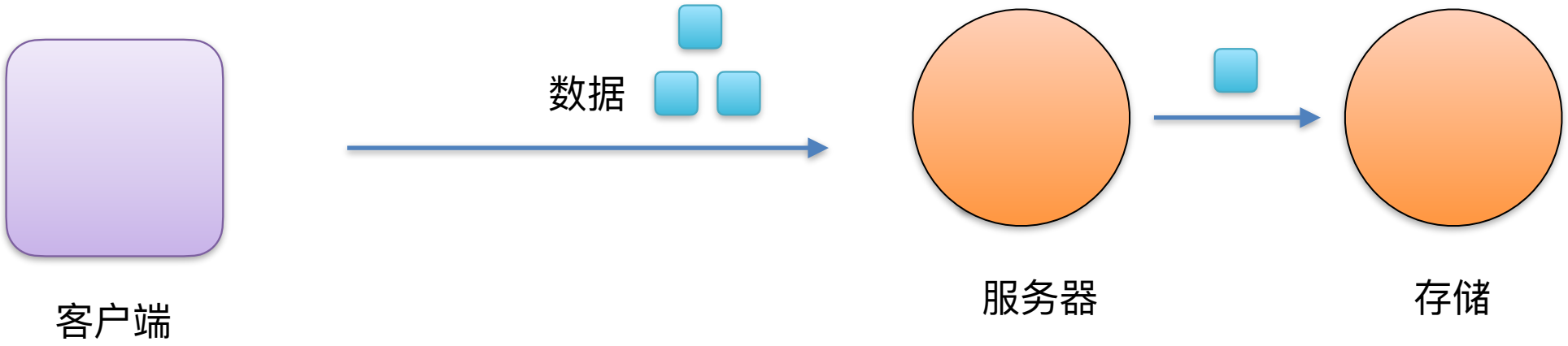
中国科学技术大学
University of Science and Technology of China

Raft—consensus algorithm

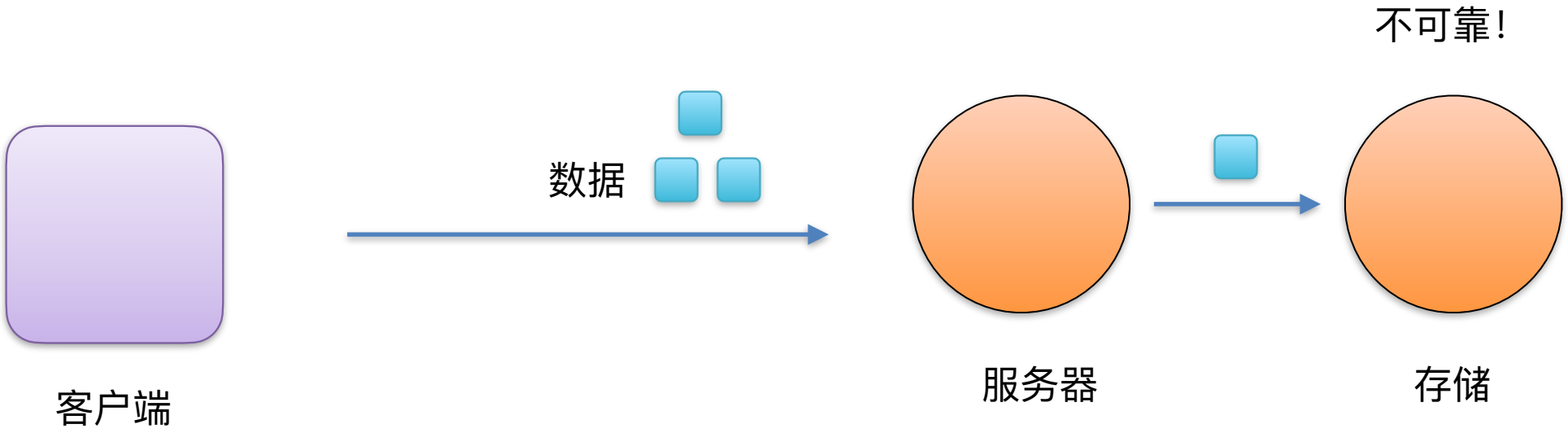
马子杰

Introduction

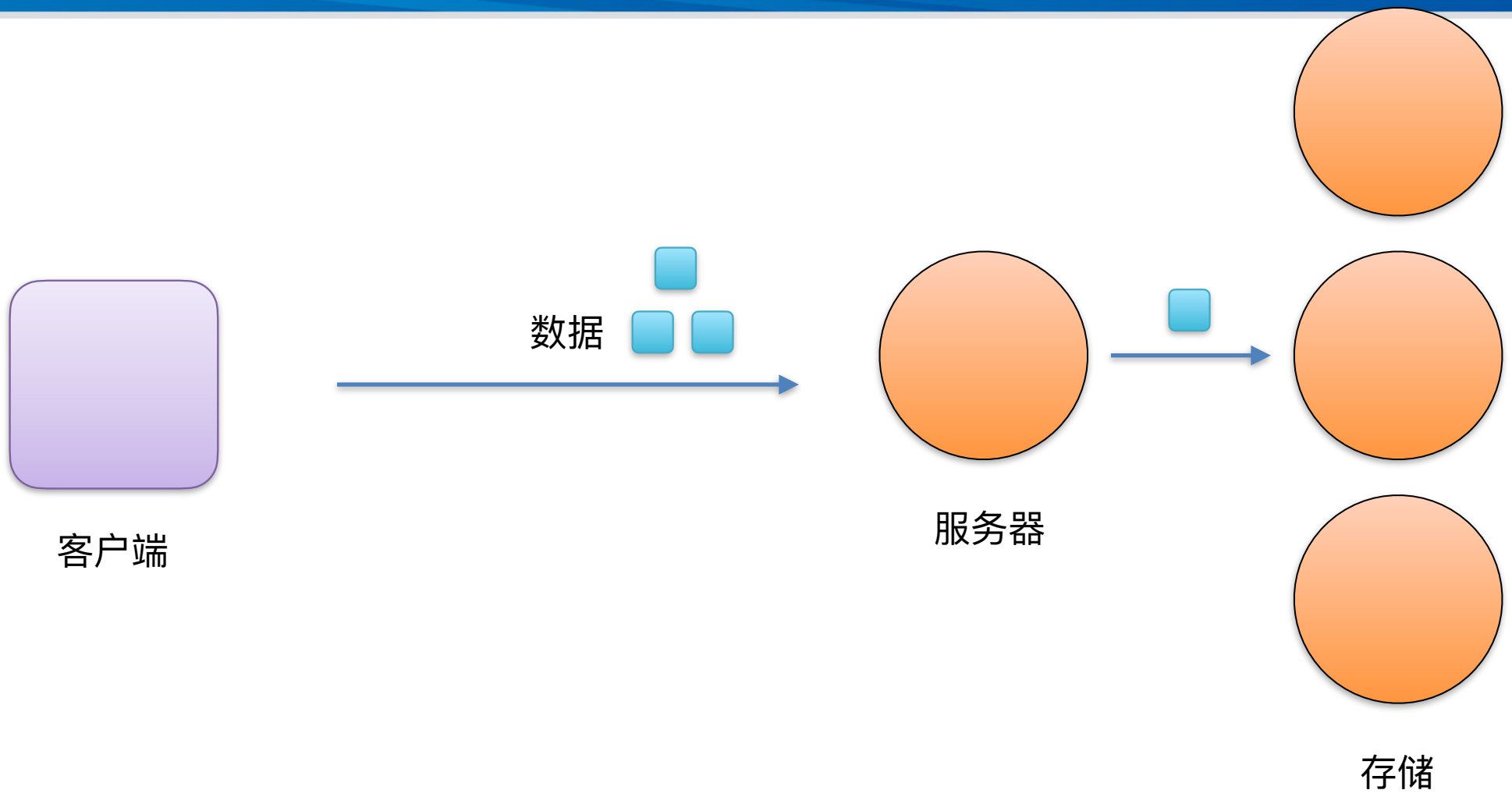
数据备份问题



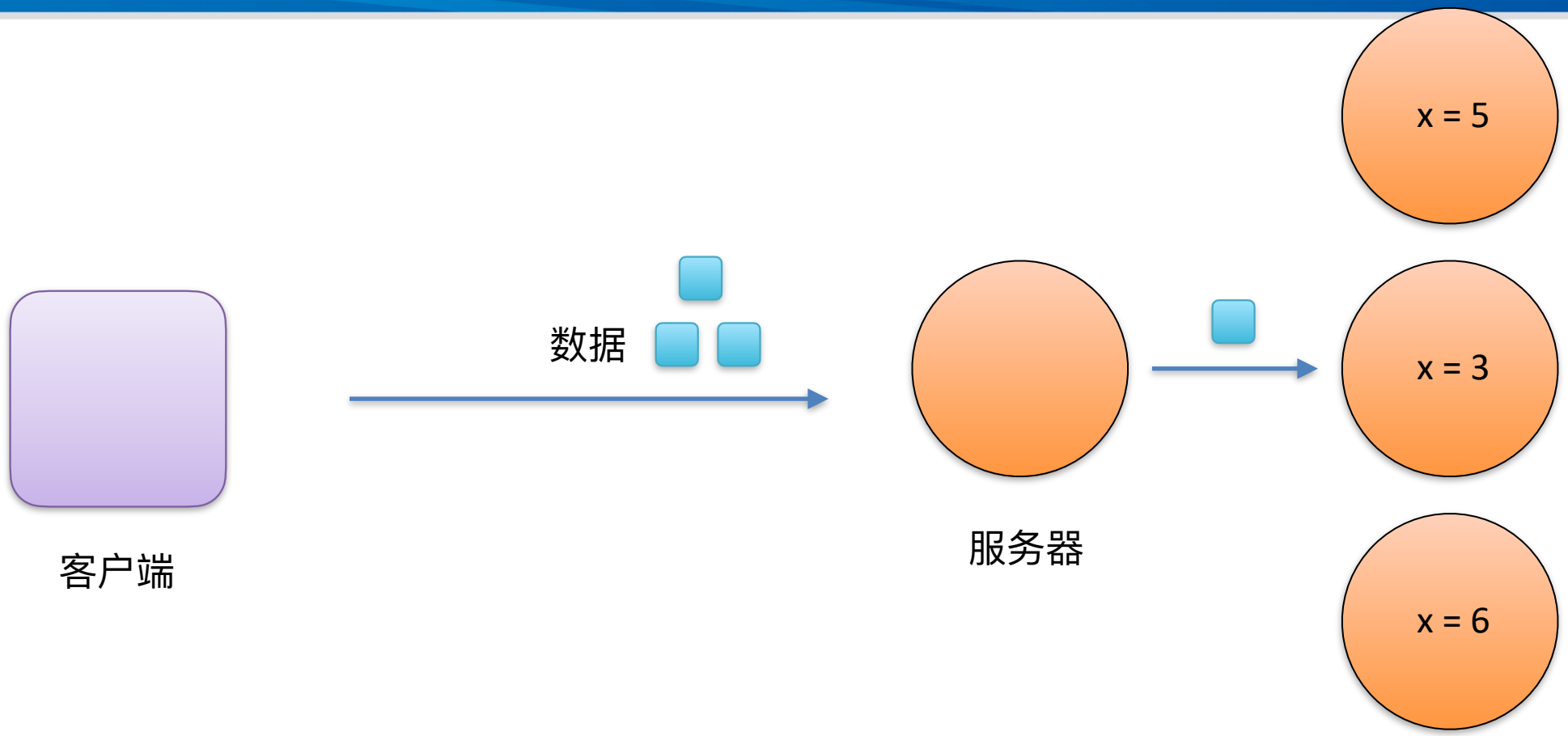
数据备份问题



数据备份问题

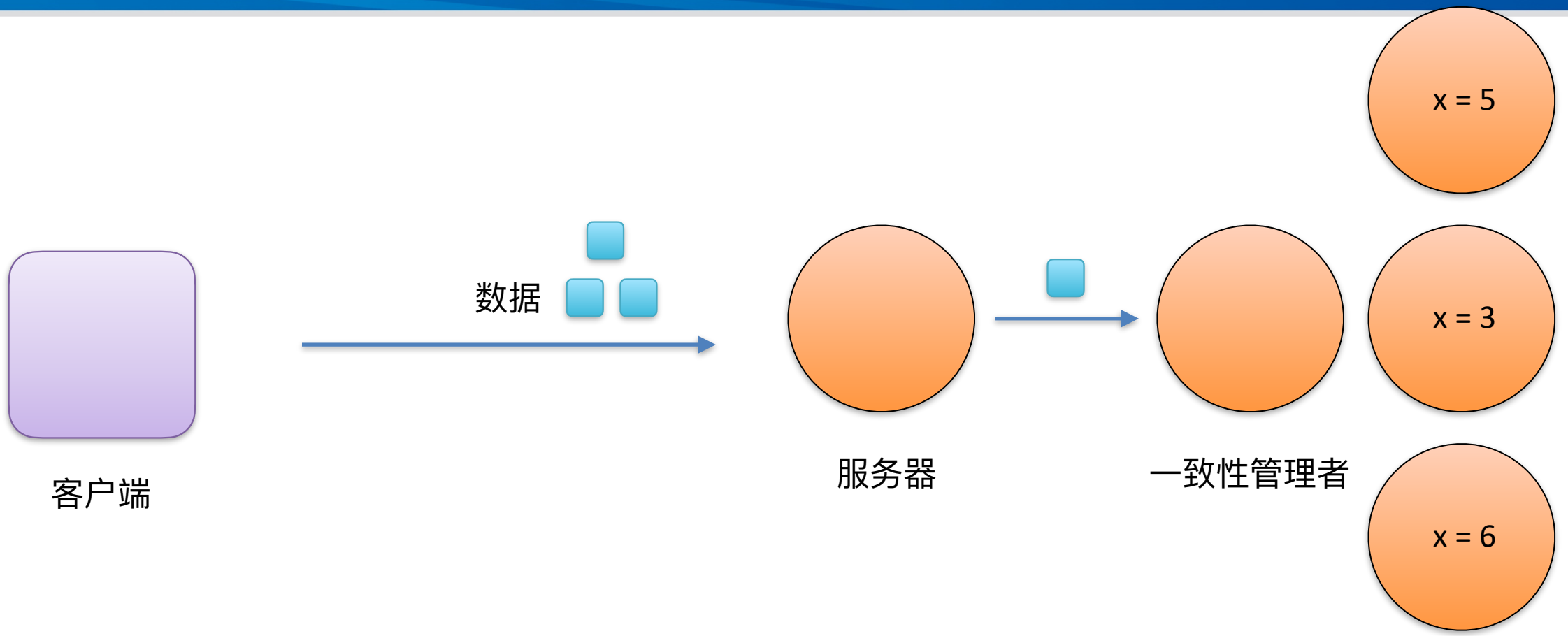


数据备份问题



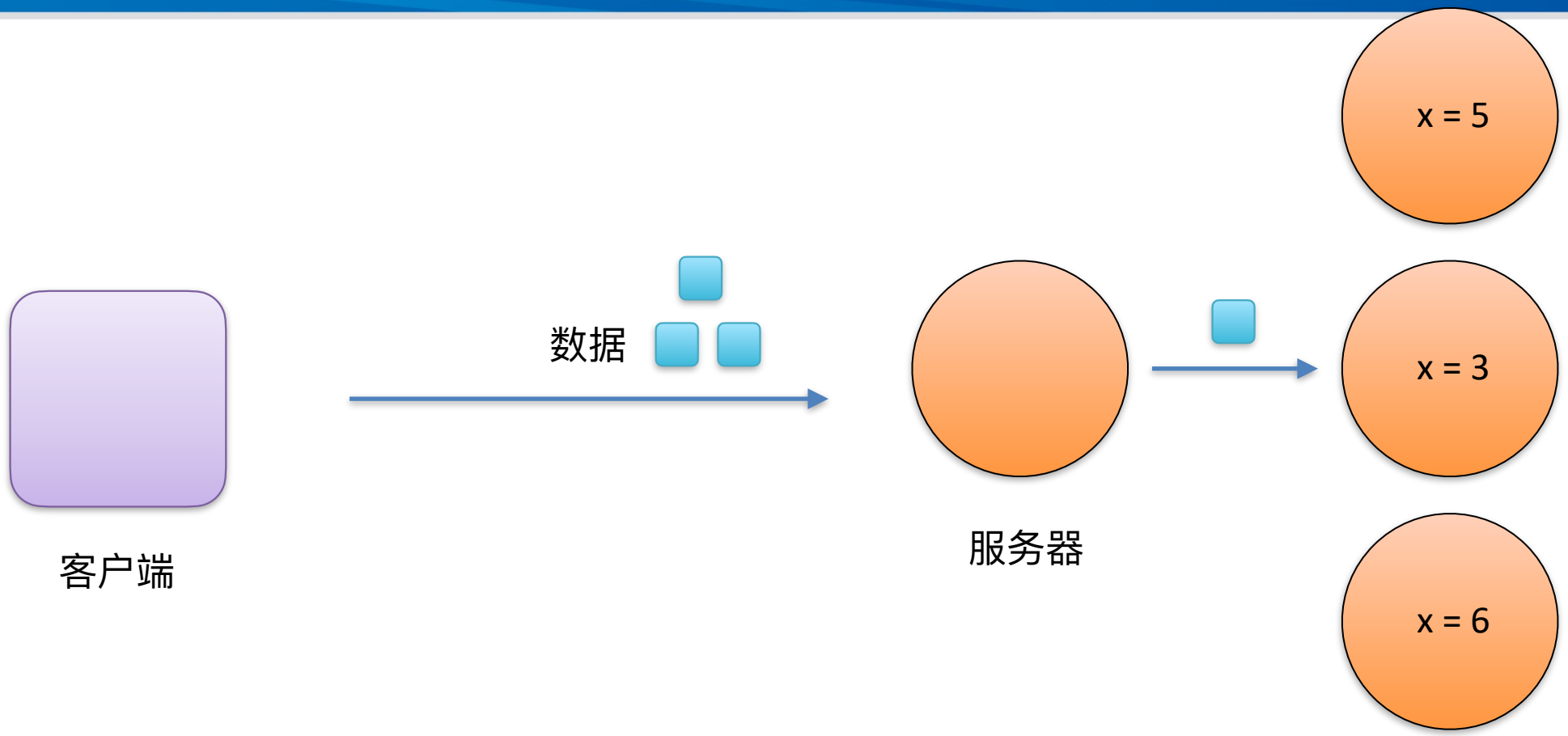
一致性问题

数据备份问题



一致性问题

数据备份问题



保持一致性

Raft

Raft

- 共识协议
- 实现了强一致性
- 强领导者协议
- 分布式存储系统
- 假设有2N+1个节点的情况下，至多在N个节点宕机的情况下，对外正常服务，保持高可用性

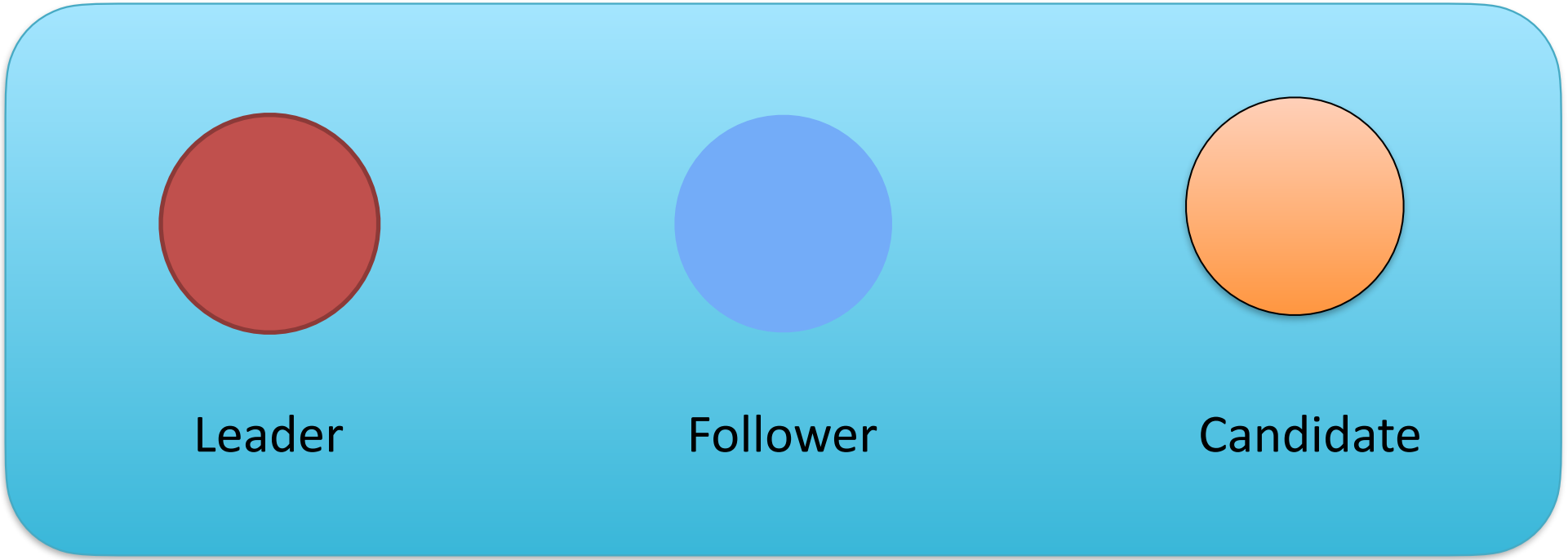
Raft

- 线性一致性 OceanBase Spanner
- 顺序一致性 Zookeeper
- 因果一致性 CorkroachDB
- 最终一致性 DNS -> 写后读一致性 单调读一致性

Raft架构

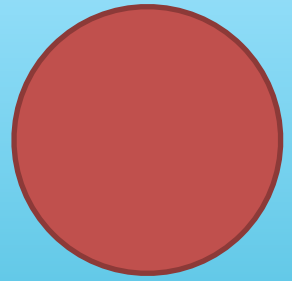
领导者选举

角色



领导者选举

角色



Leader

1. 负责接收命令
2. 将命令写入日志
3. 发送日志到其他主机中
4. 日志提交 返回结果到客户端
5. 空闲中发送心跳

领导者选举

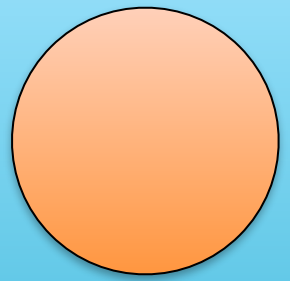
角色

Follower

1. 负责接收日志
2. 定时收到心跳
3. 未收到心跳，转变为Candidate

领导者选举

角色



Candidate

1. 负责进行选举
2. 向每台主机发送投票请求
3. 收到大多数主机的投票，变成Leader
4. 收到Leader心跳之后，变化为Follower

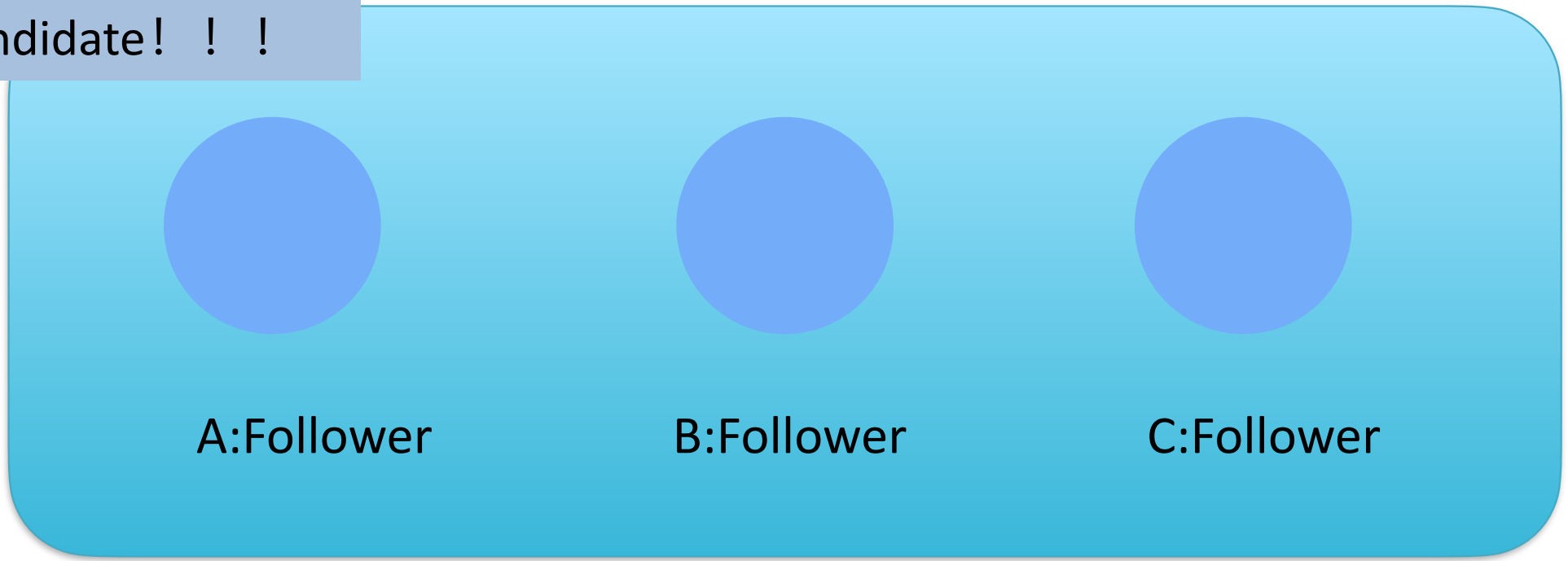
领导者选举

选举过程

领导者选举

选举过程

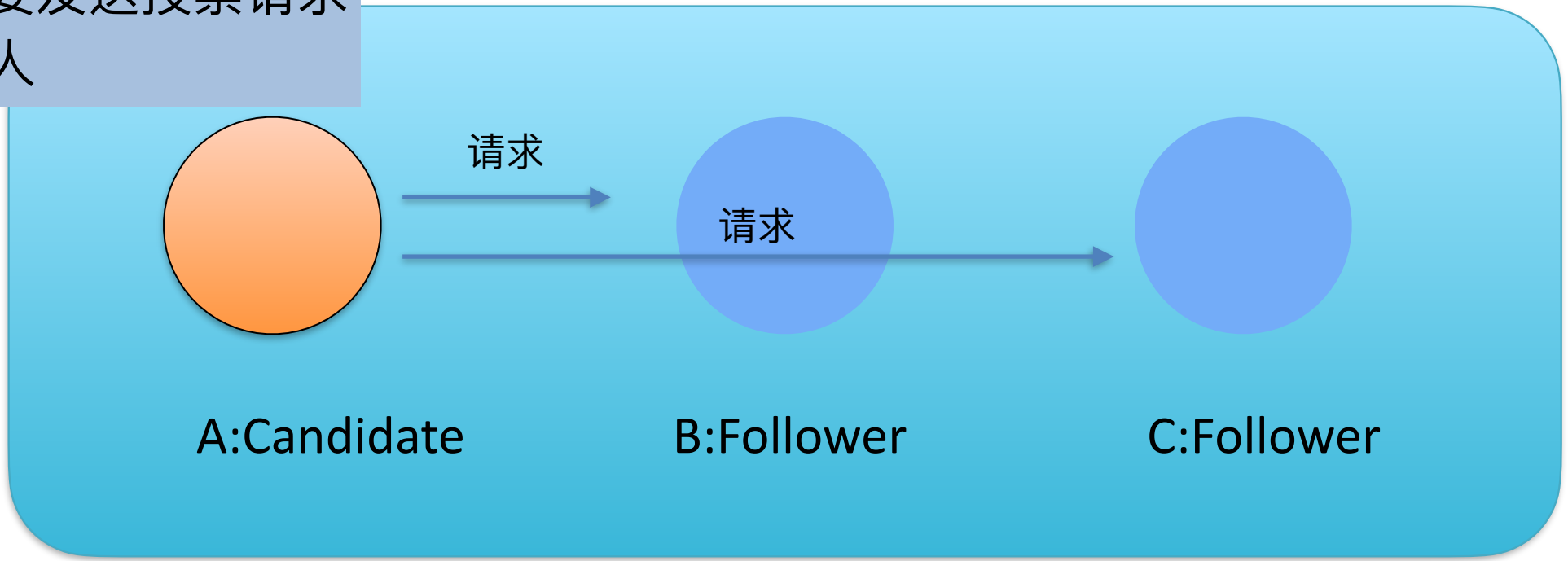
我没有收到心跳，我要变成Candidate！！！



领导者选举

我给自己先投票
之后我要发送投票请求
给其他人

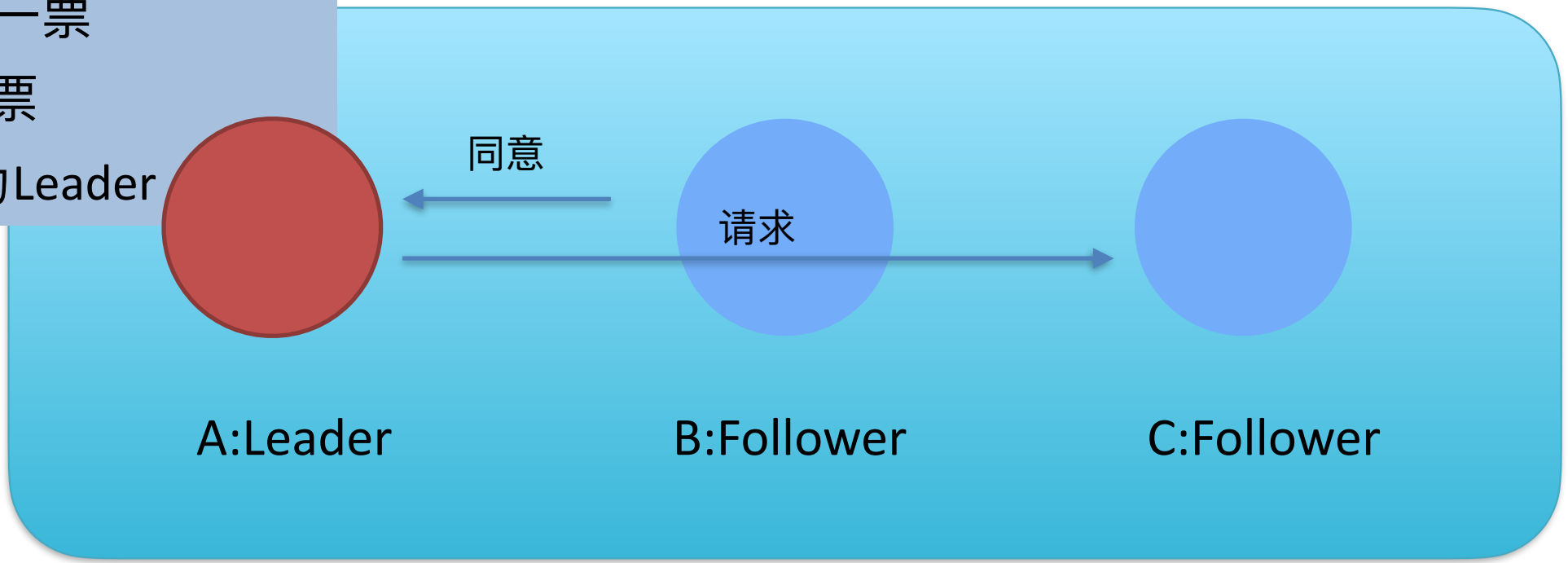
选举过程



领导者选举

选举过程

我自己有一票
B投给我一票
我收到2票
我要成为Leader



领导者选举

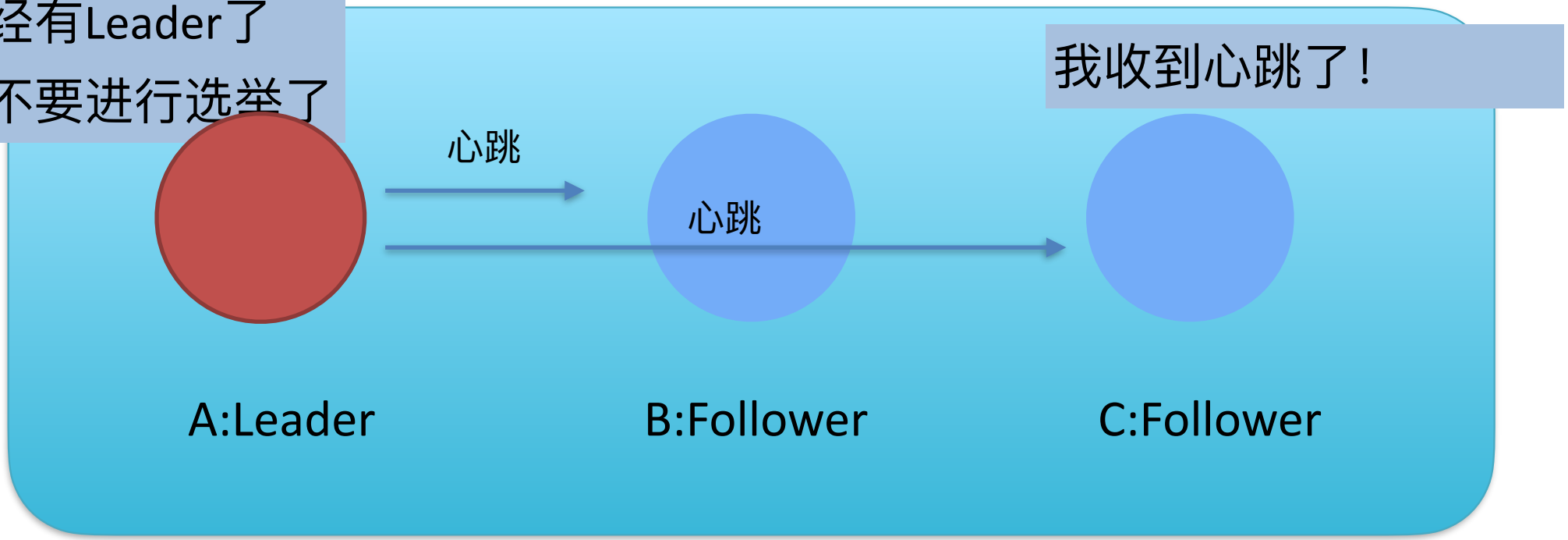
我是Leader了

我要发送我的心跳

证明已经有Leader了

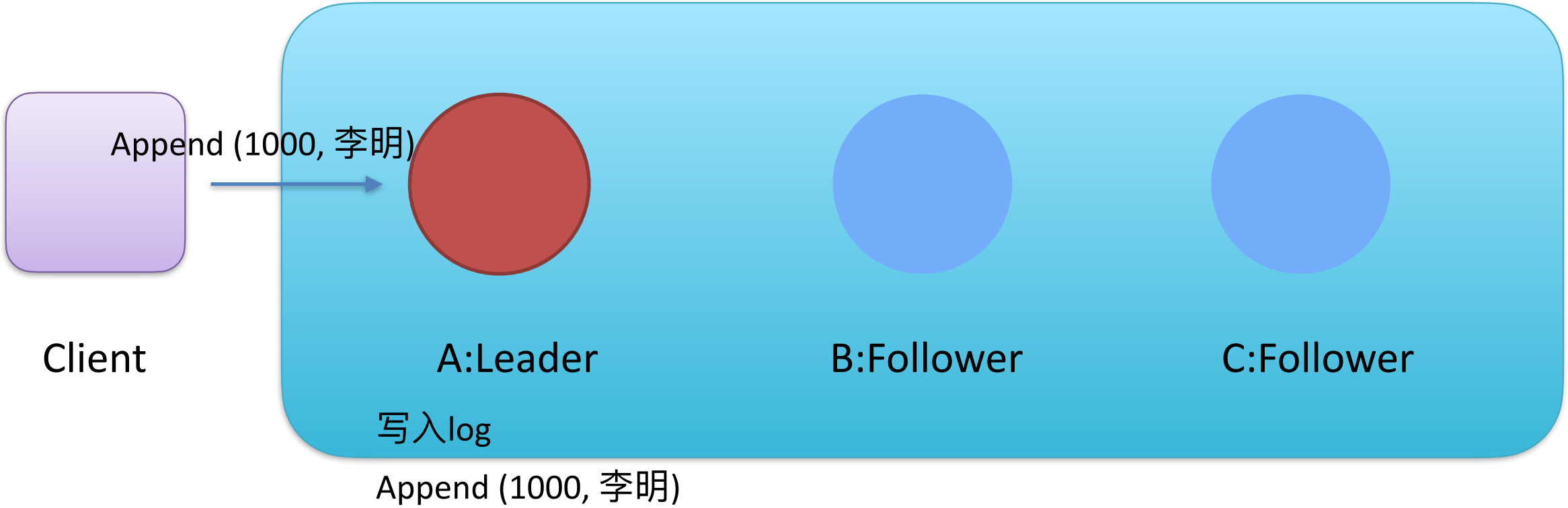
其他人不要进行选举了

选举过程



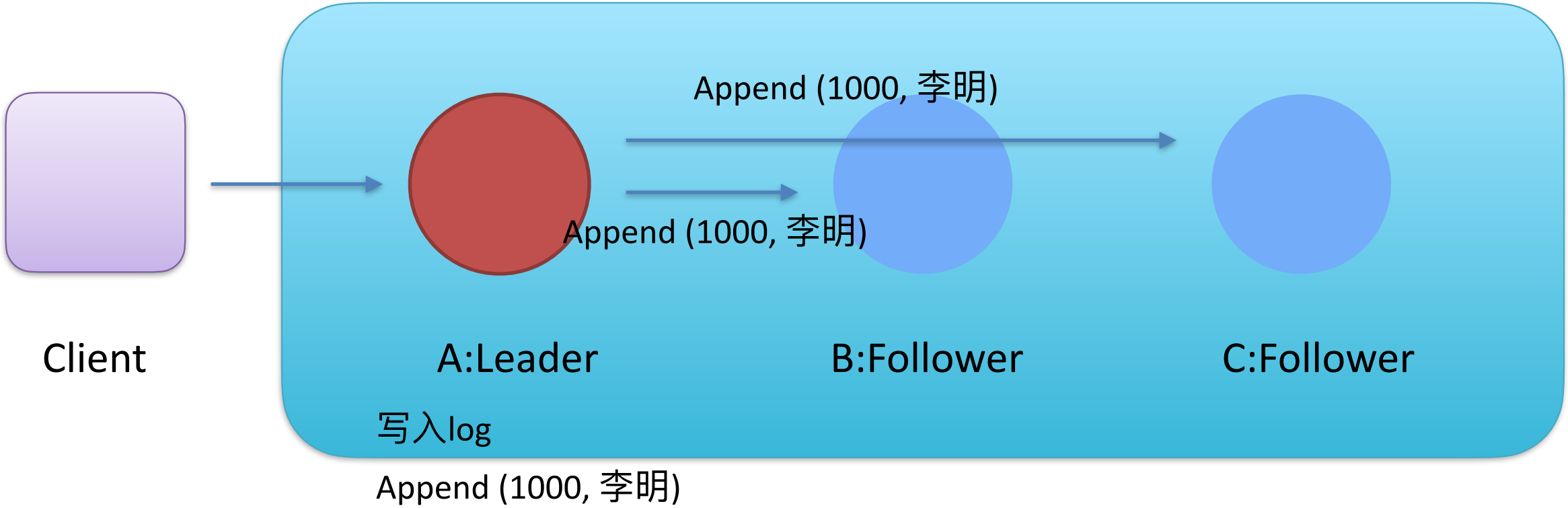
日志复制

日志复制阶段

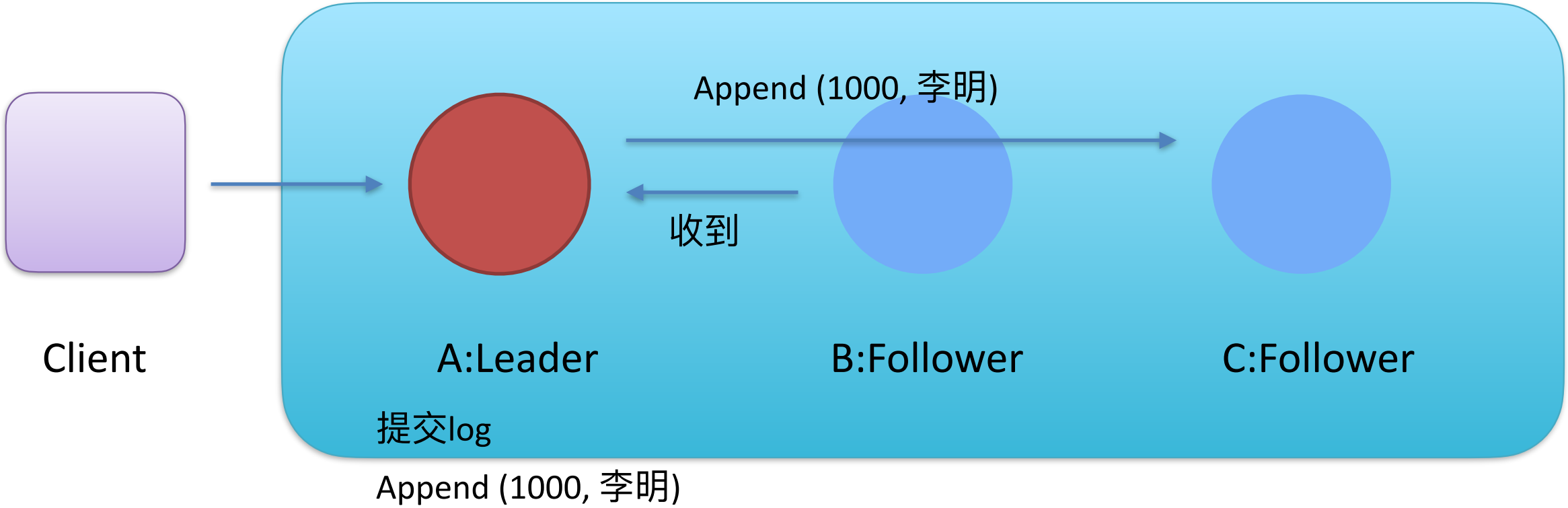

Append (1000, 李明)
Client
A:Leader
B:Follower
C:Follower
写入log
Append (1000, 李明)

日志复制

日志复制阶段

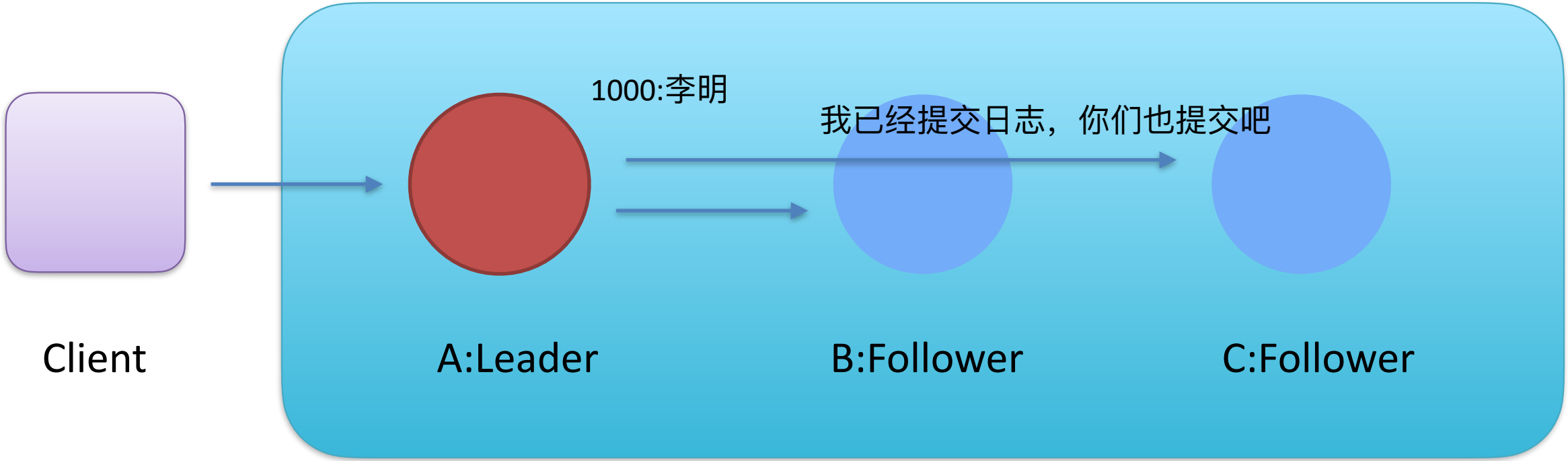

Append (1000, 李明)
Append (1000, 李明)
Client
A:Leader
B:Follower
C:Follower
写入log
Append (1000, 李明)

日志复制阶段

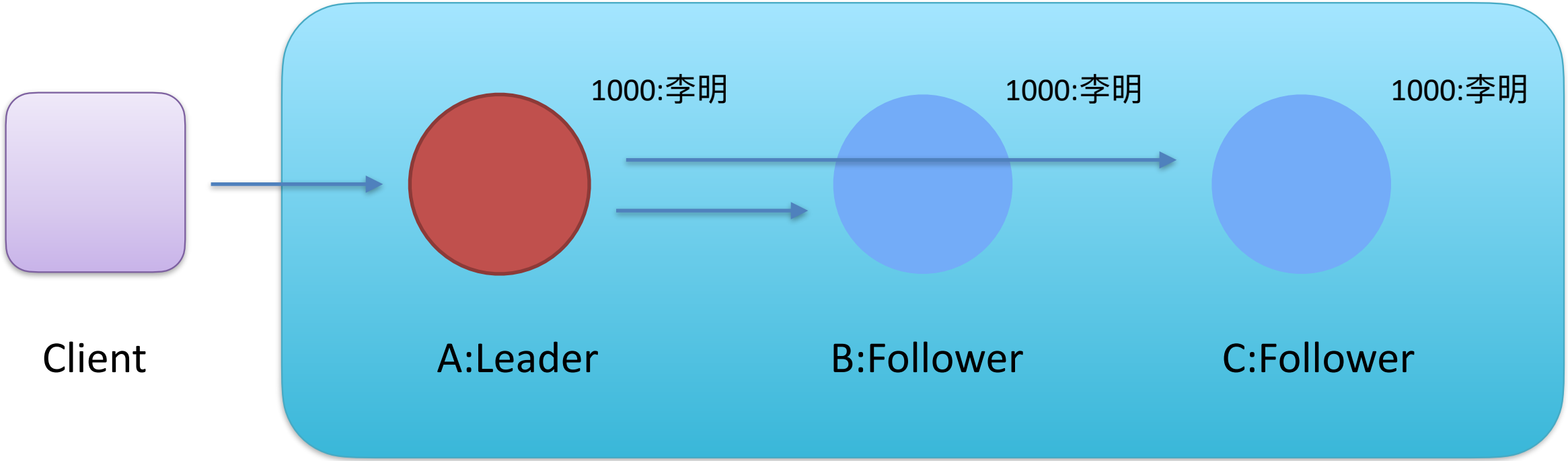

Append (1000, 李明)
收到
Client
A:Leader
B:Follower
C:Follower
提交log
Append (1000, 李明)

日志复制

日志复制阶段

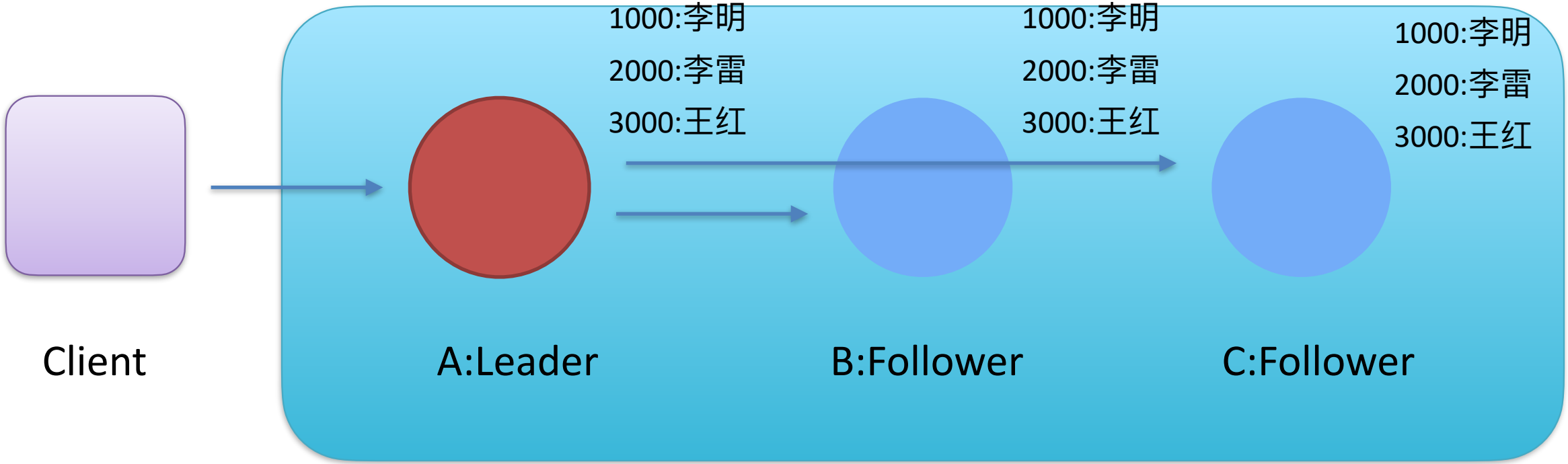

1000:李明
我已经提交日志，你们也提交吧
Client
A:Leader
B:Follower
C:Follower

日志复制阶段

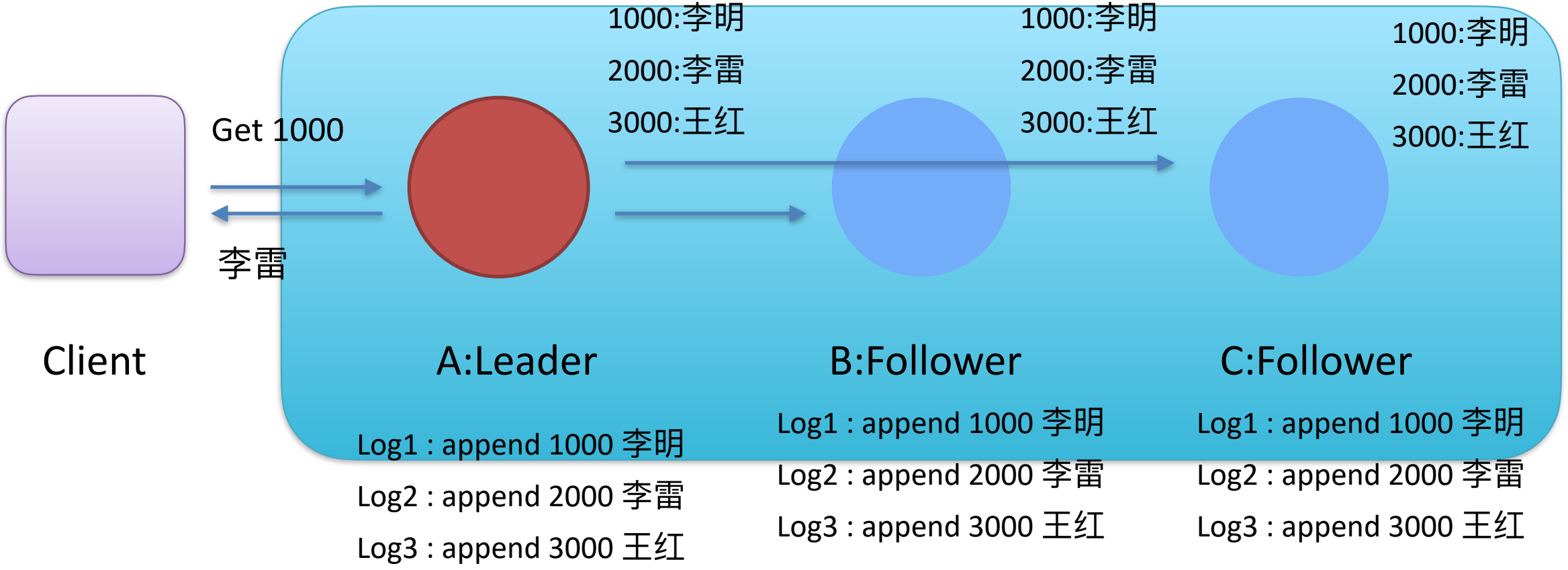


日志复制

日志复制阶段

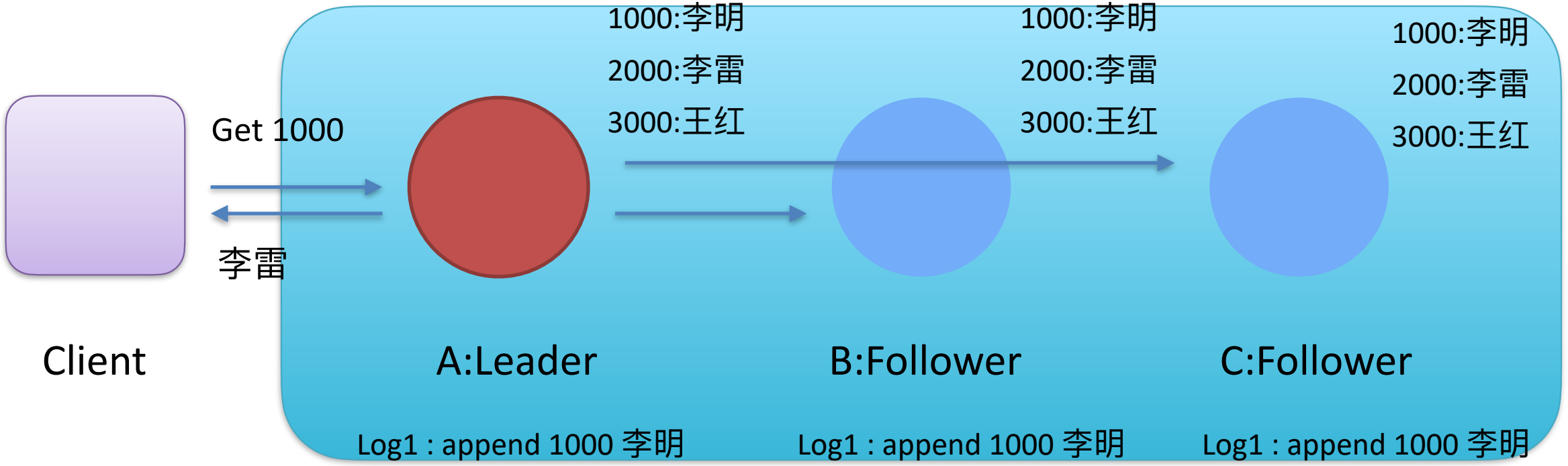

1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红
Client
A:Leader
B:Follower
C:Follower

日志复制阶段


Get 1000
李雷
1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红
Client
A:Leader
B:Follower
C:Follower
Log1 : append 1000 李明
Log2 : append 2000 李雷
Log3 : append 3000 王红
Log1 : append 1000 李明
Log2 : append 2000 李雷
Log3 : append 3000 王红
Log1 : append 1000 李明
Log2 : append 2000 李雷
Log3 : append 3000 王红

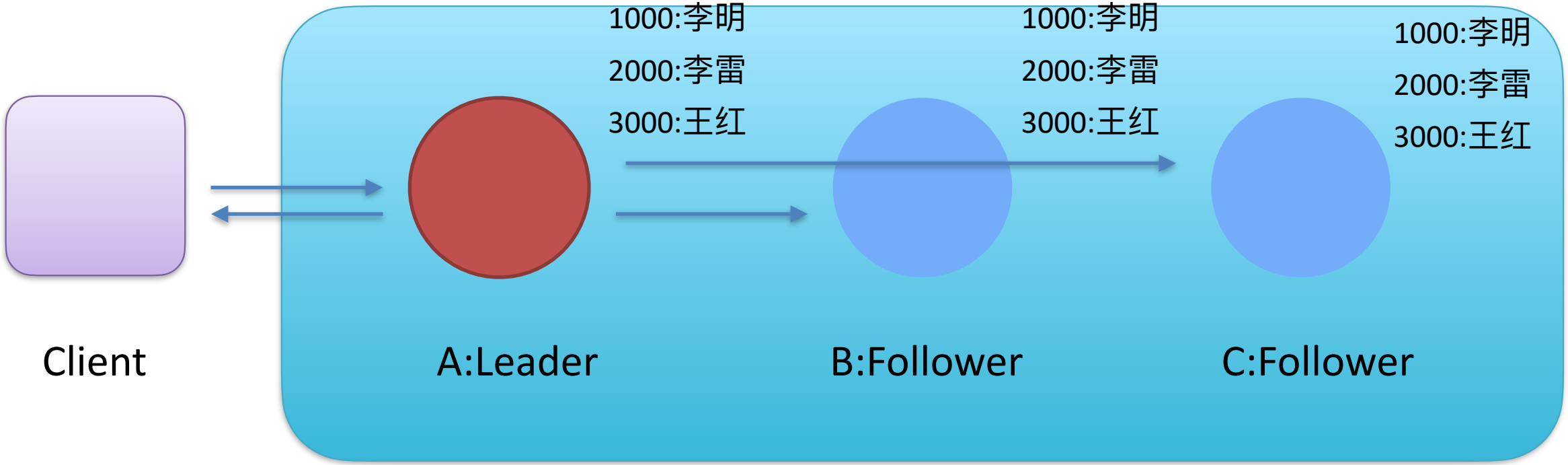
日志复制

日志复制阶段


Get 1000
李雷
1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红
Client
A:Leader
B:Follower
C:Follower
Log1 : append 1000 李明
Log2 : append 2000 李雷
Log3 : append 3000 王红
Log4 : get 1000
Log1 : append 1000 李明
Log2 : append 2000 李雷
Log3 : append 3000 王红
Log4 : get 1000
Log1 : append 1000 李明
Log2 : append 2000 李雷
Log3 : append 3000 王红
Log4 : get 1000

日志复制

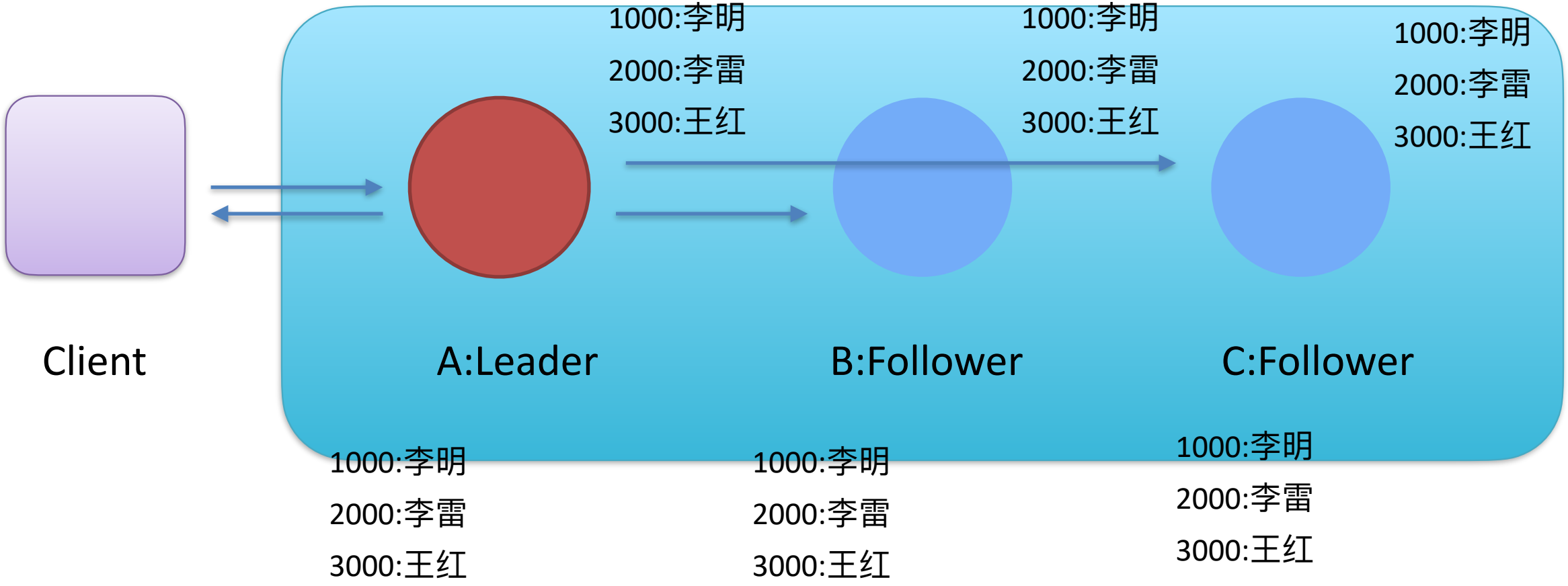
日志复制阶段



Log太多了 导致我的Log集合存不下！！！

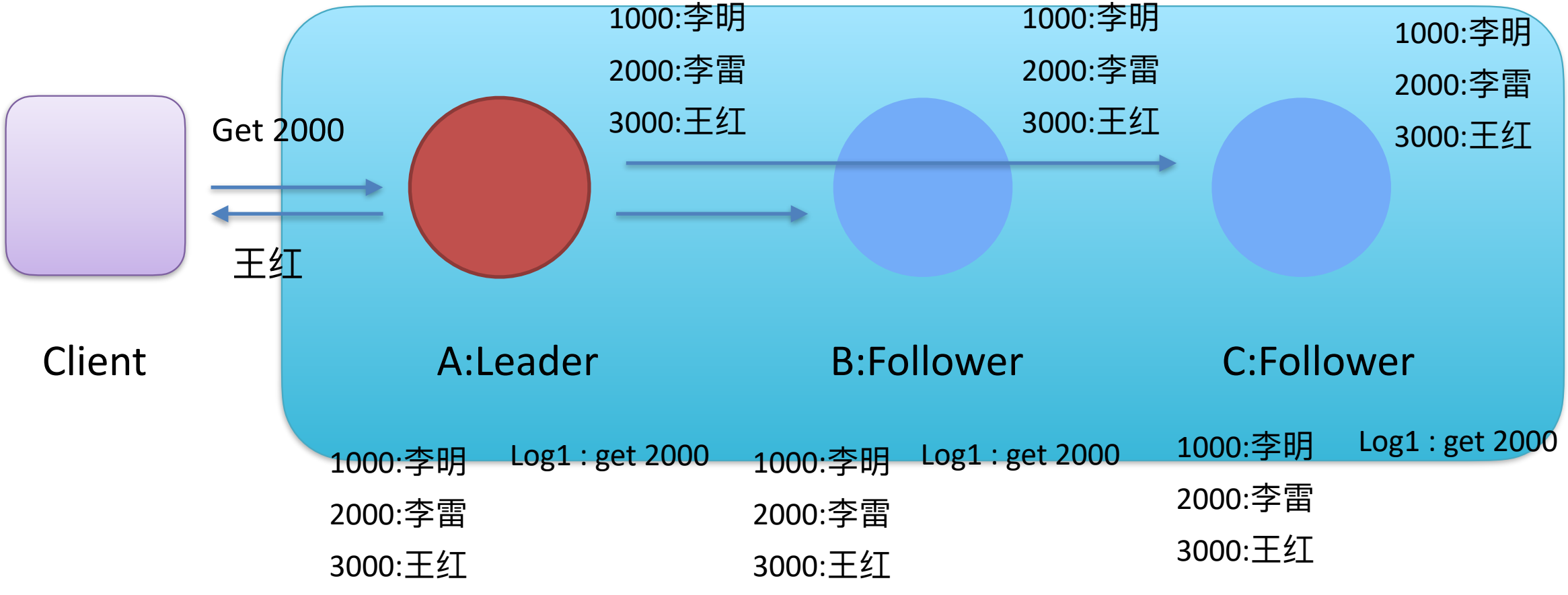
日志复制

日志复制阶段


1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红
Client
A:Leader
B:Follower
C:Follower
1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红

日志复制

日志复制阶段

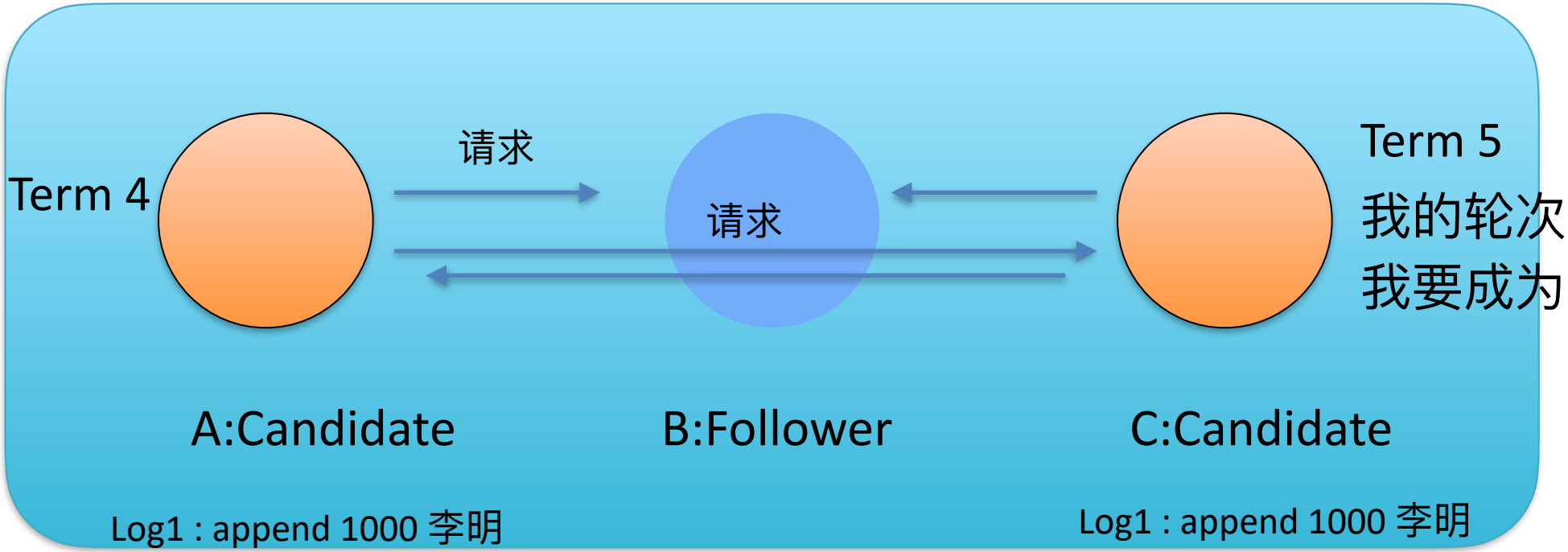

1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红
1000:李明
2000:李雷
3000:王红
Get 2000
王红
Client
A:Leader
B:Follower
C:Follower
1000:李明
2000:李雷
3000:王红
Log1 : get 2000
1000:李明
2000:李雷
3000:王红
Log1 : get 2000
1000:李明
2000:李雷
3000:王红
Log1 : get 2000

概念Term

- Term在Raft中是一个时间概念
- 每个term最多只有一个leader
- 如果leader宕机，term增加，其他的follower会重新进行选举
- 在Raft中，term越大，代表数据越新
- Term小的主机会听从term大的主机

故障恢复

case 1 term不相等



Log1 : append 1000 李明

Log2 : append 2000 李雷

Log3 : append 3000 王红

Log4 : get 1000

Log1 : append 1000 李明

Log2 : append 2000 李雷

Log3 : append 3000 王红

Log4 : get 1000

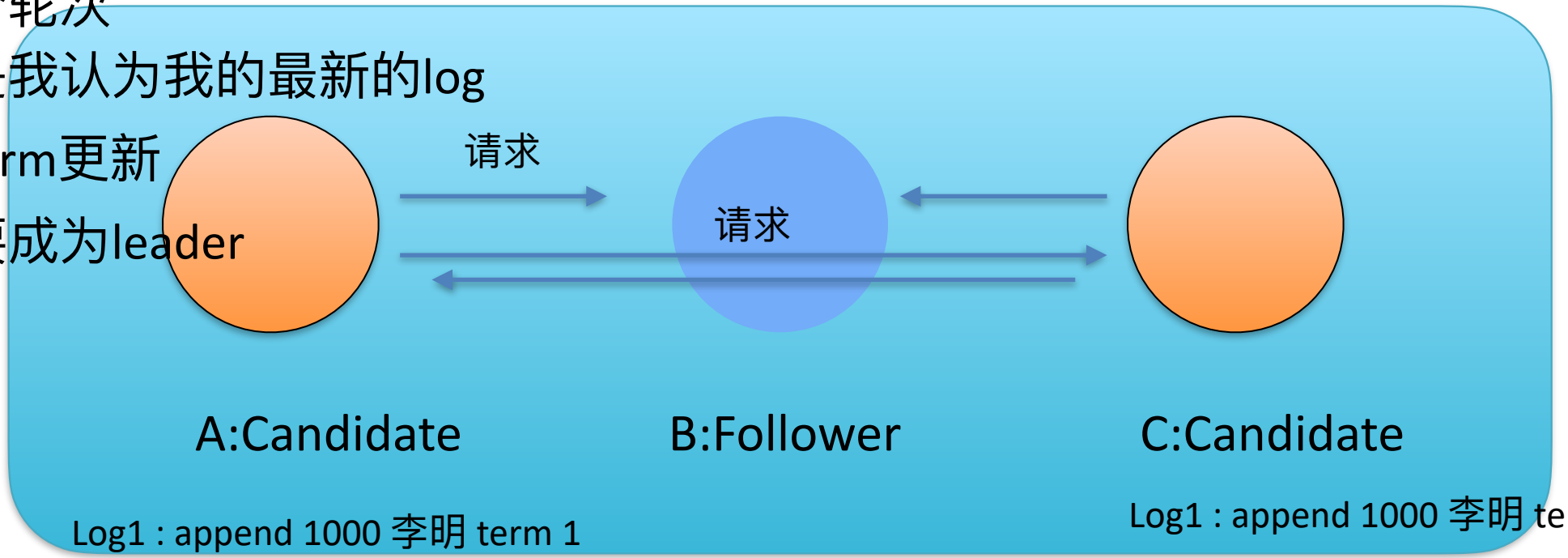
故障恢复

case 2 term相等

虽然咱们两个处于一个轮次

但是我认为我的最新的log的term更新

我要成为leader



Log1 : append 1000 李明 term 1

Log2 : append 2000 李雷 term 2

Log3 : append 3000 王红 term 3

Log4 : get 1000 term 6

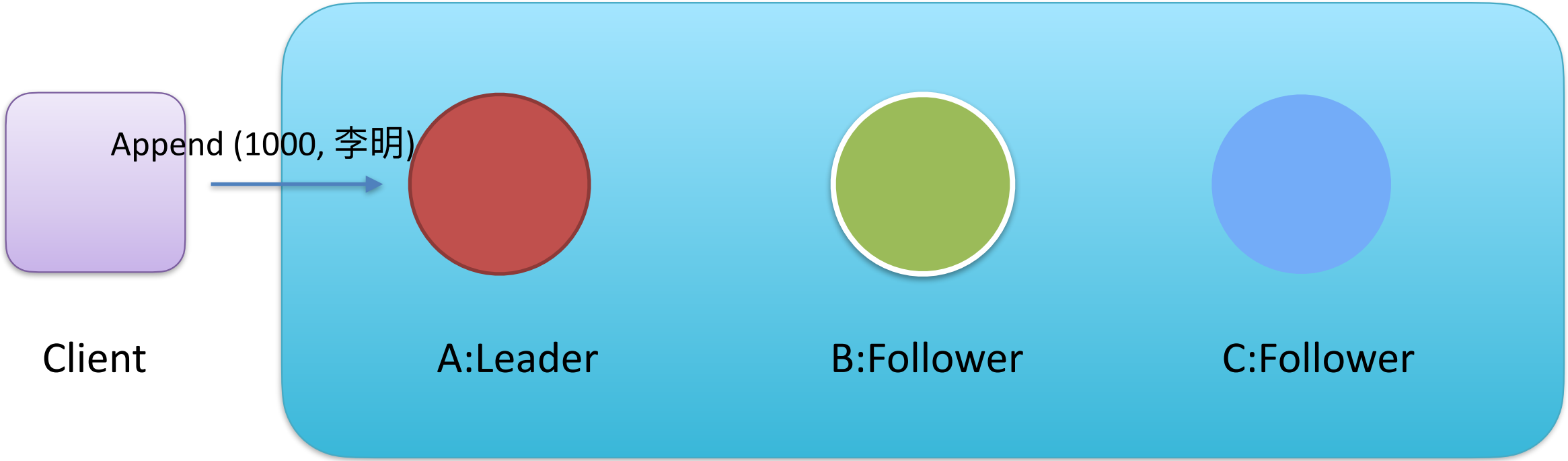
Log1 : append 1000 李明 term 1

Log2 : append 2000 李雷 term 2

Log3 : append 3000 王红 term 3

故障恢复

case 3 日志不统一

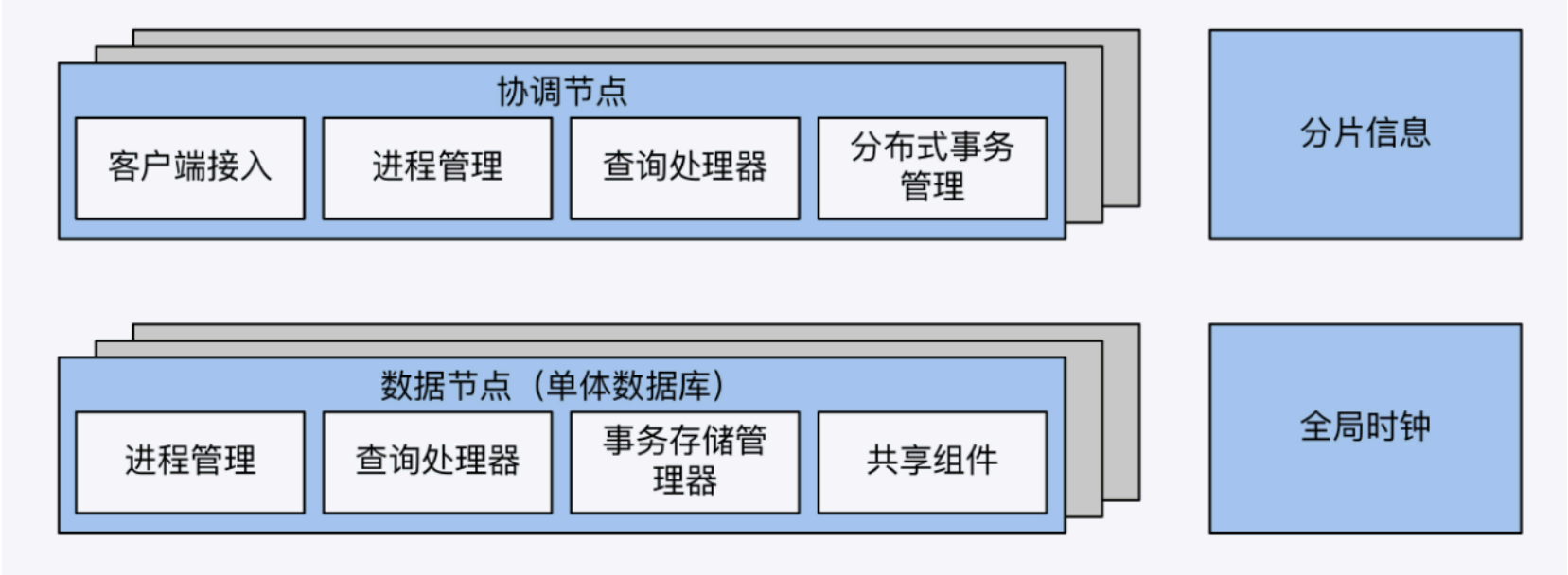


case 3 日志不统一

	index 11	index 12	index 13	Index 14
A:Leader Term5	Term 4	Term 4	Term 5	Term 5
B	Term 4	Term 4	Term 5	
C	Term 4	Term 4	Term5	
D	Term 4	Term 4	Term 4	

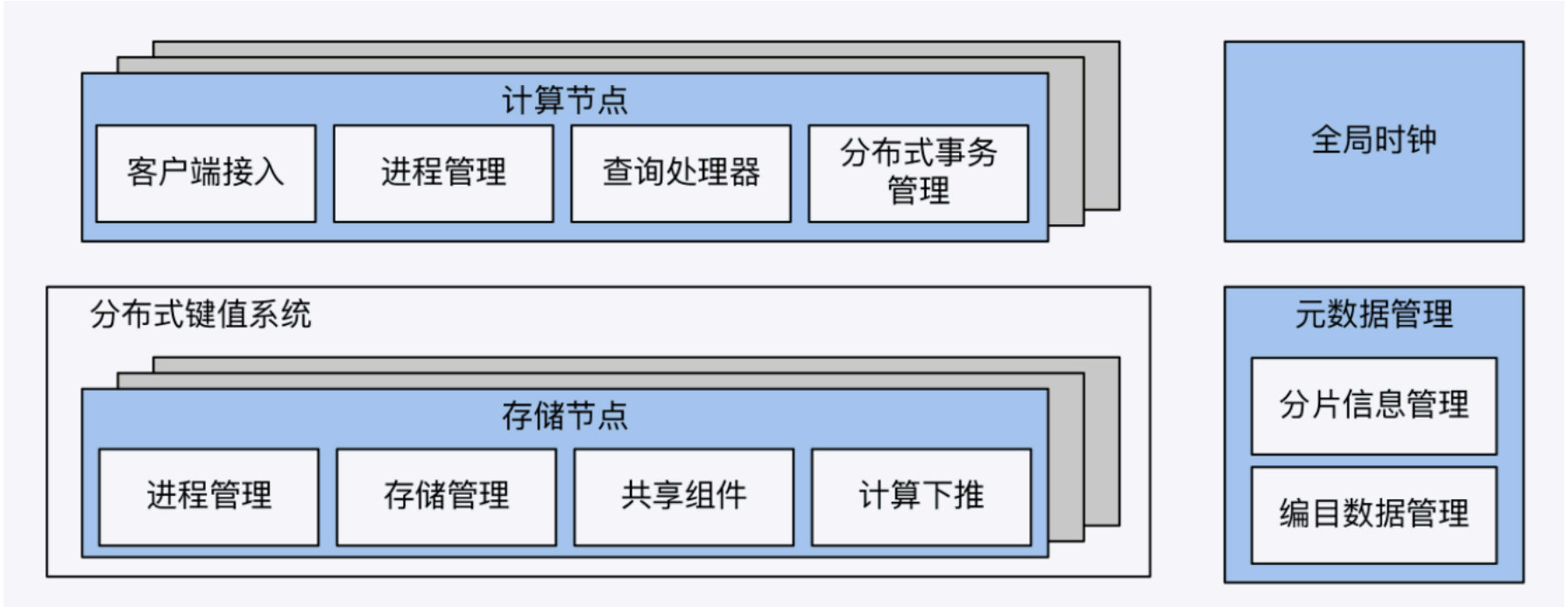
深入理解Raft

– 传统SQL（proxy）与NewSQL



深入理解Raft

- Multi-raft-group
- 传统SQL（proxy）与NewSQL



深入理解Raft

– 如何理解实现了强一致性

– Consul针对raft实现了三个一致性模型

– Default(无法防止脑裂)
– Consistent(强一致性)
– Stale(可以读到过时数据)

Q & A