分布式一致性raft实现原理

- 峰云就她了
- xiaorui.cc

介绍

- 什么是一致性协议？
- raft有哪些特点？
- raft vs paxos？
- raft的构成组件及实现原理？
- 各种所谓奇葩的raft场景？
- 如何实现raft？

单节点环境

存在数据一致性问题？

多节点环境

node 1

node 2

node 3

那么如何保证数据的一致性？

角色

Follower

Candidate

Leader

KeyWorld

- 定时器
- Term 时间片
- Term ID
- N/2 + 1
- Heartbeats

KeyWorld

- 选举成Leader需提供TermID 和 LogIndex
- Leader 绝对不会删除自己的日志
- 客户端自己携带ID帮助raft保持幂等性
- 一条记录提交了，那么它之前的记录一定都是commited.

KeyWorld

- 节点之间的Term和索引一致, 我们就认为数据是一致的.
- 在一个Term里只会有一个Leader
- 每个Follower只能选一个Leader

KeyWorld

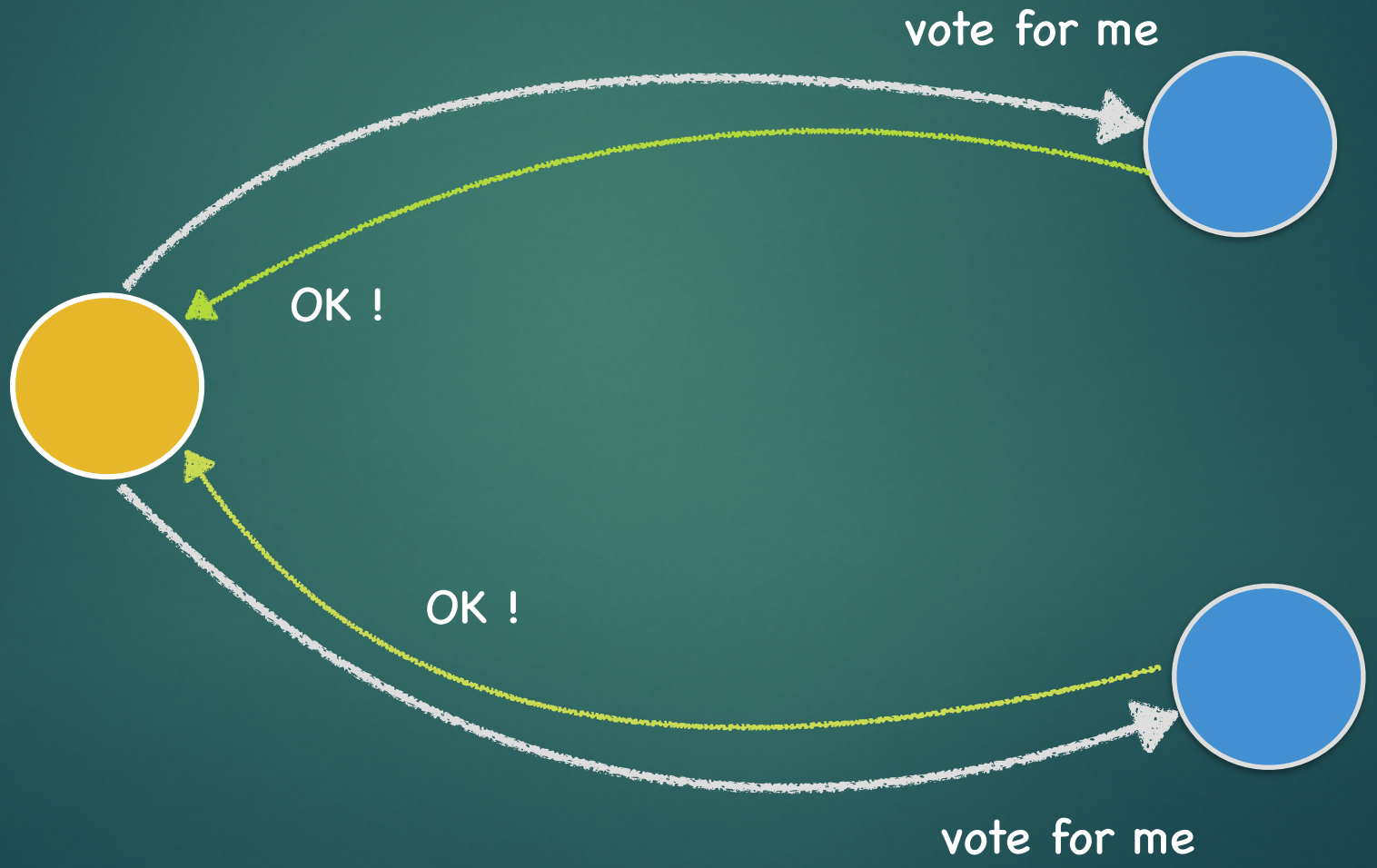
- currentTerm
  - 服务器最后一次知道的任期号（初始化为 0，持续递增）
- voteFor
  - 在当前获得选票的候选人的 Id
- log[]
  - 日志条目集(状态机指令及TermId)
- commitIndex
  - 已知最大的索引值
- nextIndex[]
  - 每个follower的下一个索引值

Vote RPC

Term	候选人的任期号
candidateid	ID
lastLogIndex	候选人的最后日志的索引值
lastLogTerm	候选人最后日志的任期号

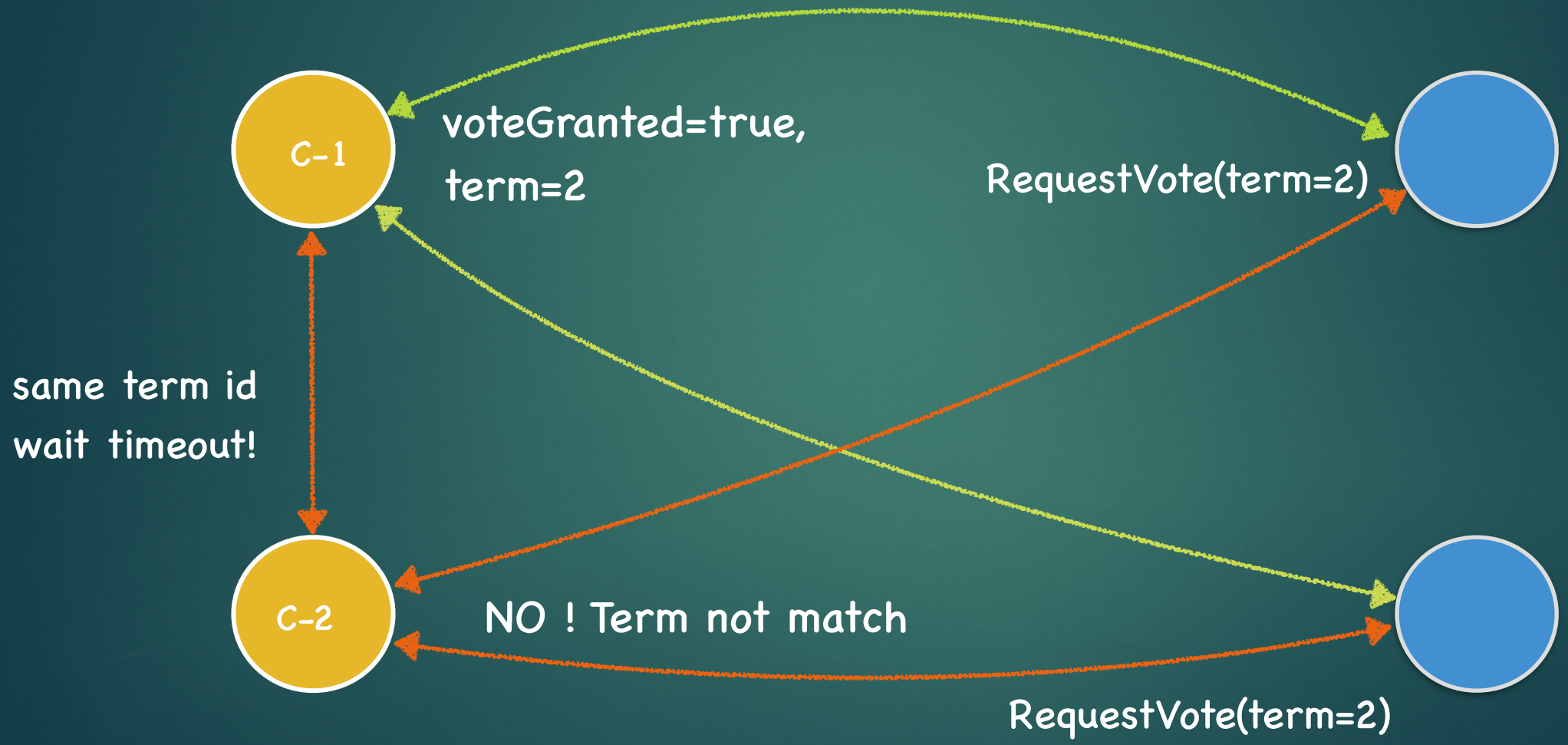
Term	当前的任期号, 用于领导人去更新自己
voteGranted	True or False

most simple election

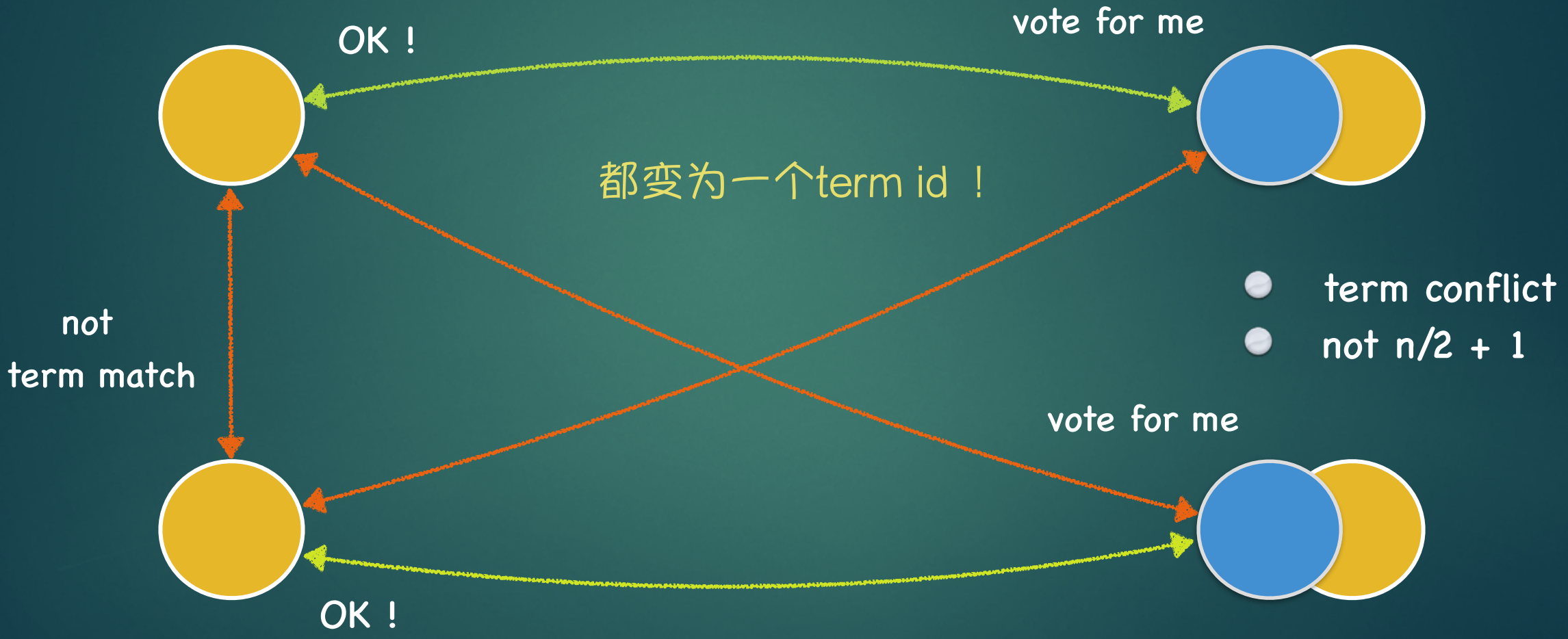


simple election

simple election



hard election -1



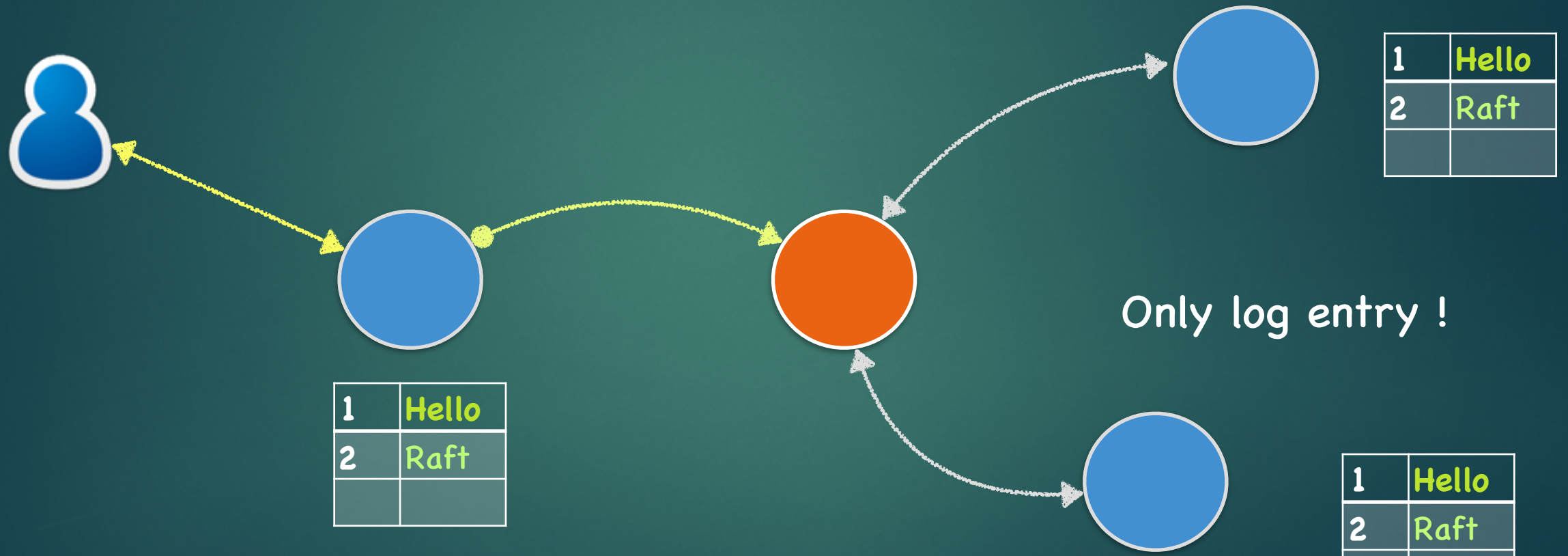
summery election

- 过程
  - 定时器触发, followers把current_term_id + 1
  - 改变成candidate状态
  - 发送RequestVoteRPC请求
- 结果
  - 成功选举
  - 别人被选
  - 重新选

Client

- Works with leader
- Leader return to response when it commits an entry !
- Assign uniquqeID to every command , Leader store latest ID with response.

client process



Log Replication

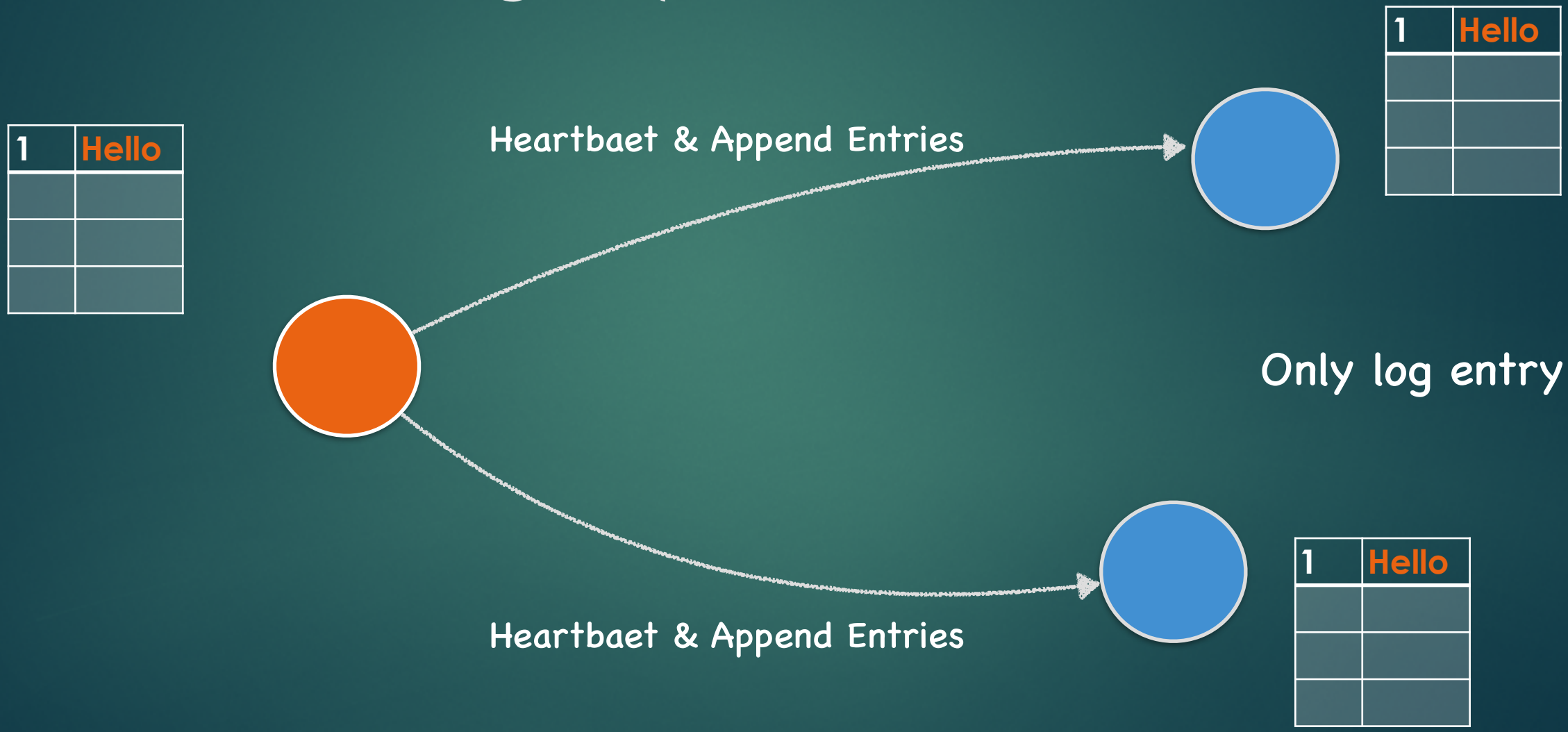
- 默认心跳为 50 ms
- 默认心跳超时为 300ms
- 每次心跳的时候做 Log entry \ commit
- 超过 n/2+1 就算成功

Log RPC

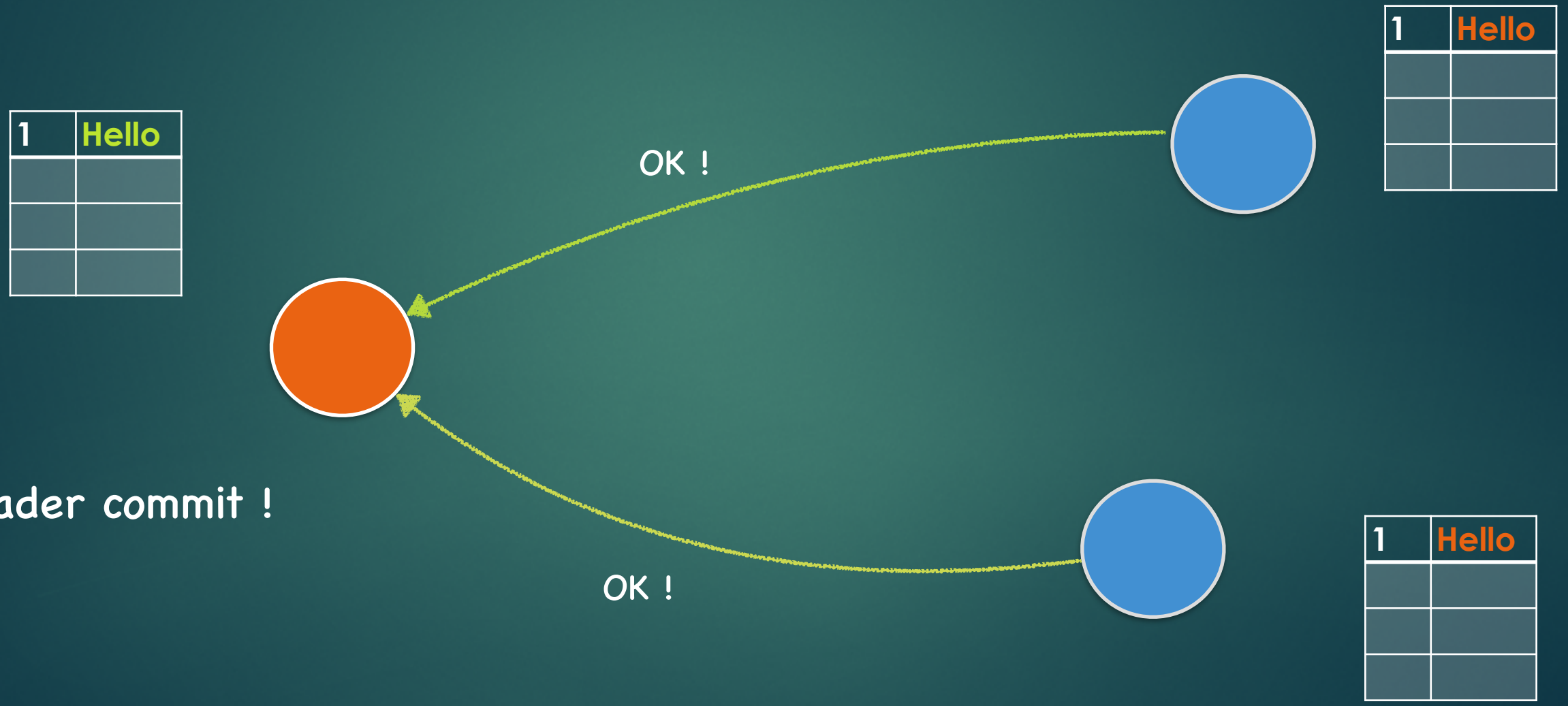
Term	领导人的任期号
LeaderID	领导人的 Id，以便于跟随者重定向请求
pervLogIndex	新的日志条目紧随之前的索引值
entries[]	需要存储当然日志条目（表示心跳时为空；一次性发送多个是为了提高效率）
LeaderCommit	领导人已经提交的日志的索引值

Term	当前的任期号，用于领导人去更新自己
success	跟随者包含了匹配上 prevLogIndex 和 prevLogTerm 的日志时为真

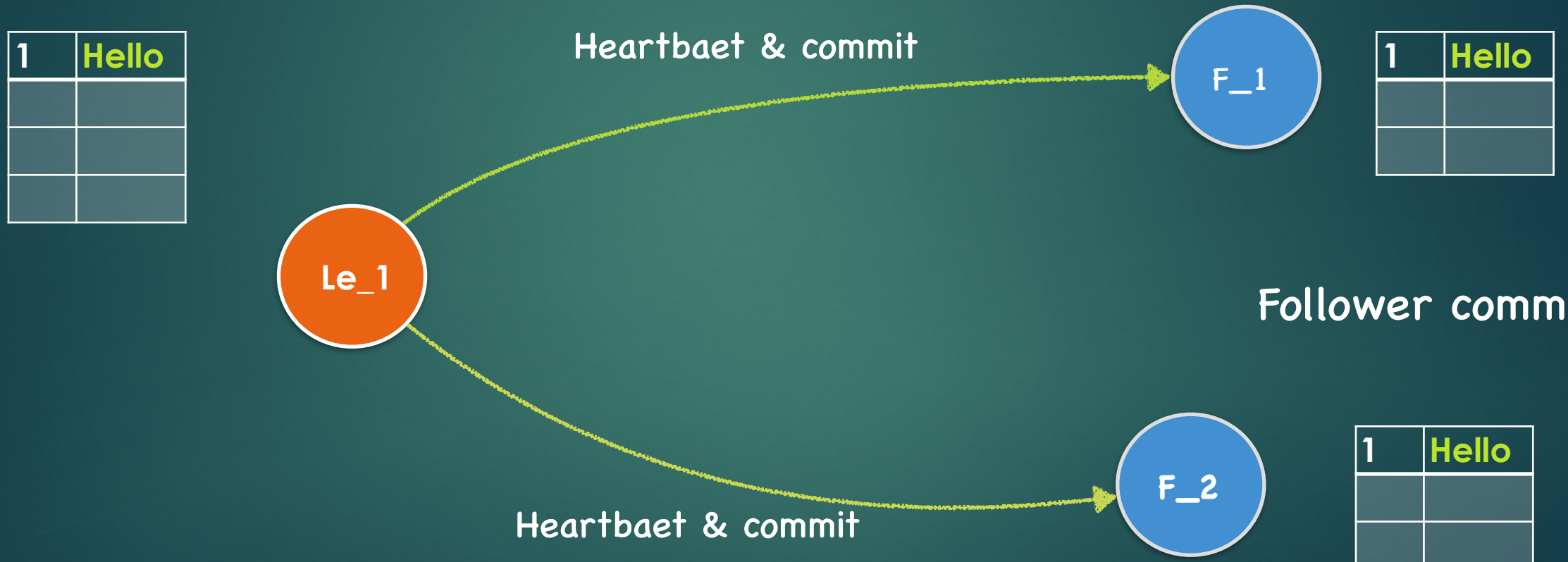
log replication - 1



log replication - 2

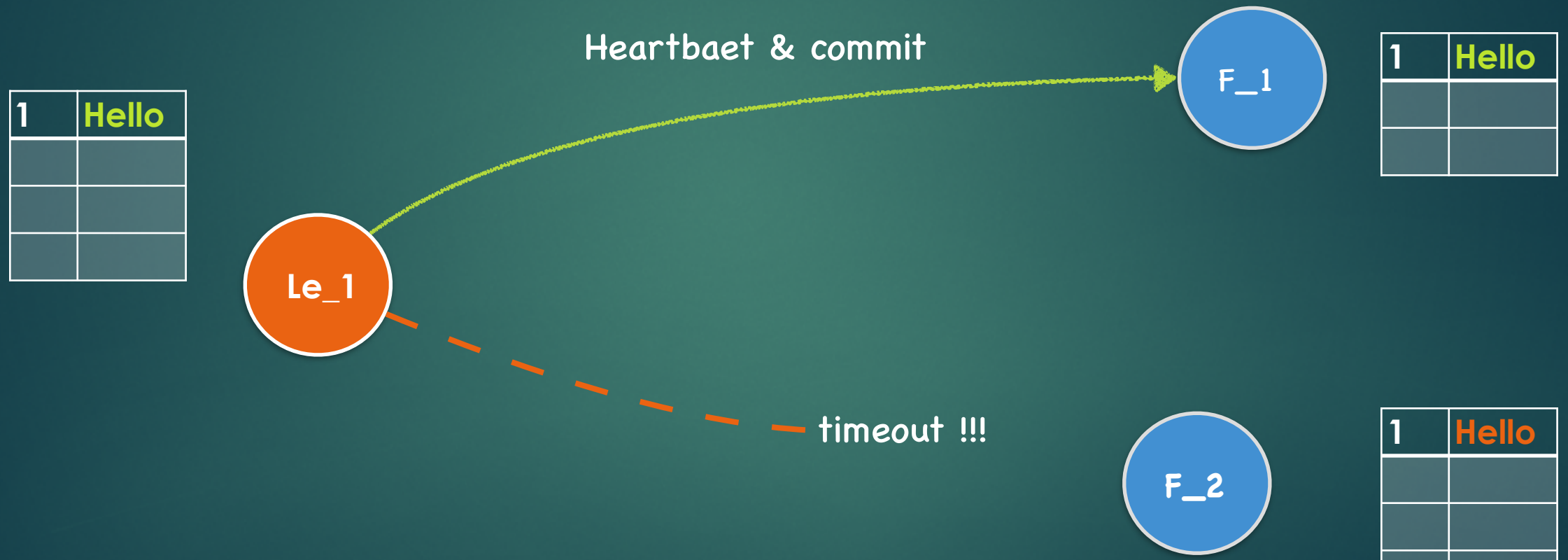


log replication - 3



常见疑难杂症

if a node reply timeout ?



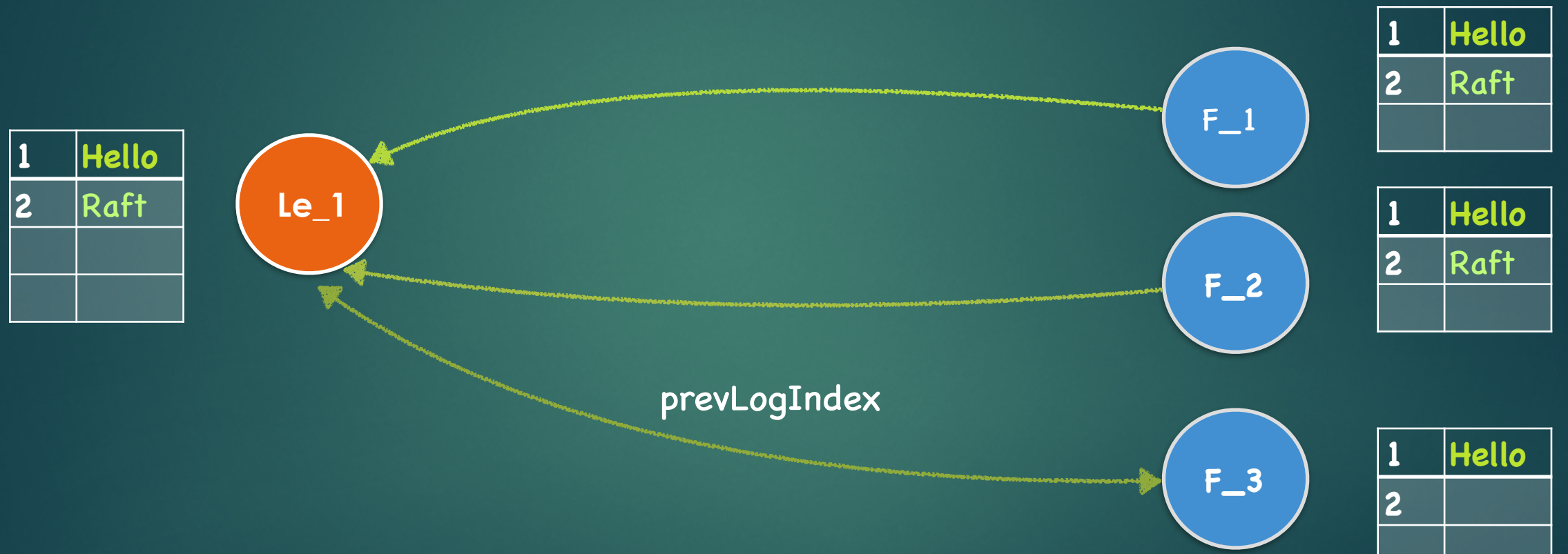
F_2 如何保持数据一致性 ？ Leader会重试！

Leader crash

Leader在本地commit后，发给follower commit 之前**crash**！

Hello 还在么？

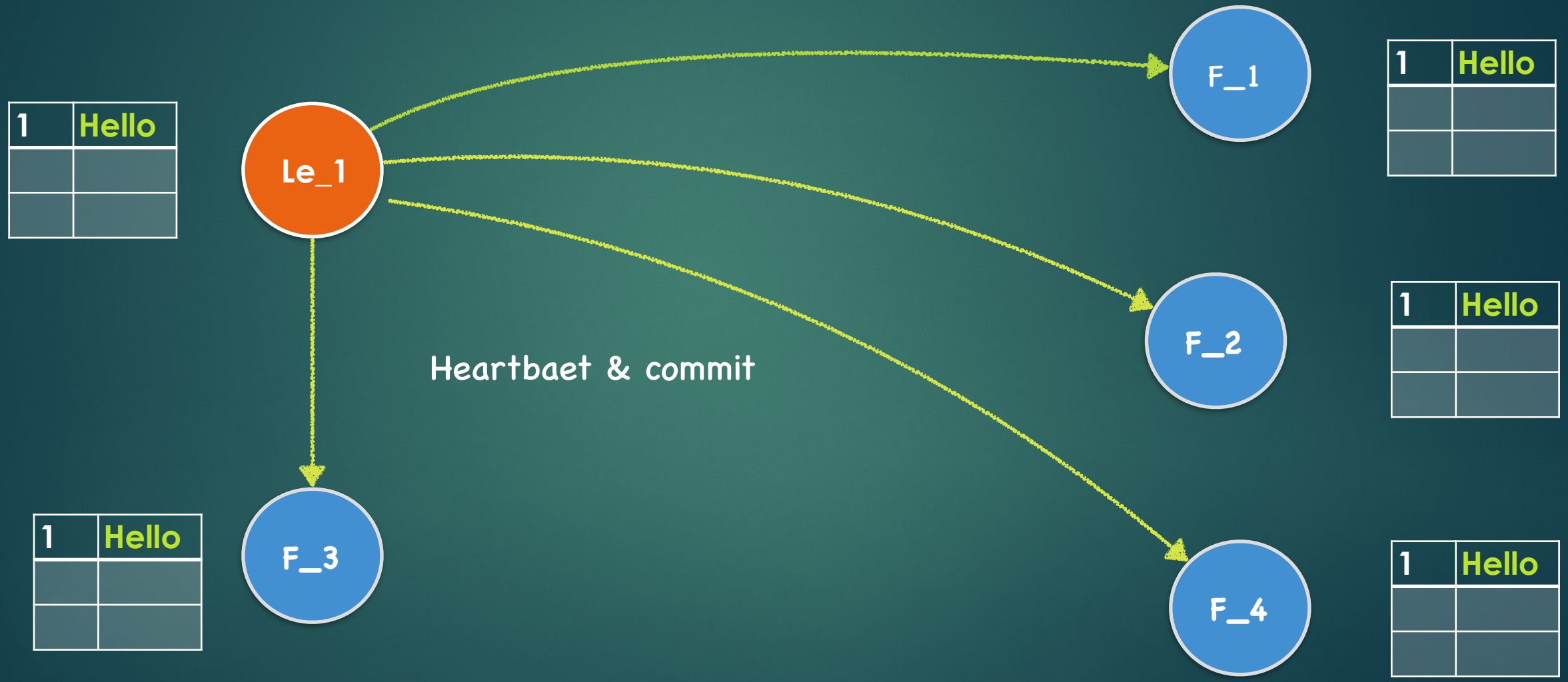
Follower crash



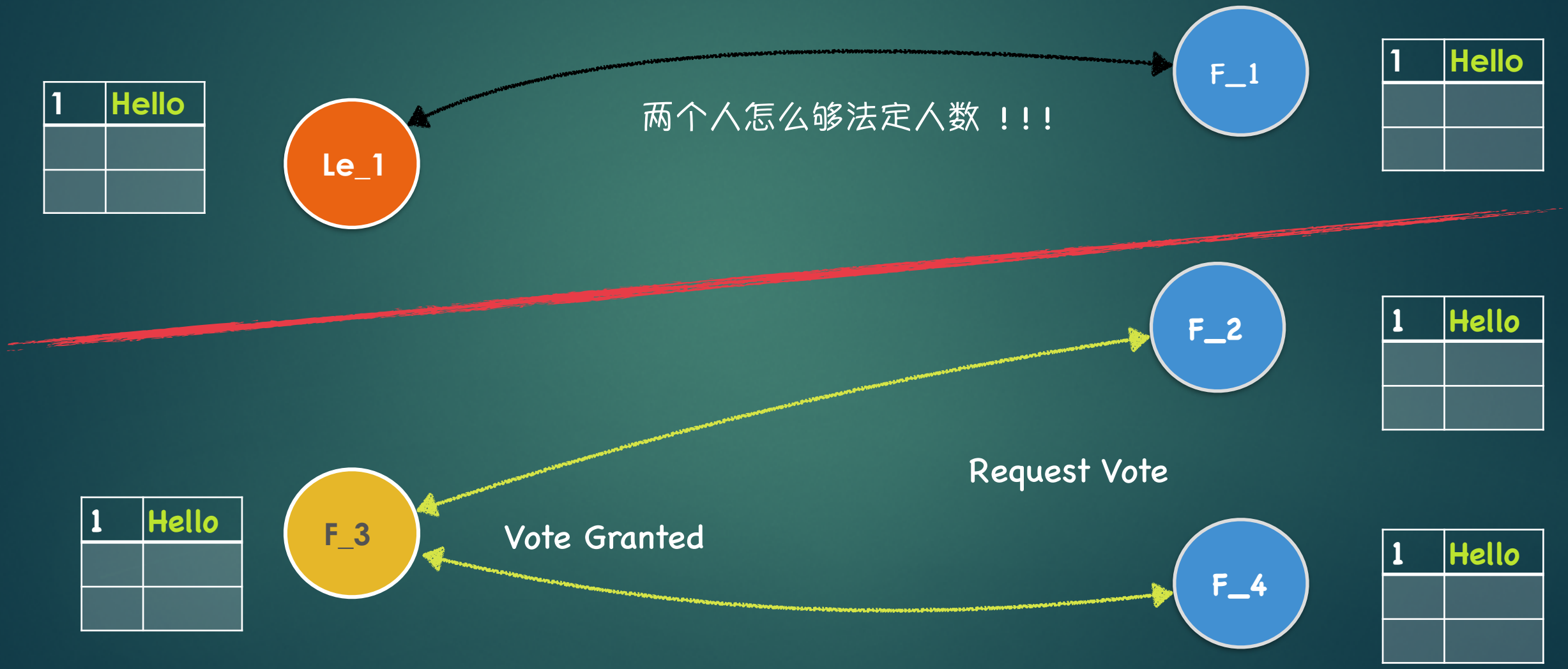
F_3 crash重新启动后如何平衡数据.

Network Partition

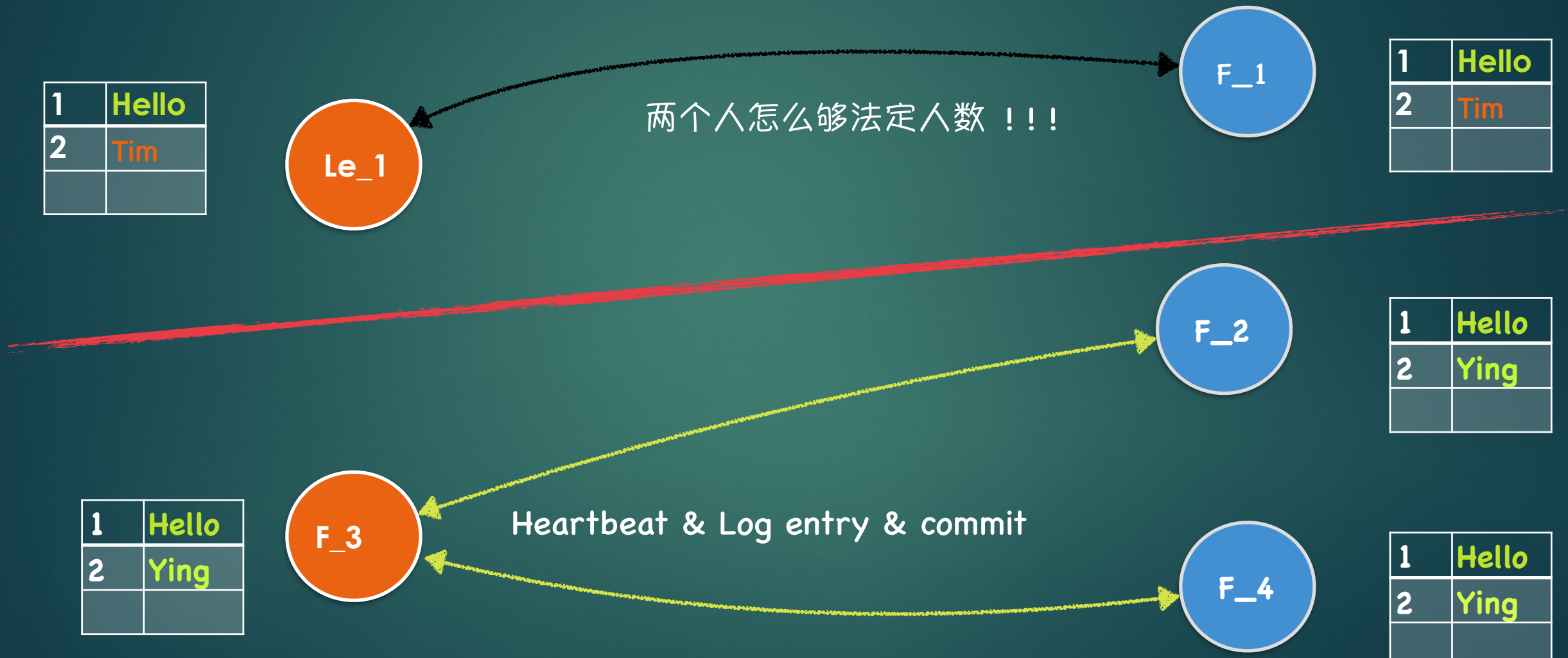
正常情况


1 Hello
Le_1
F_1
1 Hello
F_2
1 Hello
Heartbaet & commit
1 Hello
F_3
F_4
1 Hello

网络分区


1 Hello
Le_1
两个人怎么够法定人数 ！！！
F_1
1 Hello
F_2
1 Hello
Request Vote
1 Hello
F_3
Vote Granted
F_4
1 Hello

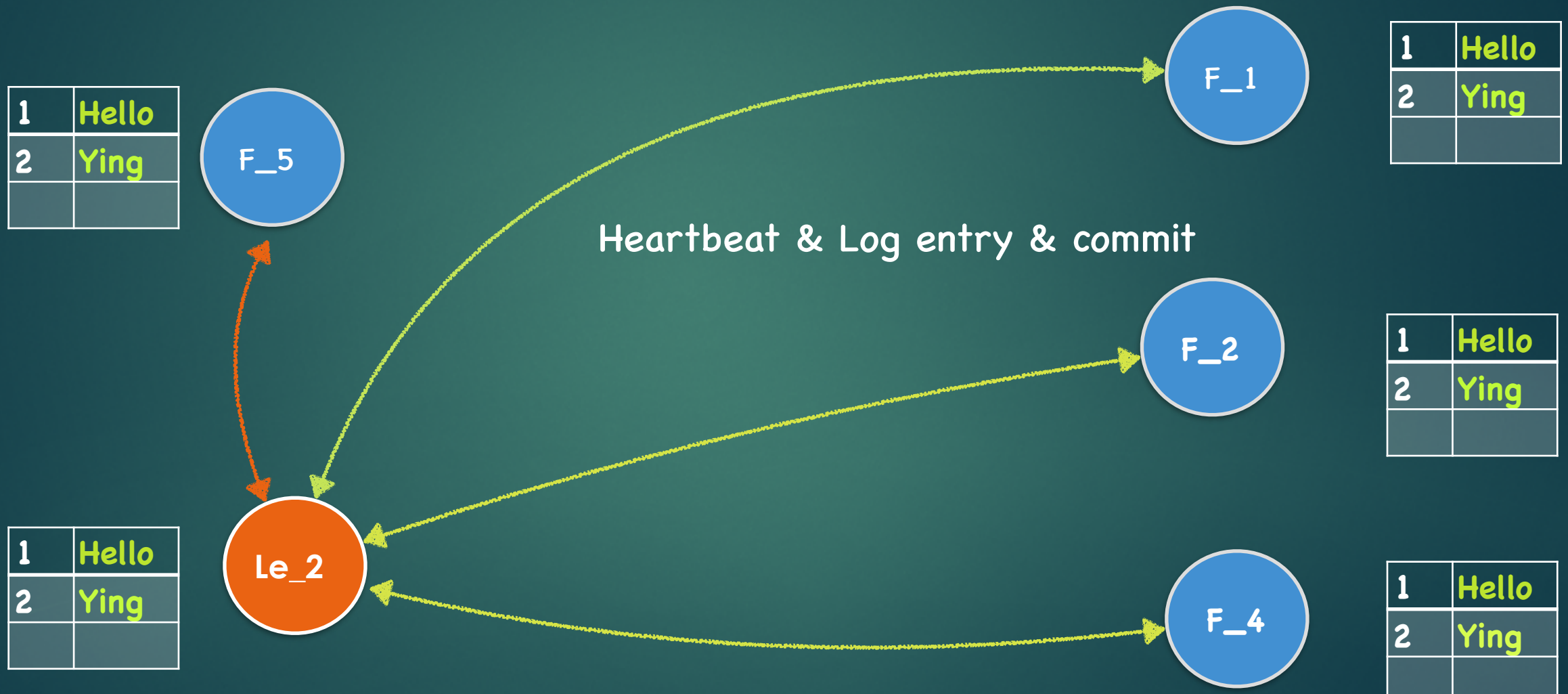
新集群正常


1 Hello
2 Tim
Le_1
两个人怎么够法定人数 ！！！
F_1
1 Hello
2 Tim
F_2
1 Hello
2 Ying
1 Hello
2 Ying
F_3
Heartbeat & Log entry & commit
F_4
1 Hello
2 Ying

网络恢复


1 Hello
Le_1
网络好了后, 开始抢夺Leader
Le_1 term 小于 Le_2 !
F_1
1 Hello
F_2
1 Hello
2 Ying
1 Hello
2 Ying
Le_2
Heartbeart & Append Log Entries
F_4
1 Hello
2 Ying

一致性


1 Hello
2 Ying
F_5
F_1
1 Hello
2 Ying
Heartbeat & Log entry & commit
F_2
1 Hello
2 Ying
1 Hello
2 Ying
Le_2
F_4
1 Hello
2 Ying

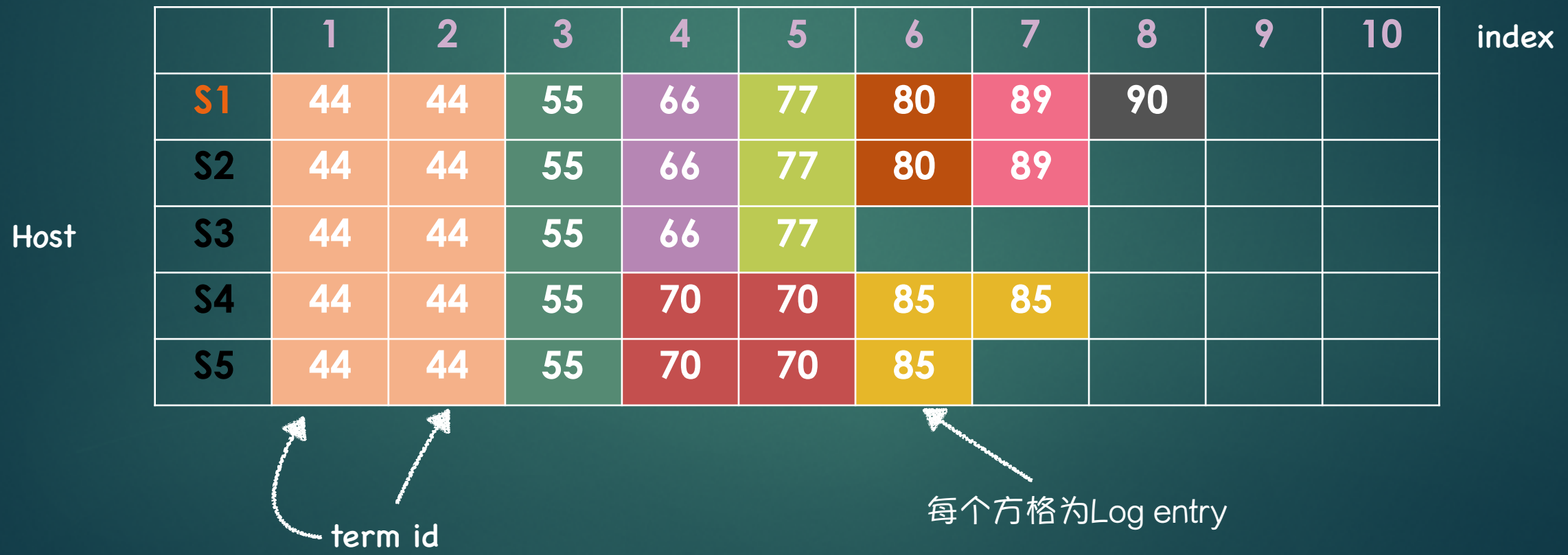
冲突Split brain

- 如符合法定人数并产生了N条数据 与 新集群怎么保持数据一致性
  - 覆盖 VS 合并？
- 被分区前有些node没有收到commit ？
  - timer check

预防Split brain

- 单播制定节点
- 指定法定人数，每次add\reduce都需要更改
- 加大timeout，retry
- 统一 client 入口，But …
- 监控脑裂情况，反查各个node的leader是否一致

复杂一致性


index
Host
1
2
3
4
5
6
7
8
9
10
S1
44
44
55
66
77
80
89
90
S2
44
44
55
66
77
80
89
S3
44
44
55
66
77
S4
44
44
55
70
70
85
85
S5
44
44
55
70
70
85
term id
每个方格为Log entry

Log compress

	1	2	3	4	5	6	7	8	9	10	index
S1	44	44	55	66	77	80	89	90			

Snapshot
Last included index : 6
Last included term : 80
state macheie state:
x <— 0
y <— 9

all commited !!!

study

动画演示:

https://ongardie.github.io/raft-talk-archive/2015/buildstuff/raftscope-replay/

文档:

http://en.youscribe.com/catalogue/tous/professional-resources/it-systems/raft-in-search-of-an-understandable-consensus-algorithm-2088704

Googole ...

Q & A