予一个服务器id。

**两个端口号**，第一个跟随者用来连接到领导者，第二个用来选举领导者。

# ZooKeeper集群搭建

## 创建 myid 文件

在dataDir目录下创建myid文件

```
1
```

一行只包含机器id的文本。id在集群中必须是惟一的，其值应该在1到255之间。如服务器1的id为“1”

# 连接ZooKeeper集群

集群的所有节点都可以提供服务，客户端连接时，连接串中可以指定多个或全部集群节点的连接地址。当一个节点不通时，客户端将自动切换另一个节点。

```
"10.168.1.23:2181,10.168.1.24:2181,10.168.1.24:2181"
```



# ZooKeeper集群监控

## 方式一，四字命令


示例：echo ruok | nc 127.0.0.1 2181

<i>conf</i>	输出相关服务配置的详细信息。
<i>cons</i>	列出所有连接到服务器的客户端的完全的连接 / 会话的详细信息。包括“接受 / 发送”的包数量、会话 id、操作延迟、最后的操作执行等等信息。
<i>crst</i>	重置当前这台服务器所有连接/会话的统计信息
<i>dump</i>	列出未经处理的会话和临时节点。
<i>envi</i>	输出关于服务环境的详细信息（区别于 <i>conf</i> 命令）。
<i>reqs</i>	列出未经处理的请求
<i>ruok</i>	测试服务是否处于正确状态。如果确实如此，那么服务返回“imok”，否则不做任何相应。
<i>srst</i>	重置服务器统计信息
<i>srvr</i>	列出服务器详细的信息，zk版本、接收/发送包数量、连接数、模式（leader/follower）、节点总数。
<i>stat</i>	输出关于性能和连接的 <a href="#">客户端</a> 的列表。
<i>wchs</i>	列出服务器 watch 的详细信息。
<i>wchc</i>	通过 session 列出服务器 watch 的详细信息，它的输出是一个与watch 相关的会话的列表。
<i>wchp</i>	通过路径列出 <a href="#">服务器</a> watch 的 <a href="#">详细</a> <a href="#">信息</a> 。它输出一个与 session相关的 <a href="#">路径</a> 。
<i>mntr</i>	列出集群的健康状态。包括“接受/发送”的包数量、操作延迟、当前服务模式（leader/follower）、节点总数、watch总数、临时节点总数。

# ZooKeeper集群监控

## 方式二，JMX

JMX (Java Management Extensions) Java 管理扩展，是一个为应用程序、设备、系统等植入管理功能的框架。



Java 监视和管理控制台 - pid: 14012 org.apache.zookeeper.server.quorum.QuorumPeerM...

连接(C) 窗口(W) 帮助(H)

概览 内存 线程 类 VM 概要 MBean

java.nio  
java.util.logging  
log4j  
org.apache.ZooKeeperService  
ReplicatedServer id1  
属性  
Name  
QuorumSize  
replica.1  
属性  
State  
MaxSessionTimeout  
MinSessionTimeout  
SyncLimit  
TickTime  
InitLimit  
QuorumAddress  
Tick  
ElectionType  
MaxClientCnxnsPerHost  
Name  
StartTime  
Leader  
属性  
操作  
InMemoryDataTree  
属性  
操作  
approximateDataSize  
countEphemerals  
replica.2  
属性  
Name  
QuorumAddress  
replica.3  
属性  
Name

属性值

名称	值
ElectionType	3
InitLimit	10
MaxClientCnxnsPerHost	60
MaxSessionTimeout	40000
MinSessionTimeout	4000
Name	replica.1
QuorumAddress	/127.0.0.1:2881
StartTime	Wed Mar 27 11:20:33 CST 2019
State	leading
SyncLimit	5
Tick	767
TickTime	2000

刷新(R)

JAVA\_HOME/bin/ jconsole

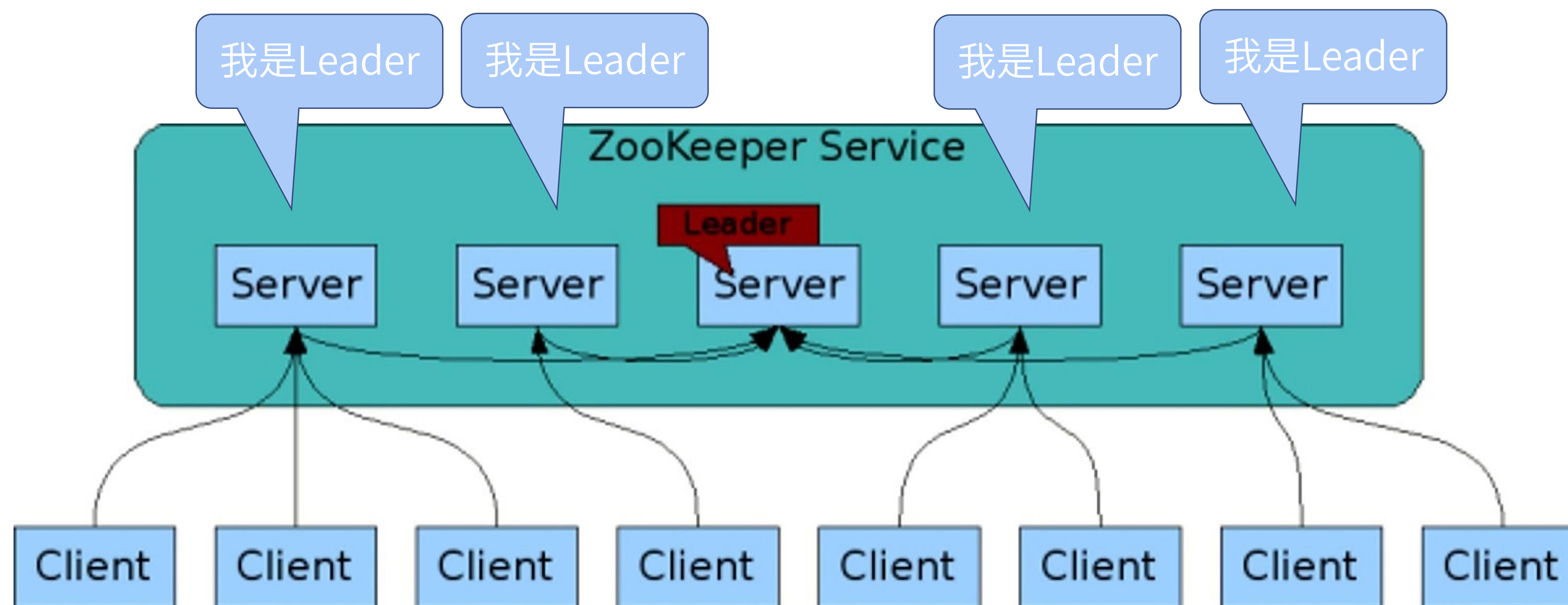


---

## ZK集群的灵魂-Leader

# 每个人都是Leader?

到底谁是Leader?



每台服务器都可能成为Leader





# Paxos算法

## 分布式一致性算法，Paxos

Paxos算法是Leslie Lamport于1990年提出的一种基于消息传递且具有高度容错特性的一致性算法，是分布式一致性中的经典算法。

Phase 1. (a) A proposer selects a proposal number  $n$  and sends a prepare request with number  $n$  to a majority of acceptors.

(b) If an acceptor receives a prepare request with number  $n$  greater than that of any prepare request to which it has already responded, then it responds to the request with a promise not to accept any more proposals numbered less than  $n$  and with the highest-numbered proposal (if any) that it has accepted.

Phase 2. (a) If the proposer receives a response to its prepare requests (numbered  $n$ ) from a majority of acceptors, then it sends an accept request to each of those acceptors for a proposal numbered  $n$  with a value  $v$ , where  $v$  is the value of the highest-numbered proposal among the responses, or is any value if the responses reported no proposals.

(b) If an acceptor receives an accept request for a proposal numbered  $n$ , it accepts the proposal unless it has already responded to a prepare request having a number greater than  $n$ .



# Paxos算法

## 分布式一致性算法，Paxos

P1a: 提议发起者选择一个提案编号n，并且给大多数接受者发送一个带有编号n的提案预请求。

P1b: 如果接受者收到了一个编号n提案预请求，请求的编号n大于前面已经响应过的预请求编号，这时接受者作出响应，承诺不再接受比编号n小的提案预请求，并且如果存在自己接受过的最高编号提案，则在响应中带上。

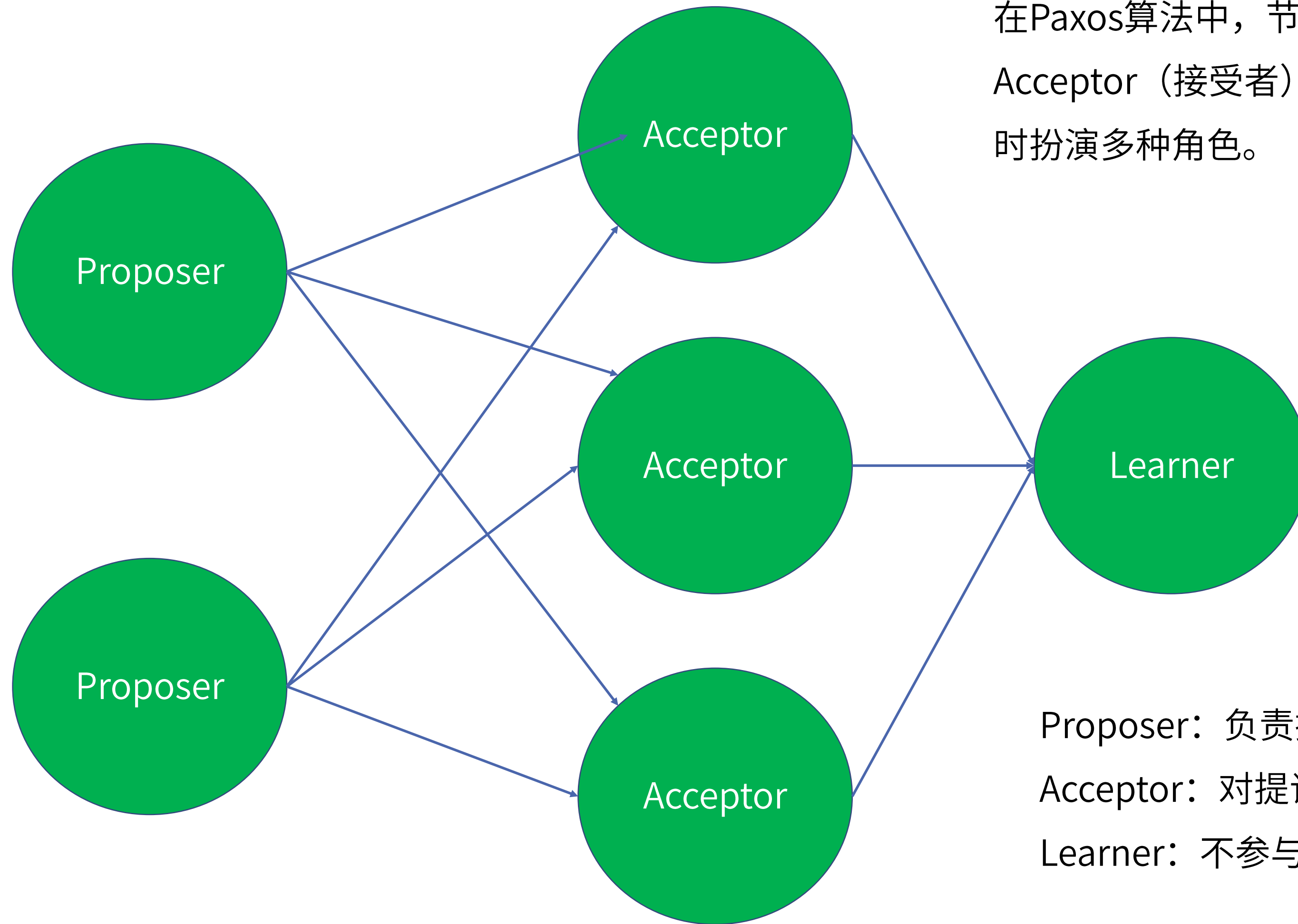
P2a: 如果提议发起者接收到了大多数接受者对于编号n预请求的响应，这时会给这些接受者的每一个服务发送接受请求，接受请求内容为，编号n的提案并带上value值v。v的取值从哪里选择呢？如果接受者响应中有其他提议内容，则取接受者响应中取最高编号对应的值，如果响应中没有其他提议内容，则可以是任意的值。

P2b: 如果接受者受到一个编号n的提案接受请求，它接受该提案；如果它已经准备对大于编号n的预请求作出响应，则不接受编号n的提案。



# Paxos中的角色

## Paxos中的角色



在Paxos算法中，节点可以分为：Proposer（提议者）、Acceptor（接受者）、Learner（学习者）三种角色，可以同时扮演多种角色。

Proposer: 负责提议，提出想要达成一致的value;  
Acceptor: 对提议投票，决定是否接受此value;  
Learner: 不参与投票，只接收投票结果。

# Paxos的约束

## Paxos的约束

最终只有一个提议值会被选择，只有被选择的提议值才会被Learner节点学习。

最终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。 **(少数服从多数)**

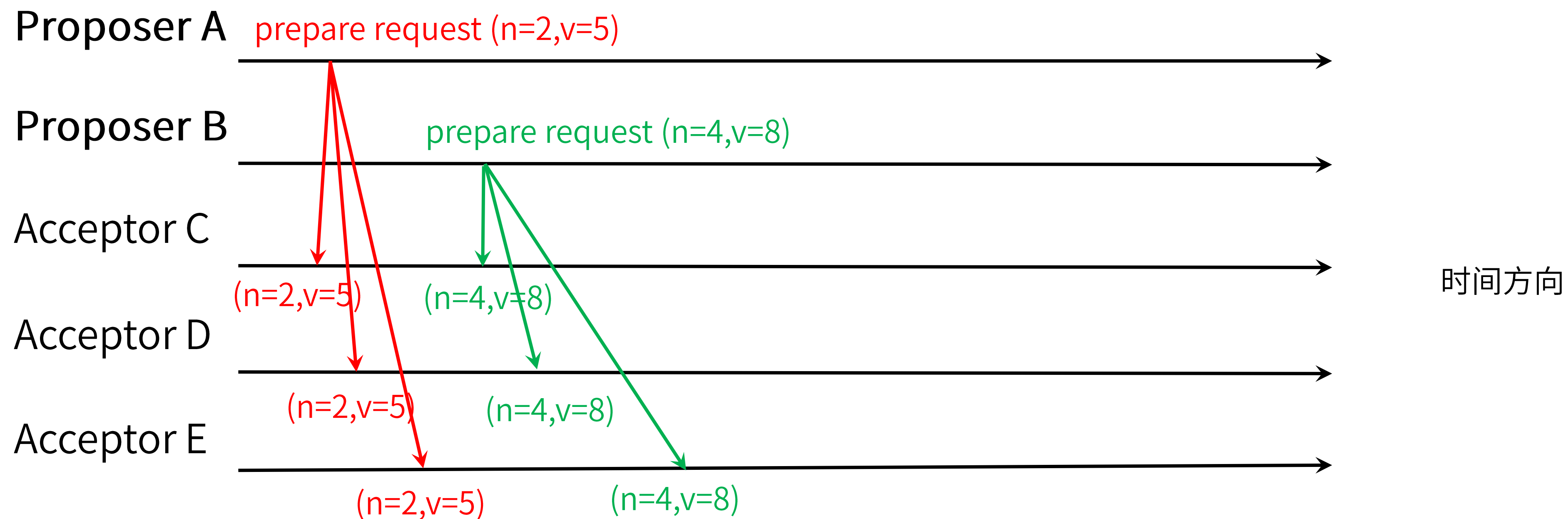


# Paxos的流程

## Paxos的流程 prepare a

Proposer会发送两种类型的消息给Acceptors: prepare (准备) 和accept (接收) 请求。

Proposer发送的提议请求由两部分组成, (n,v), n为序号 (不可重复, 具有唯一性), v为提议值。

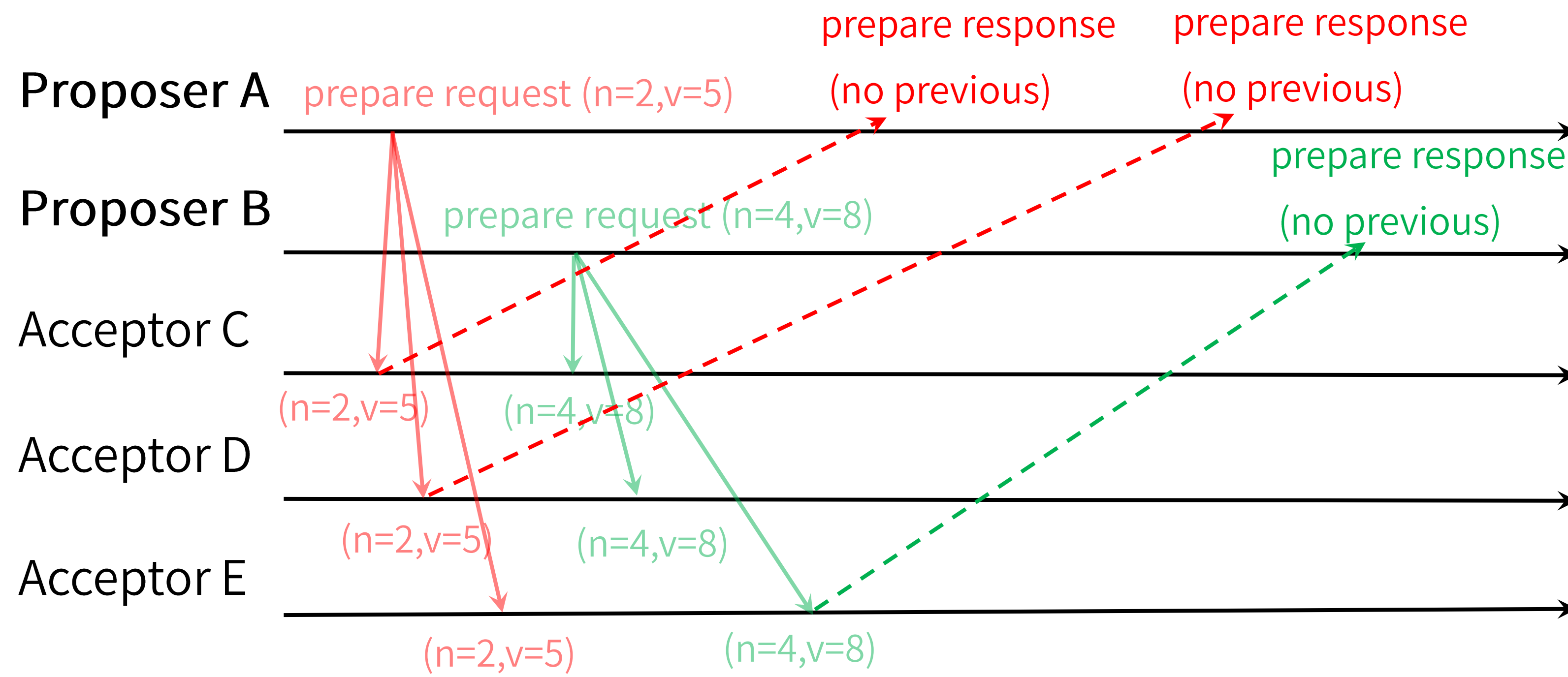


Proposer A、B都可以发送prepare提议请求, Acceptor C、D先接收到Proposer A的请求, Acceptor E先接收到Proposer B的请求。

# Paxos的流程

## Paxos的流程 prepare b

如果Acceptor接收到一个prepare提议请求(n1,v1)，并且之前还未接收过任何提议请求，会发送一个提议请求的响应，设置当前收到的提议为(n1,v1)，保证不再接收序号小于n1的提议请求。

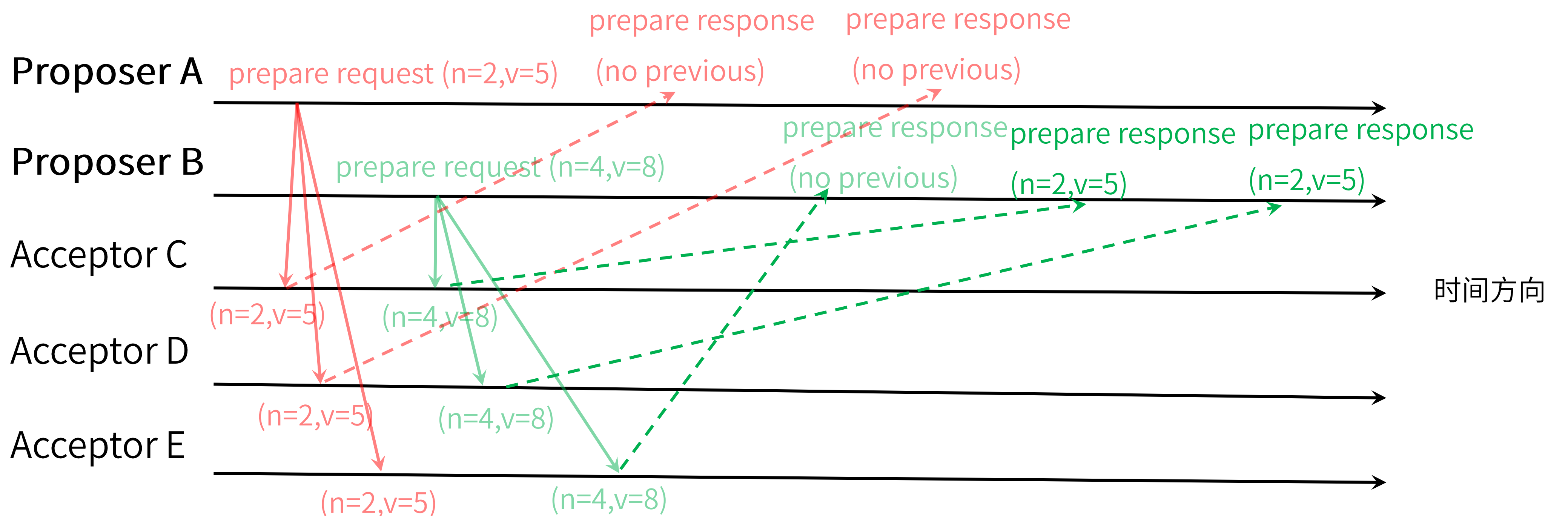


时间方向

# Paxos的流程

## Paxos的流程 accept a

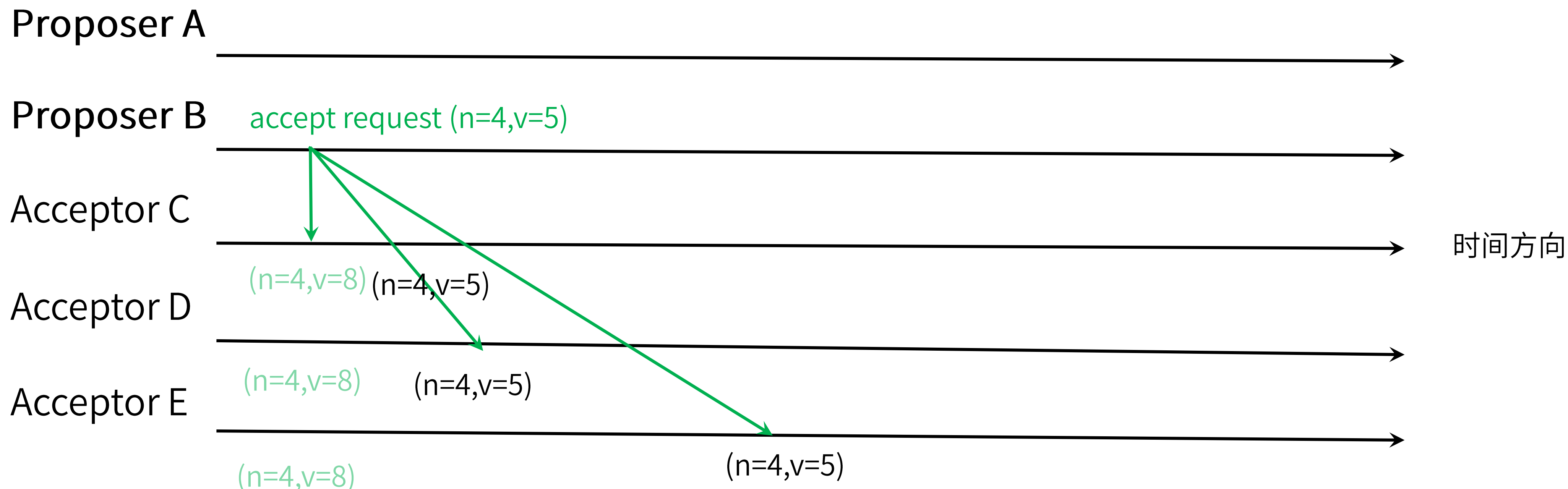
随后，Acceptor C、D会收到Proposer B的提议请求( $n=4, v=8$ )，先前的提议请求( $n=2, v=5$ )，由于 $4 > 2$ ，因此发送( $n=2, v=5$ )的提议响应，设置当前接收到的提议为( $n=4, v=8$ )，并且保证不再接收序号小于4的提议请求；Acceptor E先收到Proposer B的提议请求( $n=4, v=8$ )，后收到Proposer A的提议请求( $n=2, v=5$ )，由于 $2 < 4$ ，因此抛弃该提议请求。



# Paxos的流程

## Paxos的流程 accept b

一旦Proposer收到超过半数Acceptor所发的prepare提议响应，便会向所有Acceptor发送一个accept提议请求。

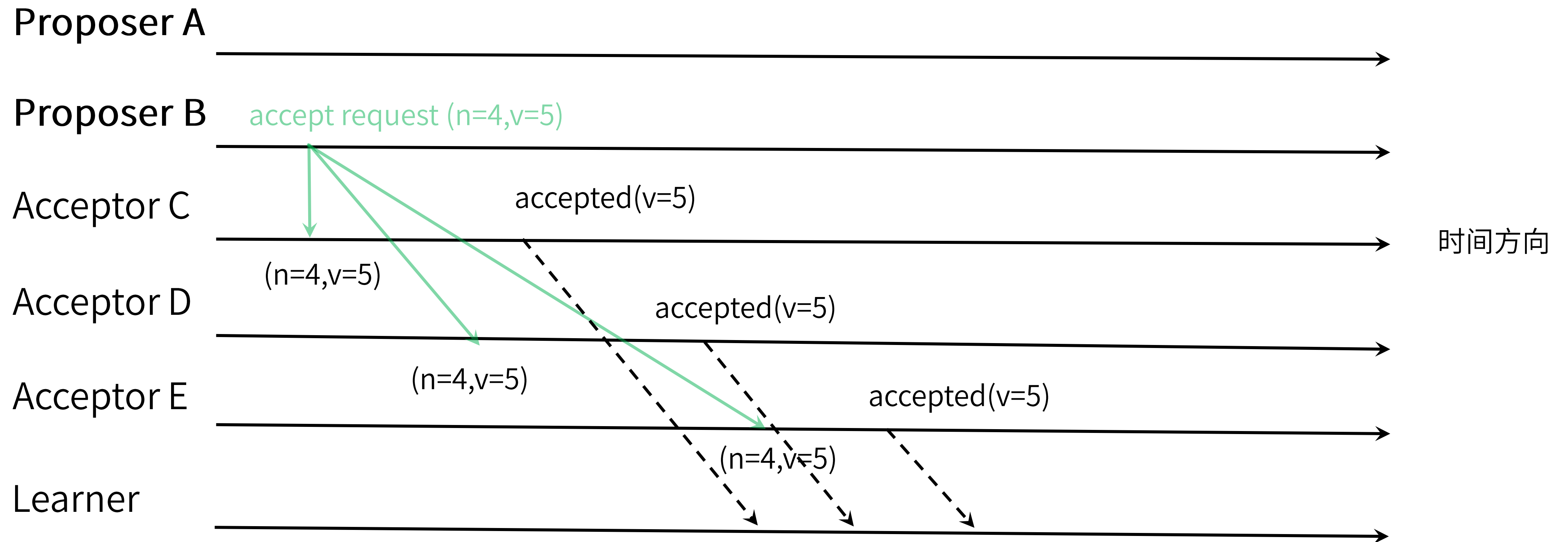


Proposer A收到两个提议响应(no previous)后，发送accept提议请求(n=2,v=5)，会被所有Acceptor丢弃；  
Proposer B收到两个提议响应(n=2,v=5)后，发送accept提议请求(n=4,v=5)，需要注意：n=4是最初Proposer B使用的序号，v=5不是初始值，而是提议响应中更高的v值。

# Paxos的流程

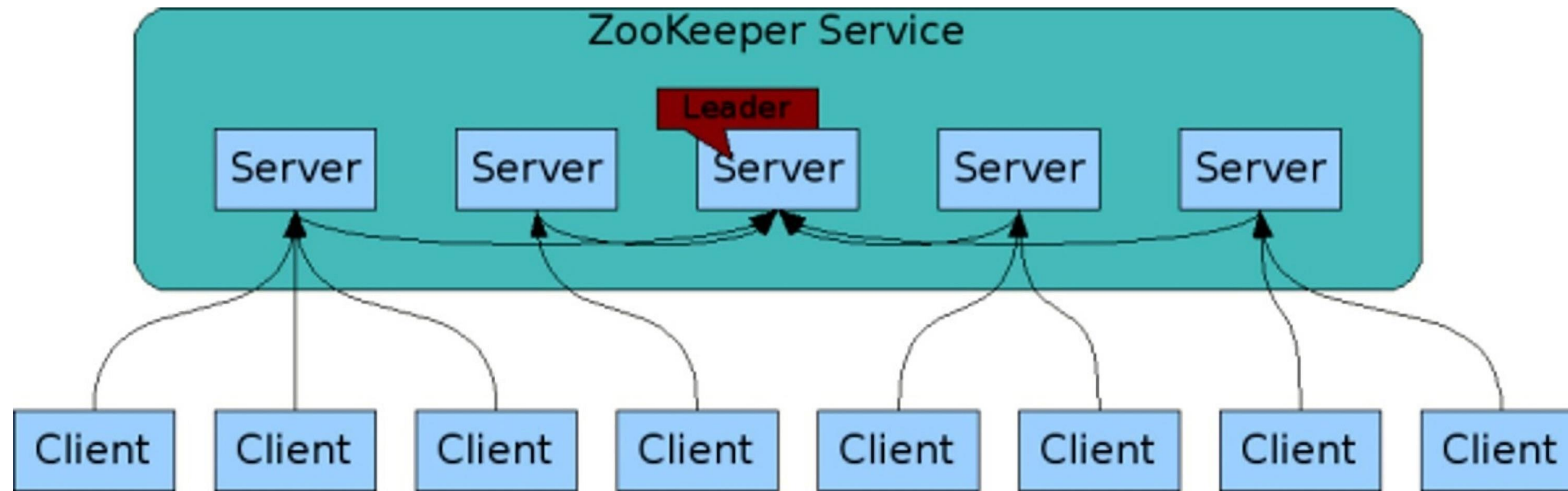
## Paxos的流程 learner

Acceptor C、D、E收到accept提议请求后，会通知所有Learner





# ZooKeeper集群Leader选举



对选举Leader的要求：

- 选出的Leader节点上要持有最高的zxid
- 过半数节点同意

内置实现的选举算法：

- LeaderElection
- FastLeaderElection 默认的
- AuthFastLeaderElection

# ZooKeeper集群Leader选举机制概念

## 选举机制中的概念：

- 服务器id myid
- 事务id，服务器中存放的最大Zxid
- 逻辑时钟，发起的投票轮数计数
- 选举状态：
  - LOOKING，竞选状态。
  - FOLLOWING，随从状态，同步leader状态，参与投票。
  - OBSERVING，观察状态,同步leader状态，不参与投票。
  - LEADING，领导者状态。

## 选举消息内容：

- 服务器id
- 事务id
- 逻辑时钟
- 选举状态

# ZooKeeper集群Leader选举算法

选举算法：

1. 每个服务实例均发起选举自己为领导者的投票（自己的投给自己）；
2. 其他服务实例收到投票邀请时，比较发起者的数据事务ID是否比自己最新的事务ID大，大则给它投一票，小则不投票给它，相等则比较发起者的服务器ID，大则投票给它；
3. 发起者收到大家的投票反馈后，看投票数（含自己的）是否大于集群的半数，大于则胜出，担任领导者；未超过半数且领导者未选出，则再次发起投票。

胜出条件：

投票赞成数大于半数则胜出的逻辑。



# ZooKeeper集群Leader选举流程示例说明

有5台服务器，每台服务器均没有数据，它们的编号分别是1,2,3,4,5,按编号依次启动，它们的选择举过程如下：

- 服务器1启动，给自己投票，然后发投票信息，由于其它机器还没有启动所以它收不到反馈信息，服务器1的状态一直属于Looking。
- 服务器2启动，给自己投票，同时与之前启动的服务器1交换结果，由于服务器2的编号大所以服务器2胜出，但此时投票数没有大于半数，所以两个服务器的状态依然是LOOKING。
- 服务器3启动，给自己投票，同时与之前启动的服务器1,2交换信息，由于服务器3的编号最大所以服务器3胜出，此时投票数正好大于半数，所以服务器3成为领导者，服务器1,2成为小弟。
- 服务器4启动，给自己投票，同时与之前启动的服务器1,2,3交换信息，尽管服务器4的编号大，但之前服务器3已经胜出，所以服务器4只能成为小弟。
- 服务器5启动，后面的逻辑同服务器4成为小弟。

# 问题

有3台zk服务器，编号1,2,3 按编号依次启动，谁会成为Leader?

有3台zk服务器，编号是1,2,3 按编号依次启动，谁会成为Leader?

# 谢谢观看