

BT

- [投稿](#)
 - [活动大本营](#)
 - [关于我们](#)
 - [合作伙伴](#)
 - [EGO](#)
 - [StuQ](#)
 - [GIT](#)
- 欢迎关注我们的：
- -
 -

InfoQ - 促进软件开发领域知识与创新的传播

[登录](#)



- [En](#)
- [中文](#)
- [日本](#)
- [Fr](#)
- [Br](#)

966,690 三月 独立访问用户

- [语言 & 开发](#)
 - [Java](#)
 - [Clojure](#)
 - [Scala](#)
 - [.Net](#)
 - [移动](#)
 - [Android](#)
 - [iOS](#)
 - [HTML 5](#)
 - [JavaScript](#)
 - [函数式编程](#)
 - [Web API](#)

特别专题 语言 & 开发

[为什么我不再使用MVC框架](#)



用户界面使用MVC模式已经有数十年的历史了，但是它并没有与现代应用的需求相匹配。
为了满足新的需求并提升开发的速度，Jean-Jacques Dubray引入一个新的模式：状态-行为-模型（State-Action-Model，SAM）。 SAM是一个函数式反应型的编程模式，它致力于简化数据Model和View之间的交互。

[浏览所有 语言 & 开发](#)

- [架构 & 设计](#)

- 架构
- 企业架构
- 性能和可伸缩性
- 设计
- 案例分析
- 设计模式
- 安全

特别专题 架构 & 设计

[为什么我不再使用MVC框架](#)



用户界面使用MVC模式已经有数十年的历史了，但是它并没有与现代应用的需求相匹配。
为了满足新的需求并提升开发的速度，Jean-Jacques Dubray引入一个新的模式：状态-行为-模型（State-Action-Model，SAM）。 SAM是一个函数式反应型的编程模式，它致力于简化数据Model和View之间的交互。

[浏览所有 架构 & 设计](#)

- [数据科学](#)

- 大数据
- NoSQL
- 数据库

特别专题 数据科学

IAP:HTTP的替代者，更快、更丰富



现如今物联网横行天下，大家都在抢夺这块地盘，技术当仁不让、创新不断，从操作系统、到各种层出不穷到嵌入式设备、再到云计算等等，那么是否协议就采用一万年不变的 HTTP 了呢？

浏览所有 数据科学

- 文化 & 方法
 - Agile
 - 领导能力
 - 团队协作
 - 测试
 - 用户体验
 - Scrum
 - 精益

特别专题 文化 & 方法

中国顶尖技术团队访谈录·第五季



本次的《中国顶尖技术团队访谈录》·第四季挑选的五个团队虽然都来自互联网企业，却是风格各异。希望通过这样的记录，能够让一家家品牌背后的技术人员形象更加鲜活，让更多人感受到他们的可爱与坚持。

浏览所有 文化 & 方法

- DevOps
 - 持续交付
 - 自动化操作
 - 云计算

特别专题 DevOps

[中国顶尖技术团队访谈录·第五季](#)



本次的《中国顶尖技术团队访谈录》·第四季挑选的五个团队虽然都来自互联网企业，却是风格各异。希望通过这样的记录，能够让一家家品牌背后的技术人员形象更加鲜活，让更多人感受到他们的可爱与坚持。

[浏览所有 DevOps](#)

- [架构](#)
- [移动](#)
- [Docker](#)
- [云计算](#)
- [大数据](#)
- [架构师](#)
- [运维](#)
- [QCon](#)
- [ArchSummit](#)
- [AWS](#)
- [腾讯云](#)
- [活动大本营](#)

[全部话题](#)

您目前处于：[InfoQ首页](#) [文章](#) CoreOS 实战：剖析 etcd

CoreOS 实战：剖析 etcd



作者 [桂阳](#) 发布于 2014年11月20日 / 注意:QCon全球软件开发大会（北京）2016年4月21-23日，[了解更多详情！](#)

[8 讨论](#)

分享到：[微博](#) [微信](#) [Facebook](#) [Twitter](#) [有道云笔记](#) [邮件分享](#)

● "稍后阅读"

- "我的阅读清单"

【编者按】CoreOS是一个基于Docker的轻量级容器化Linux发行版，专为大型数据中心而设计，旨在通过轻量的系统架构和灵活的应用程序部署能力简化数据中心的维护成本和复杂度。CoreOS作为Docker生态圈中的重要一员，日益得到各大云服务商的重视，目前已经完成了A轮融资，发展风头正劲。InfoQ希望《CoreOS实战》系列文章能够帮助读者了解CoreOS以及相关的使用方法。如果说Docker是下一代的虚拟机，那CoreOS就应该是下一代的服务器Linux，InfoQ愿和您一起探索这个新生事物。另外，欢迎加入InfoQ Docker技术交流群，QQ群号：272489193。

1. 概述

etcd 是一个应用在分布式环境下的 key/value 存储服务。利用 etcd 的特性，应用程序可以在集群中共享信息、配置或作服务发现，etcd 会在集群的各个节点中复制这些数据并保证这些数据始终正确。etcd 无论是在 CoreOS 还是 Kubernetes 体系中都是不可或缺的一环。笔者由于项目的原因对 etcd 进行了一些了解，也已经使用了一段时间。同时在与同行的交流中常被问到 etcd 是什么、与 ZooKeeper 有什么不同。那么借着 etcd 0.5.0 即将发布的机会，向感兴趣的读者介绍一下 etcd，希望可以帮助读者了解 etcd 的工作原理以及具体实现，同时也作为 CoreOS 实战的第二部分内容为后面相关部分进行铺垫。

随着 etcd 0.5.0 alpha（本文完稿时为 etcd 0.5.0 alpha.3）版发布，etcd 将在未来几周内迎来一次重要的更新。在 0.5.0 版里除了修复现有稳定版中的一系列 Bug 之外，一些新的特性和变更也将随之发布。这些变化将提升 etcd 服务安全性、可靠性和可维护性。

新的特性包括

- 规范词汇表；
- 新的 raft 算法实现；
- 新增 etcd node 身份标记；
- WAL (Write-ahead logging) 增加 CRC 校验；
- API 中新增 member {list, add, remove} 接口，原来的 list machines 接口将被移除，未来 etcd 集群中将不存在 machine 的称呼；
- 两个主要端口变更为 2379 (for client) 与 2380 (for peer/raft) 并成为 IANA 的注册端口。

重要的变更包括

- 运行时重构 (runtime reconfiguration)。用户不需要重启 etcd 服务即可实现对 etcd 集群结构进行变更。
- 摒弃通过配置文件配置 etcd 属性的方式，转而以 CLI (command-line interface) flags 或环境变量的方式实现 etcd 节点的配置。
- 每个节点可监听多个广播地址。监听的地址由原来的一个扩展到多个，用户可以根据需求实现更加复杂的集群环境，如搭建一个混合了私有云与公有云服务的 etcd 集群。
- 新增 proxy mode。

2. 规范词汇表

etcd 0.5.0 版首次对 etcd 代码、文档及 CLI 中使用的术语进行了定义。

2.1. node

node 指一个 raft 状态机实例。每个 node 都具有唯一的标识，并在处于 leader 状态时记录其它节点的步进数。

2.2. member

member 指一个 etcd 实例。member 运行在每个 node 上，并向这一 node 上的其它应用程序提供服务。

2.3. Cluster

Cluster 由多个 member 组成。每个 member 中的 node 遵循 raft 共识协议来复制日志。Cluster 接收来自 member 的提案消息，将其提交并存储于本地磁盘。

2.4. Peer

同一 Cluster 中的其它 member。

2.5. Client

Client 指调用 Cluster API 的对象。

3. Raft 共识算法

etcd 集群的工作原理基于 raft 共识算法 ([The Raft Consensus Algorithm](#))。etcd 在 0.5.0 版本中重新实现了 raft 算法，而非像之前那样依赖于第三方库 [go-raft](#)。raft 共识算法的优点在于可以在高效的解决分布式系统中各个节点日志内容一致性问题的同时，也使得集群具备一定的容错能力。即使集群中出现部分节点故障、网络故障等问题，仍可保证其余大多数节点正确的步进。甚至当更多的节点（一般来说超过集群节点总数的一半）出现故障而导致集群不可用时，依然可以保证节点中的数据不会出现错误的结果。

3.1. 集群建立与状态机

raft 集群中的每个节点都可以根据集群运行的情况在三种状态间切换：follower, candidate 与 leader。leader 向 follower 同步日志，follower 只从 leader 处获取日志。在节点初始启动时，节点的 raft 状态机将处于 follower 状态并被设定一个 election timeout，如果在这一时间周期内没有收到来自 leader 的 heartbeat，节点将发起选举：节点在将自己的状态切换为 candidate 之后，向集群中其它 follower 节点发送请求，询问其是否选举自己成为 leader。当收到来自集群中过半数节点的接受投票后，节点即成为 leader，开始接收保存 client 的数据并向其它的 follower 节点同步日志。leader 节点依靠定时向 follower 发送 heartbeat 来保持其地位。任何时候如果其它 follower 在 election timeout 期间都没有收到来自 leader 的 heartbeat，同样会将自己的状态切换为 candidate 并发起选举。每成功选举一次，新 leader 的步进数都会比之前 leader 的步进数大1。

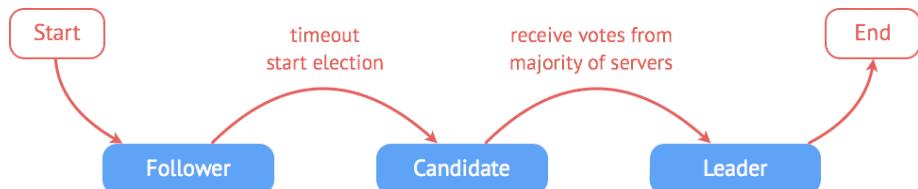


图 3-1 raft 状态切换示意图

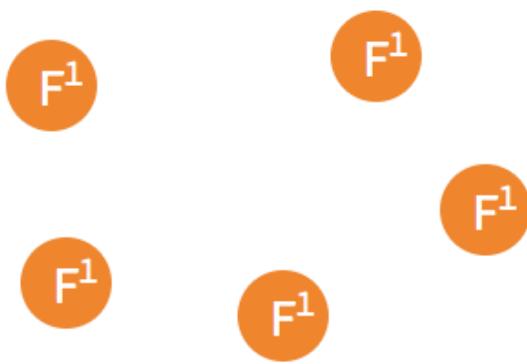
3.2. 选举

3.2.1. 一个 candidate 成为 leader 需要具备三个要素：

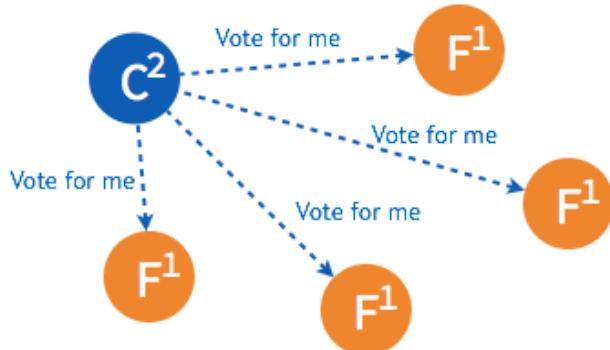
- 获得集群多数节点的同意；
- 集群中不存在比自己步进数更高的 candidate；
- 集群中不存在其他 leader。

3.2.2. 下面为一个 etcd 集群选举过程的简单描述：

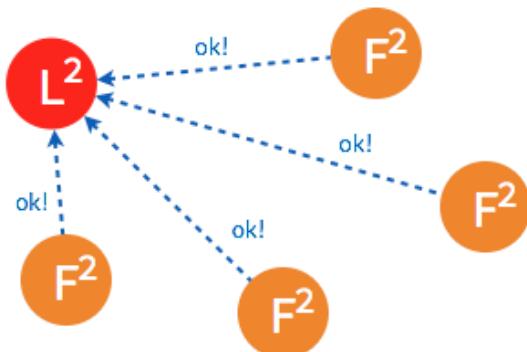
➤ 初始状态下集群中的所有节点都处于 follower 状态。



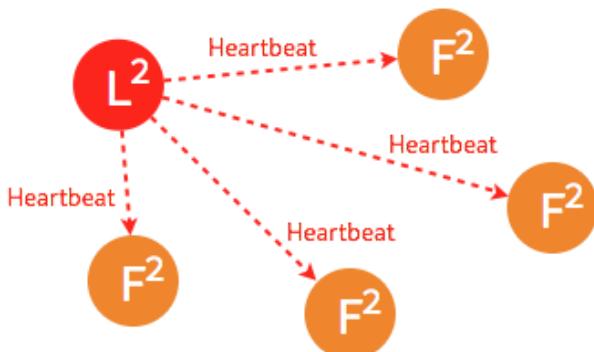
➤ 某一时刻，其中的一个 follower 由于没有收到 leader 的 heartbeat 率先发生 election timeout 进而发起选举。



➤ 只要集群中超过半数的节点接受投票，candidate 节点将成为即切换 leader 状态。



➤ 成为 leader 节点之后，leader 将定时向 follower 节点同步日志并发送 heartbeat。



3.3. 节点异常

集群中各个节点的状态随时都有可能发生变化。从实际的变化上来分类的话，节点的异常大致可以分为四种类

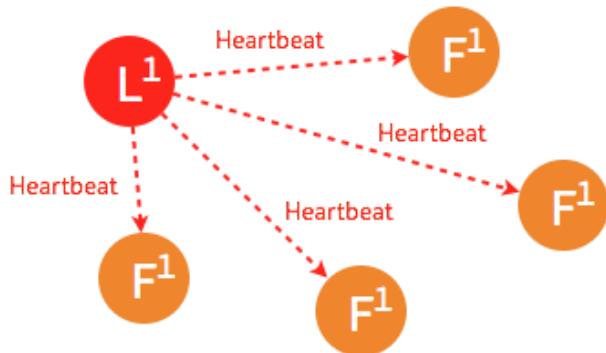
型：

- leader 不可用；
- follower 不可用；
- 多个 candidate 或多个 leader；
- 新节点加入集群。

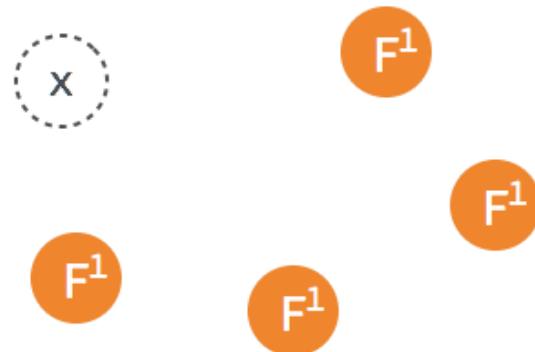
3.3.1. leader 不可用

下面将说明当集群中的 leader 节点不可用时，raft 集群是如何应对的。

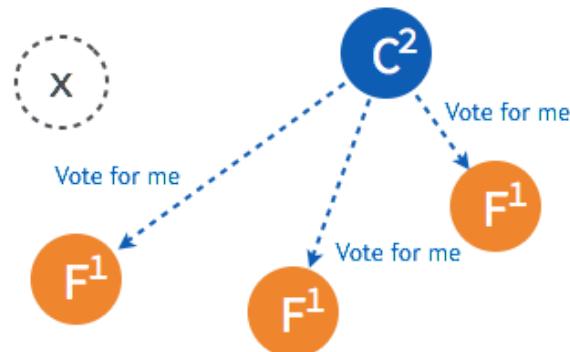
➤ 一般情况下，leader 节点定时发送 heartbeat 到 follower 节点。



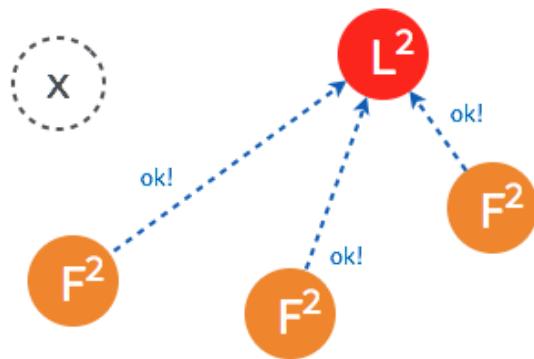
➤ 由于某些异常导致 leader 不再发送 heartbeat，或 follower 无法收到 heartbeat。



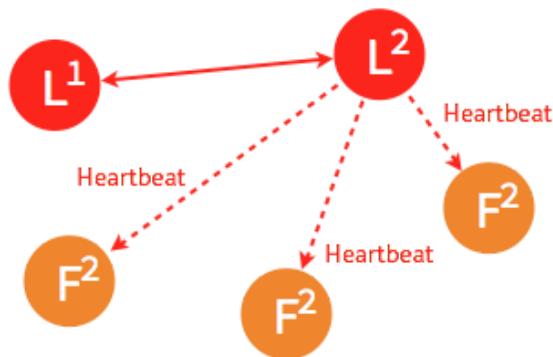
➤ 当某一 follower 发生 election timeout 时，其状态变更为 candidate，并向其他 follower 发起投票。



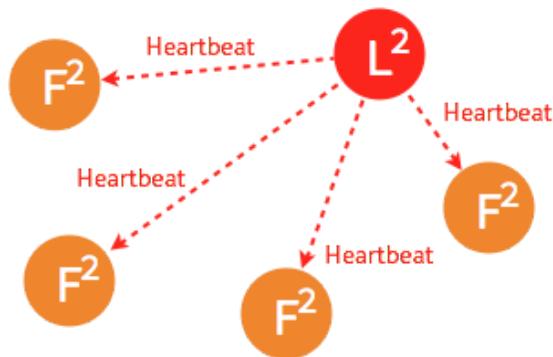
➤ 当超过半数的 follower 接受投票后，这一节点将成为新的 leader，leader 的步进数加1并开始向 follower 同步日志。



当一段时间之后，如果之前的 leader 再次加入集群，则两个 leader 比较彼此的步进数，步进数低的 leader 将切换自己的状态为 follower。



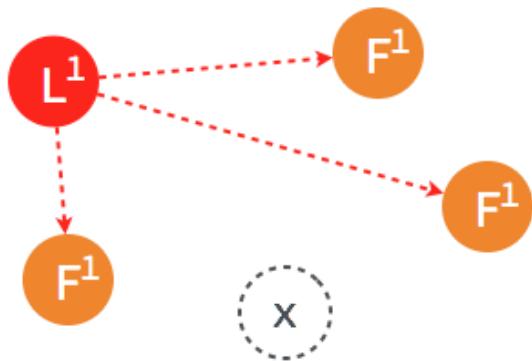
较早前 leader 中不一致的日志将被清除，并与现有 leader 中的日志保持一致。



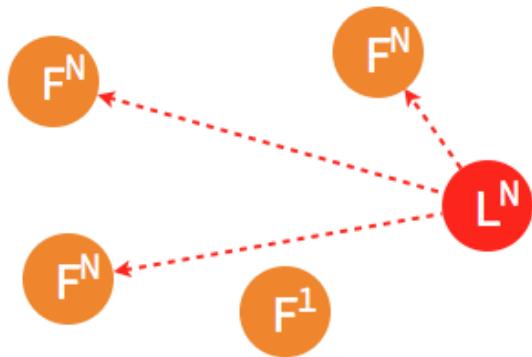
3.3.2. follower 节点不可用

follower 节点不可用的情况相对容易解决。因为集群中的日志内容始终是从 leader 节点同步的，只要这一节点再次加入集群时重新从 leader 节点处复制日志即可。

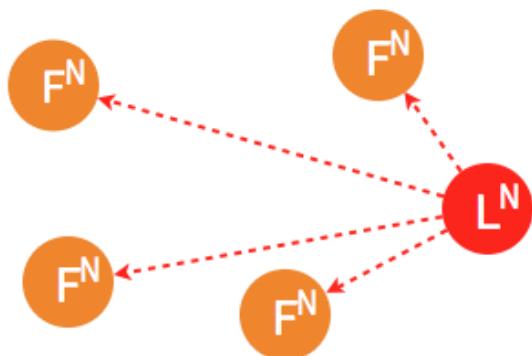
集群中的某个 follower 节点发生异常，不再同步日志以及接收 heartbeat。



➤ 经过一段时间之后，原来的 follower 节点重新加入集群。



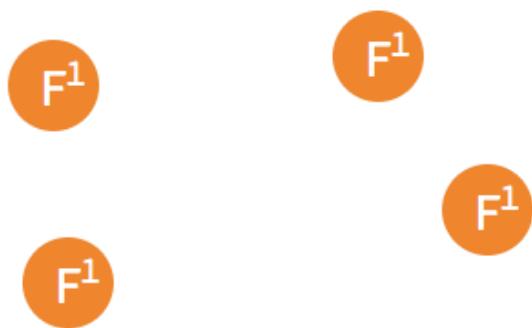
➤ 这一节点的日志将从当时的 leader 处同步。



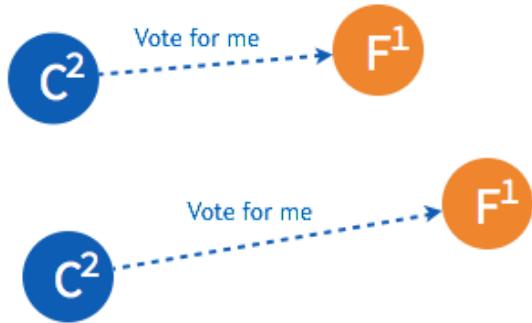
3.3.3. 多个 candidate 或多个 leader

在集群中出现多个 candidate 或多个 leader 通常是由于数据传输不畅造成的。出现多个 leader 的情况相对少见，但多个 candidate 比较容易出现在集群节点启动初期尚未选出 leader 的“混沌”时期。

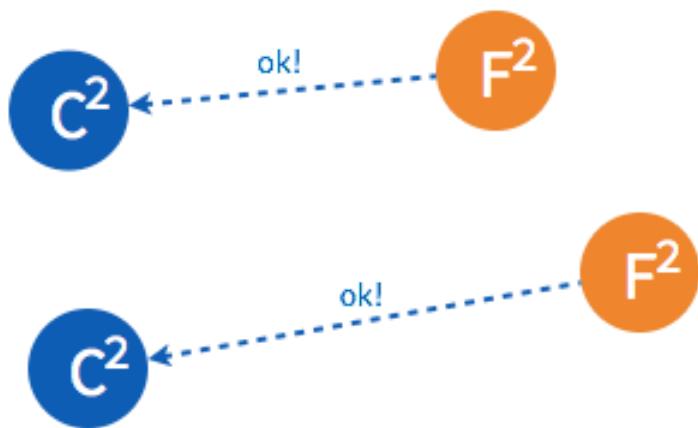
➤ 初始状态下集群中的所有节点都处于 follower 状态。



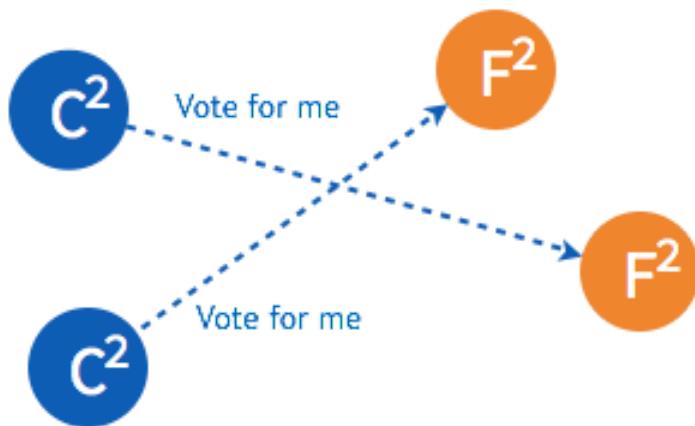
➤ 两个节点同时成为 candidate 发起选举。



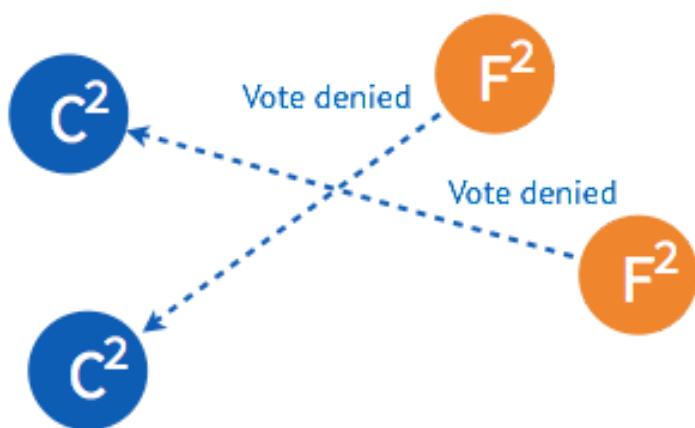
➤ 两个 candidate 都只得到了少部分 follower 的接受投票。



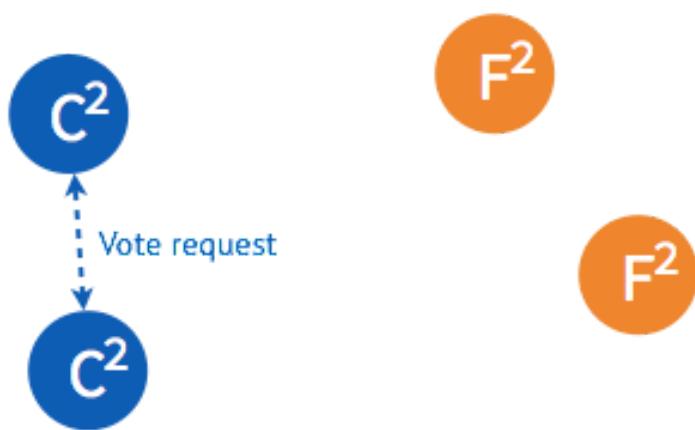
➤ candidate 继续向其他的 follower 询问。



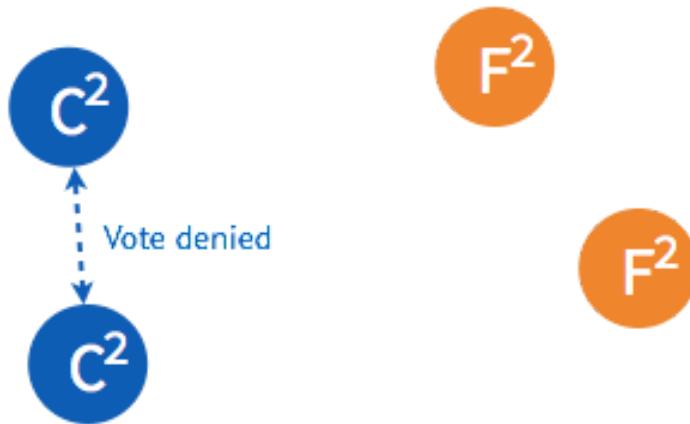
➤ 由于一些 follower 已经投过票了，所以均返回拒绝接受。



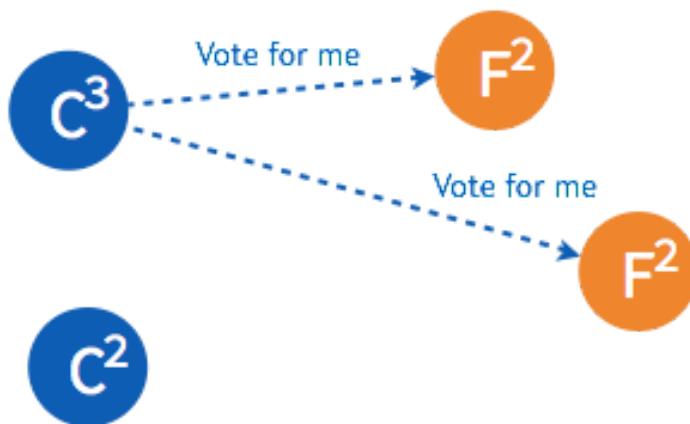
➤ candidate 也可能向一个 candidate 询问投票。



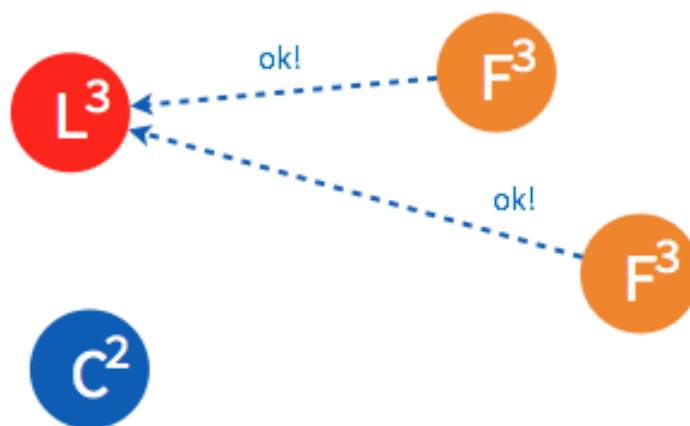
➤ 在步进数相同的情况下，candidate 将拒绝接受另一个 candidate 的请求。



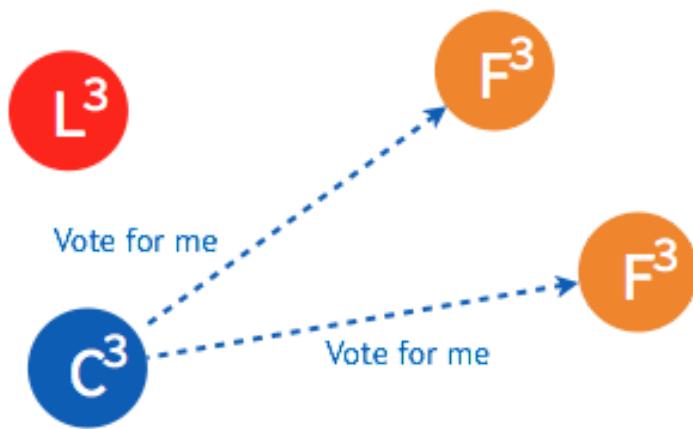
➤ 由于第一次未选出 leader, candidate 将随机选择一个等待间隔 (150ms ~ 300ms) 再次发起投票。



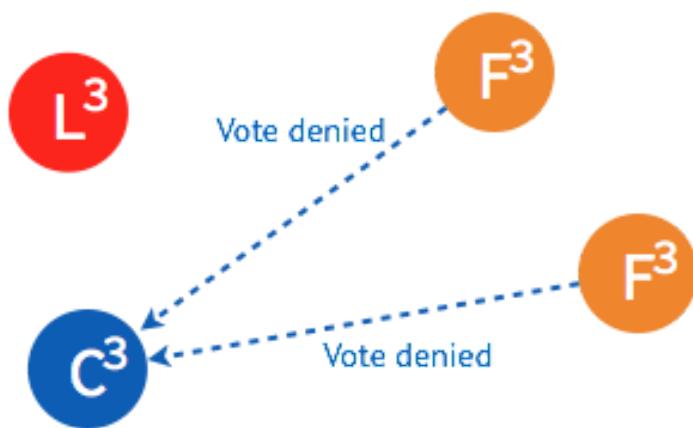
➤ 如果得到集群中半数以上的 follower 的接受, 这一 candidate 将成为 leader。



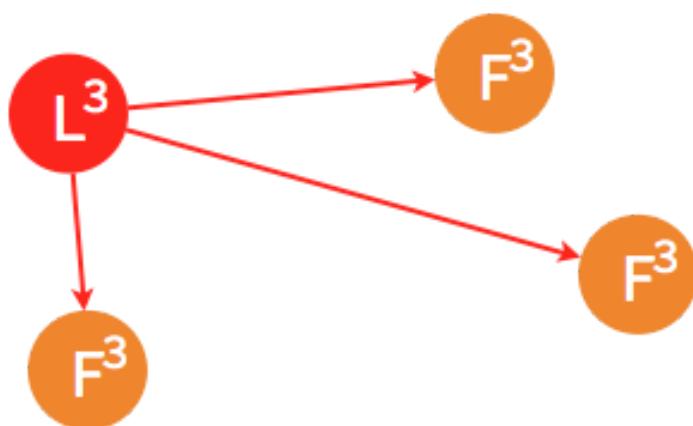
➤ 稍后另一个 candidate 也将再次发起投票。



➤ 由于集群中已经选出 leader，candidate 将收到拒绝接受的投票。



➤ 在被多数节点拒绝之后，并已知集群中已存在 leader 后，这一 candidate 节点将终止投票请求、切换为 follower，从 leader 节点同步日志。

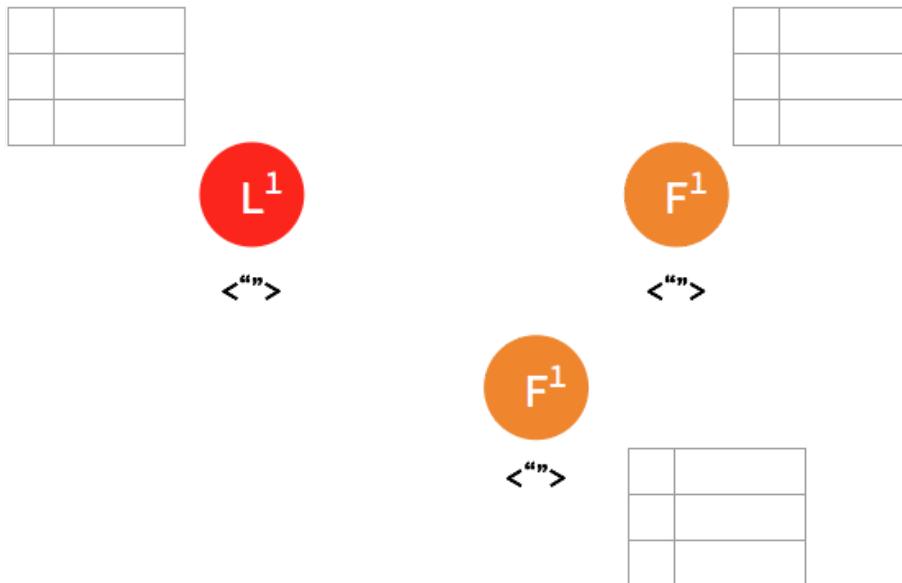


3.4. 日志

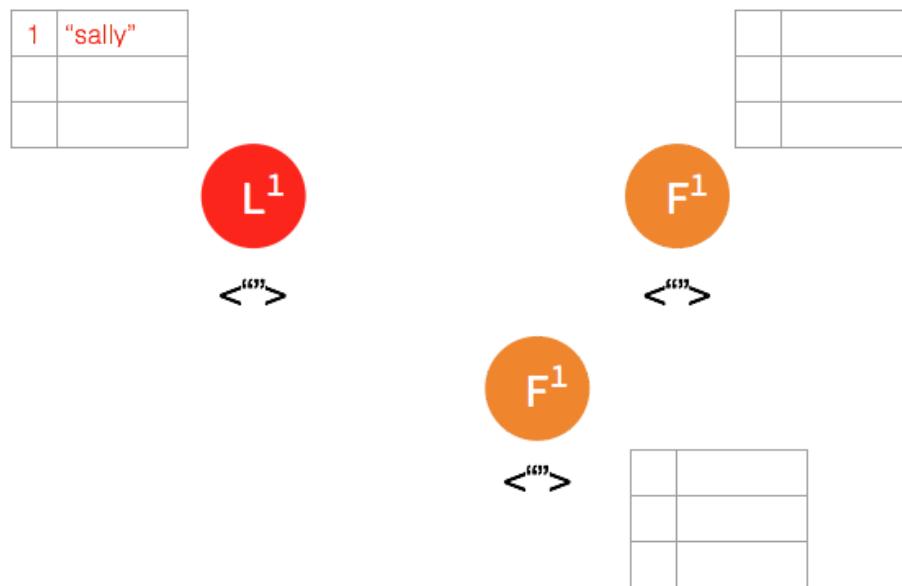
3.4.1. 复制

在 raft 集群中，所有日志都必须首先提交至 leader 节点。leader 在每个 heartbeat 向 follower 同步日志，follower

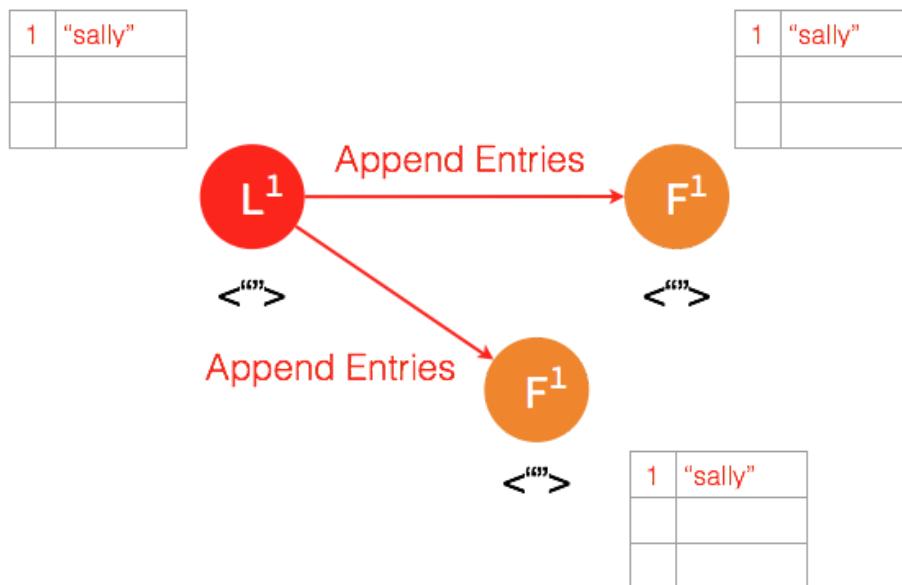
在收到日志之后向 leader 反馈结果，leader 在确认日志内容正确之后将此条目提交并存储于本地磁盘。



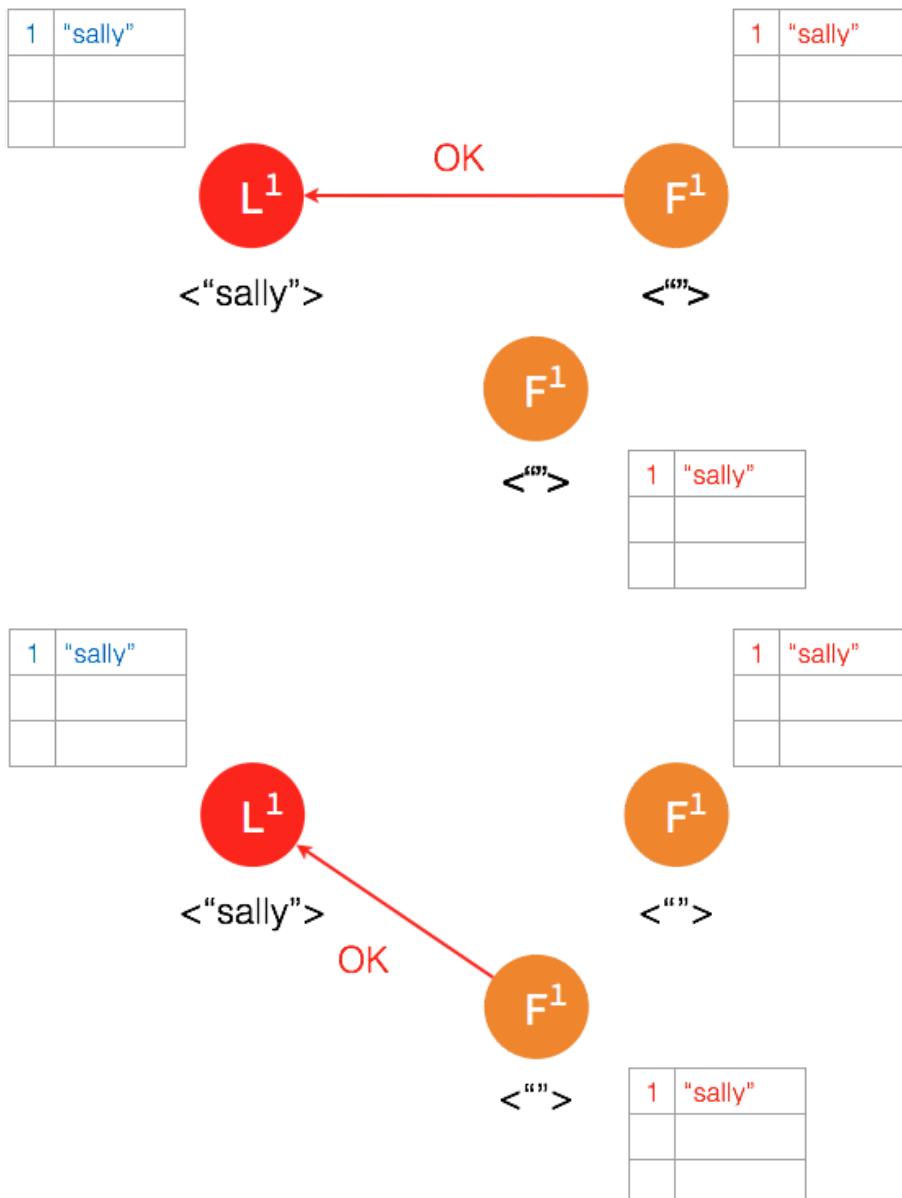
➤ 首先有一条 uncommitted 的日志条目提交至 leader 节点。



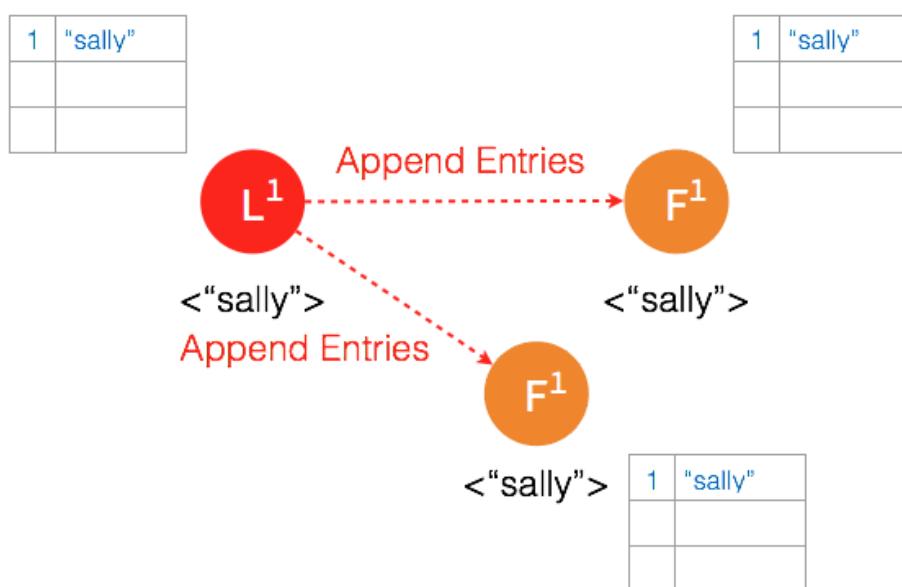
➤ 在下一个 heartbeat，leader 将此条目复制给所有的 follower。



➤ 当大多数节点记录此条目之后，leader 节点认定此条目有效，将此条目设定为已提交并存储于本地磁盘。



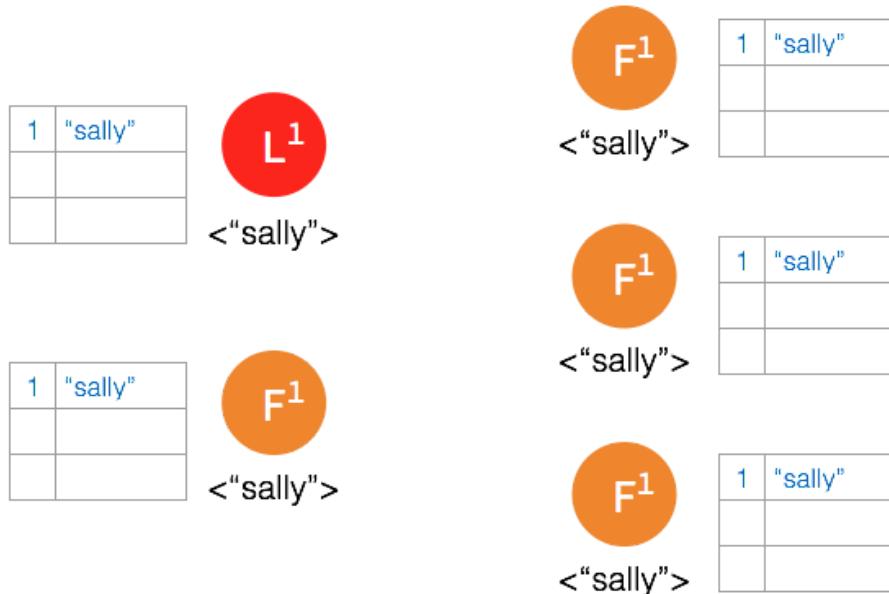
➤ 在下一个 heartbeat，leader 通知所有 follower 提交这一日志条目并存储于各自的磁盘内。



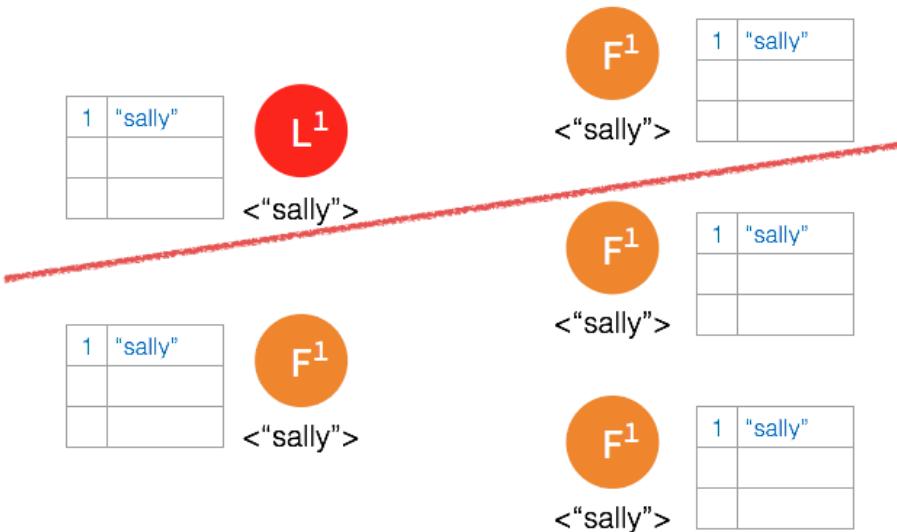
3.4.2. 容错

如果由于网络的隔断，造成集群中多数的节点在一段时间内无法访问到 leader 节点。按照 raft 共识算法，没有 leader 的那一组集群将会通过选举投票出新的 leader，甚至会在两个集群内产生不一致的日志条目。在集群重新完整连通之后，原来的 leader 仍会按照 raft 共识算法从步进数更高的 leader 同步日志并将自己切换为 follower。

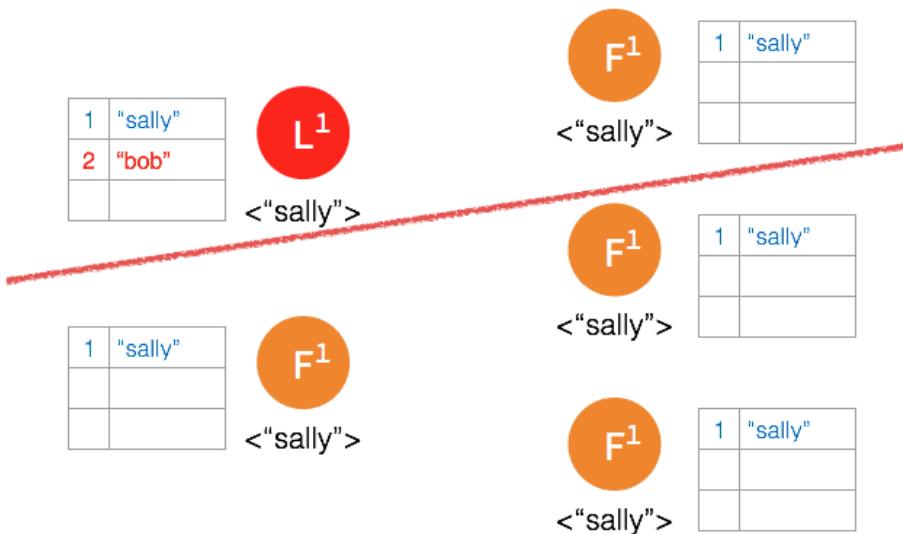
➤ 集群的理想状态。



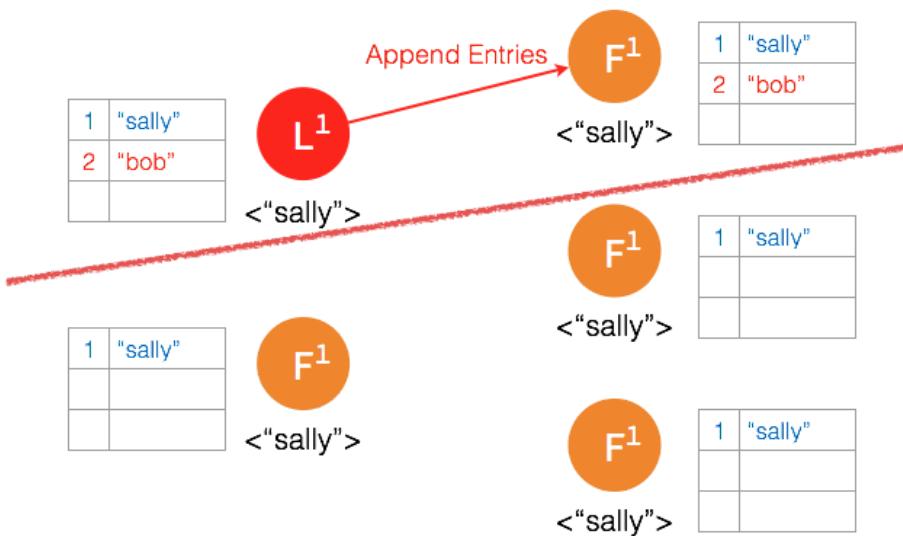
➤ 网络间隔造成大多数的节点无法访问 leader 节点。



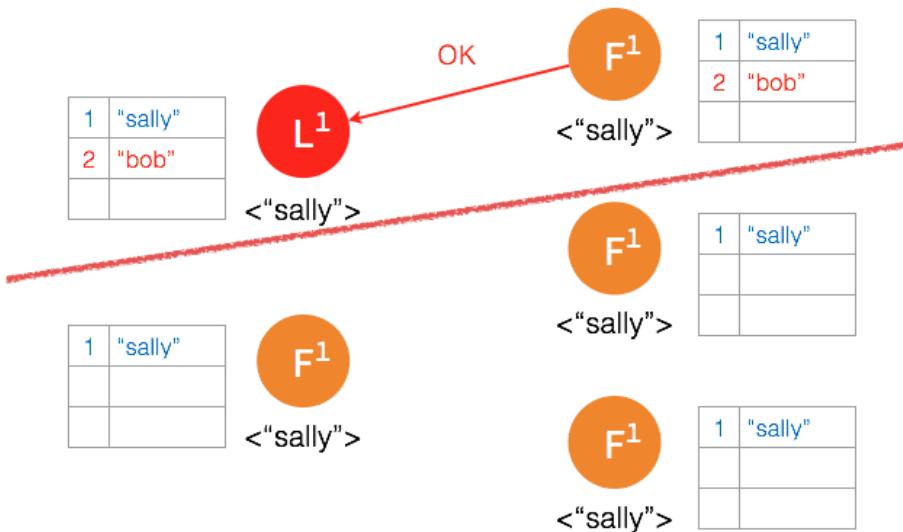
➤ 新的日志条目添加到 leader 中。



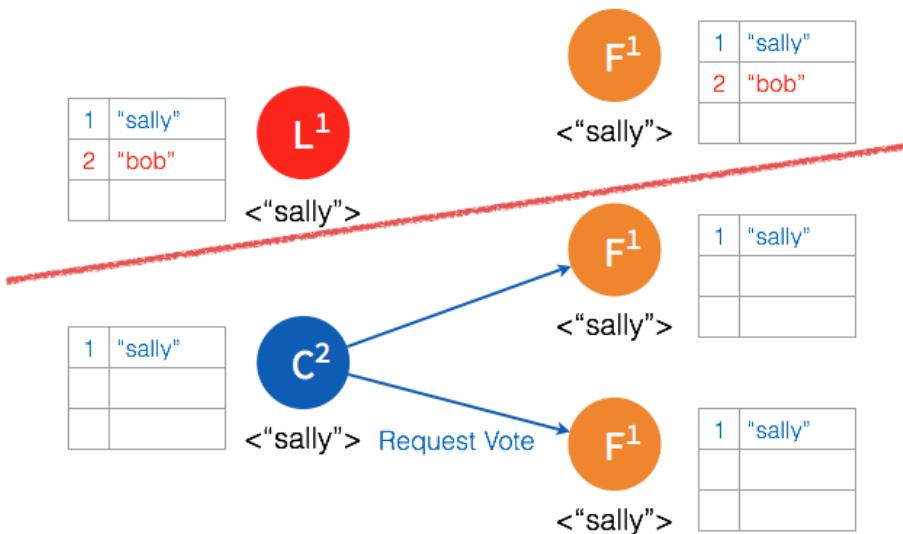
➤ leader 节点将此条日志同步至能够访问到 leader 的节点。



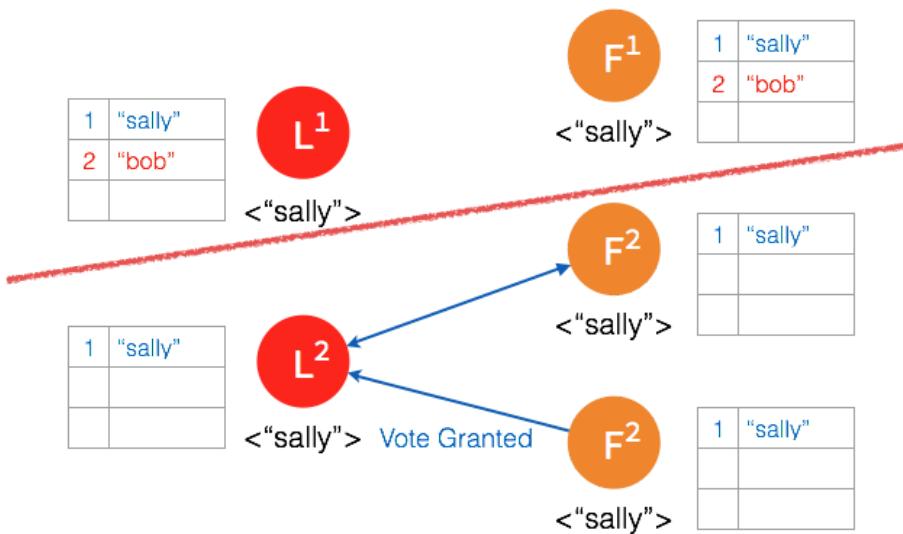
➤ follower 确认日志被记录，但是确认记录日志的 follower 数量没有超过集群节点的半数，leader 节点并不将此条日志存档。



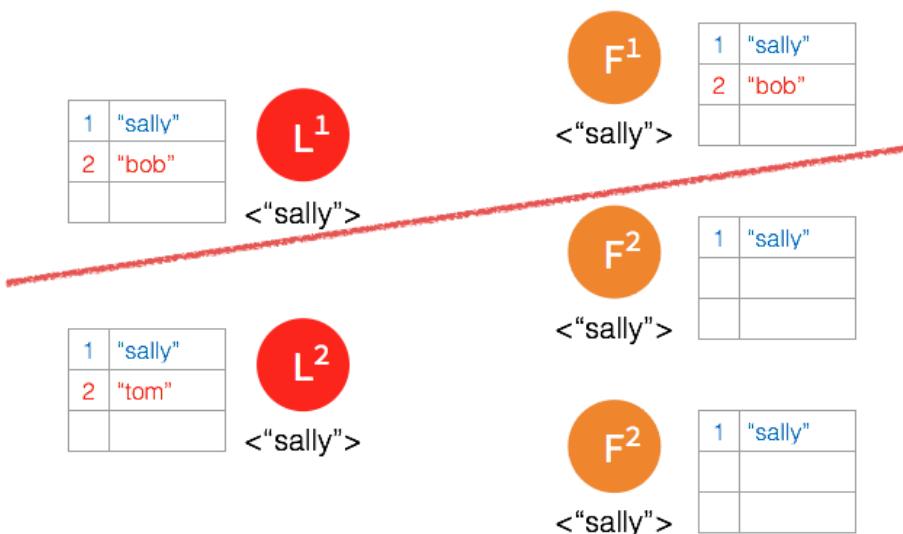
➤ 在被隔断的这部分节点，在 election timeout 之后，followers 中产生 candidate 并发起选举。



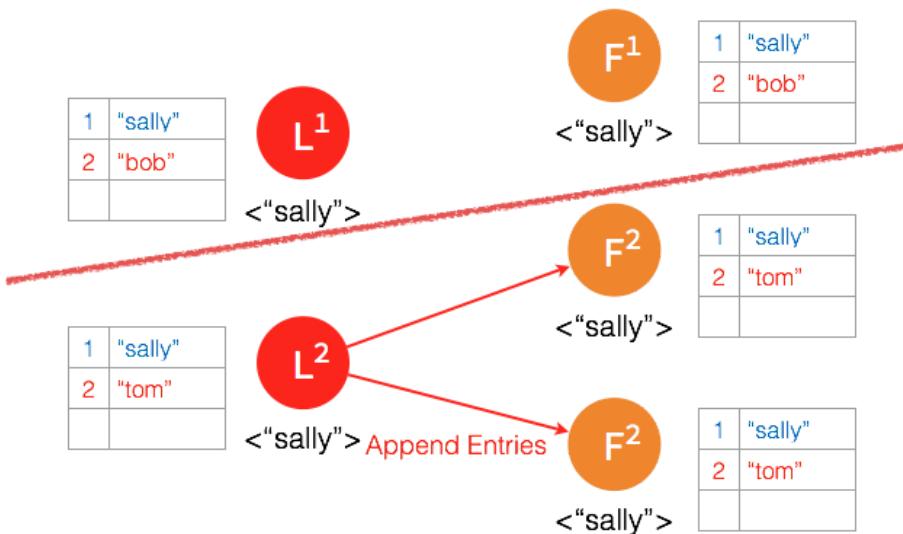
➤ 多数节点接受投票之后，candidate 成为 leader。



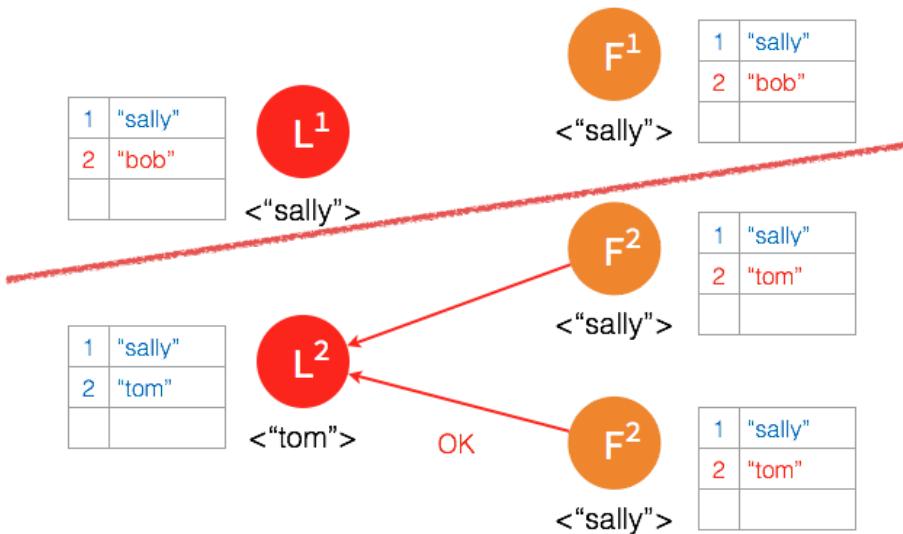
➤ 一个日志条目被添加到新的 leader。



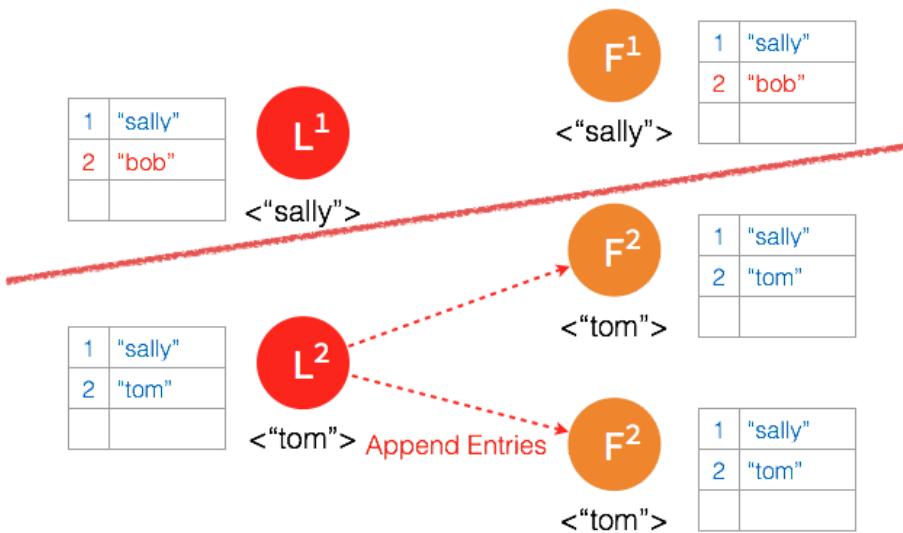
➤ 日志被复制给新 leader 的 follower。



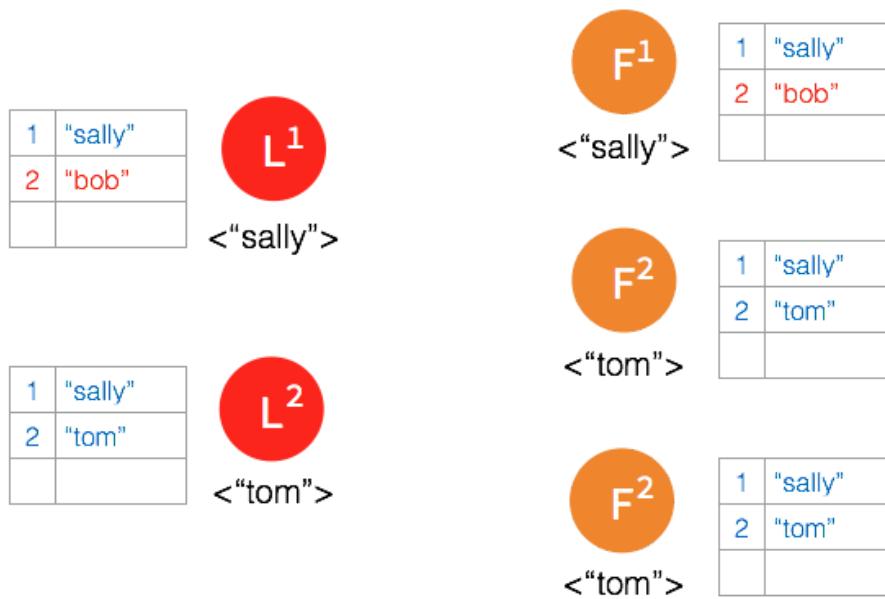
➤ 多数节点确认之后，leader 将日志条目提交并存储。



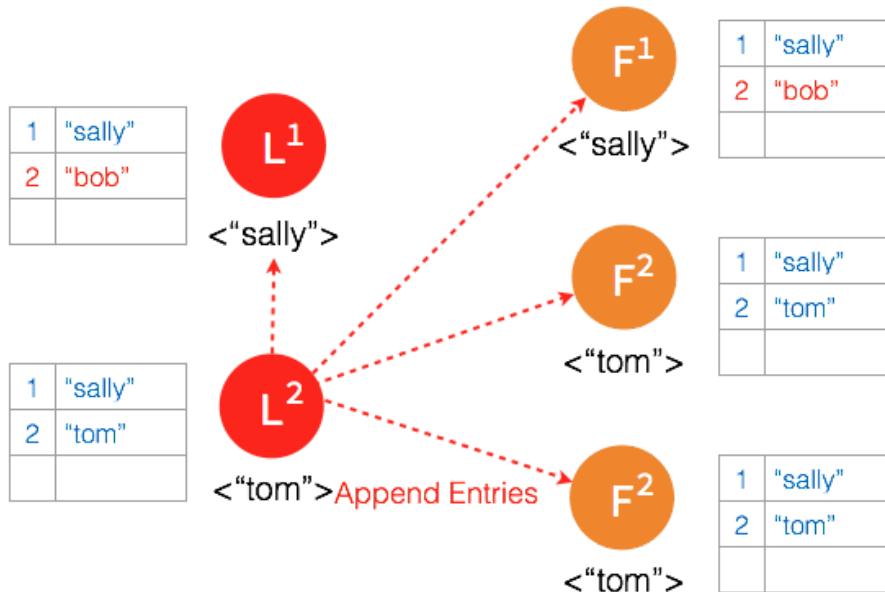
➤ 在下一个 heartbeat，leader 通知 follower 各自提交并保存在本地磁盘。



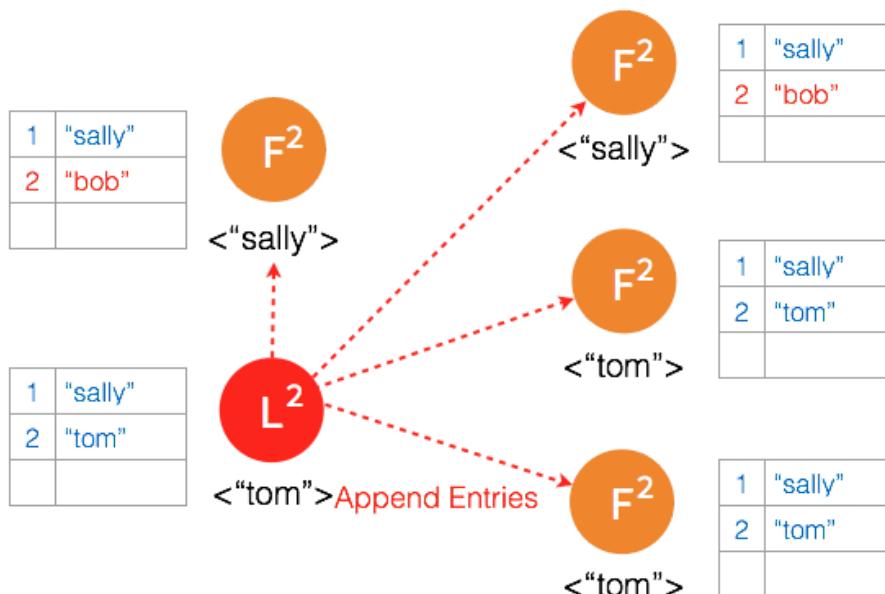
➤ 经过一段时间之后，集群重新连通到一起，集群中出现两个 leader 并且存在不一致的日志条目。



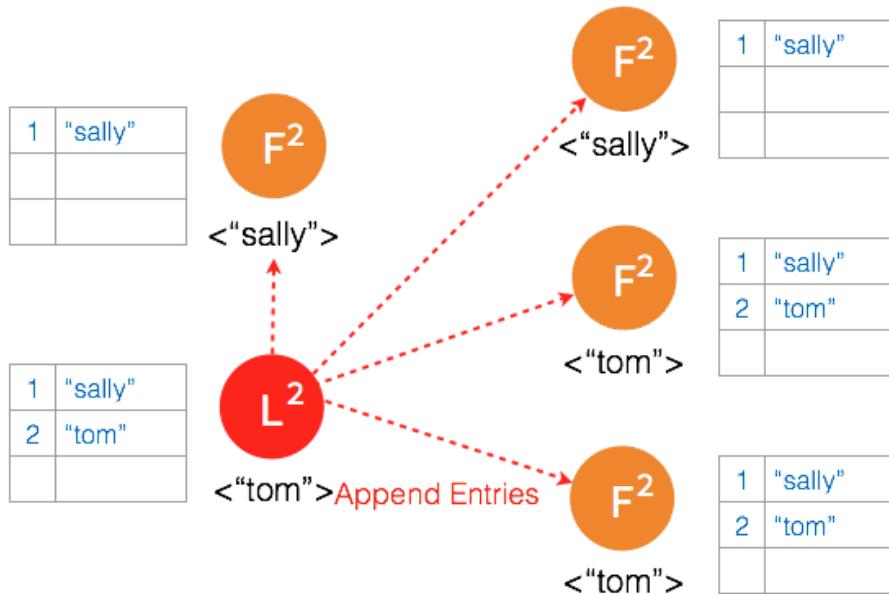
➤ 新的 leader 在下一次 heartbeat timeout 时向所有的节点发送一次 heartbeat。



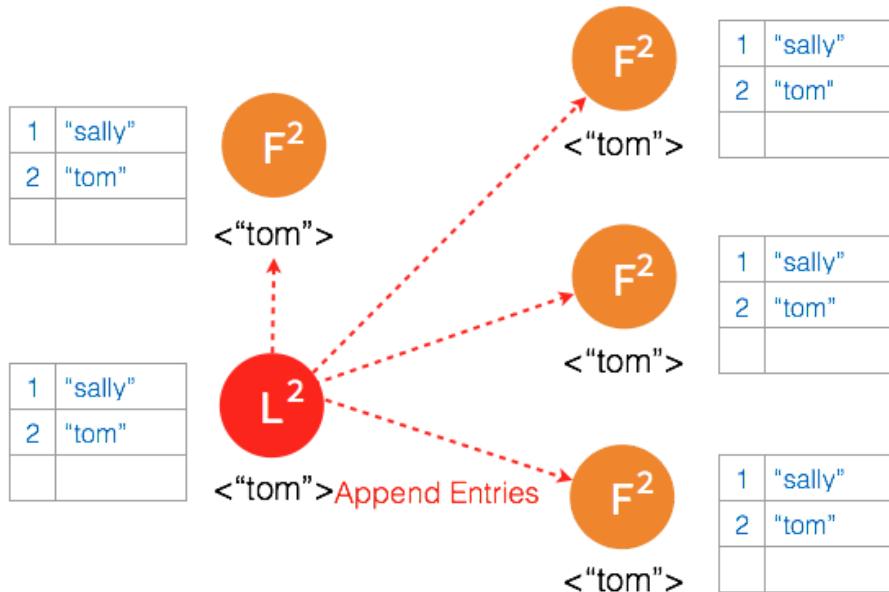
➤ #1 leader 在收到步进数更高的 #2 leader heartbeat 时放弃 leader 地位并切换到 follower 状态。



➤ 节点中所有未存档的日志条目都将被丢弃。



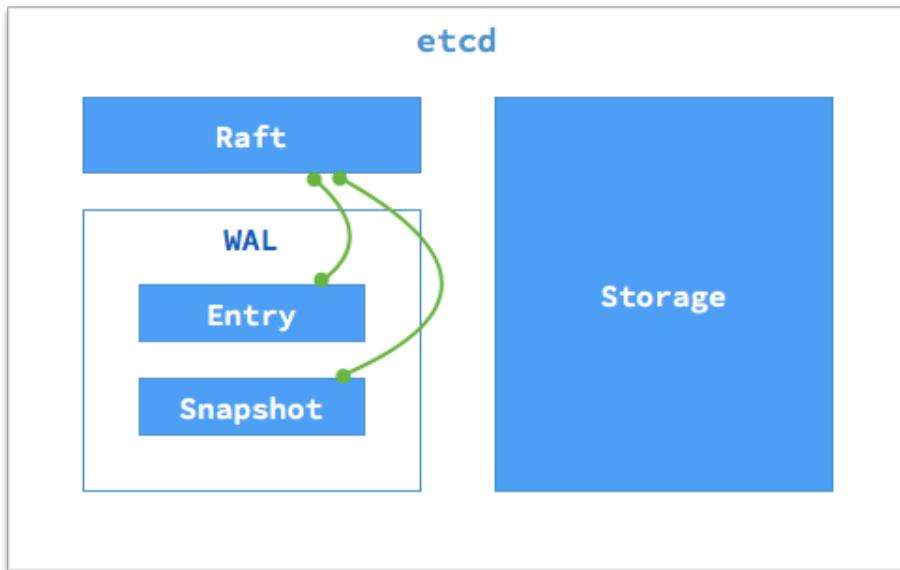
➤ 未被复制的日志条目将会被同步给所有的 follower。



通过这种方式，只要集群中有效连接的节点超过总数的一半，集群将一直以这种规则运行下去并始终确保各个节点中的数据始终一致。

4. 实现

4.1. etcd 结构



一个 etcd 节点的核心由三部分组成：

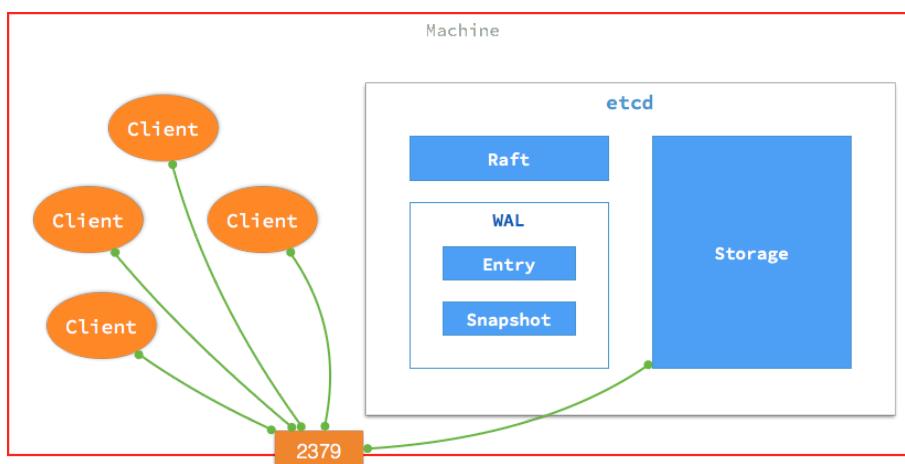
- Raft: raft 状态机是对 raft 共识算法的实现
- WAL: raft 日志存储
- Storage: 数据的存储与索引

WAL (Write-ahead logging)，是用于向系统提供原子性和持久性的一系列技术。在使用 WAL 的系提供中，所有的修改在提交之前都要先写入 log 文件中。etcd 的 WAL 由日志存储与快照存储两部分组成，其中 Entry 负责存储具体日志的内容，而 Snapshot 负责在日志内容发生变化的时候保存 raft 的状态。WAL 会在本地磁盘的一个指定目录下分别日志条目与快照内容。

4.2. 服务

4.2.1. Clients

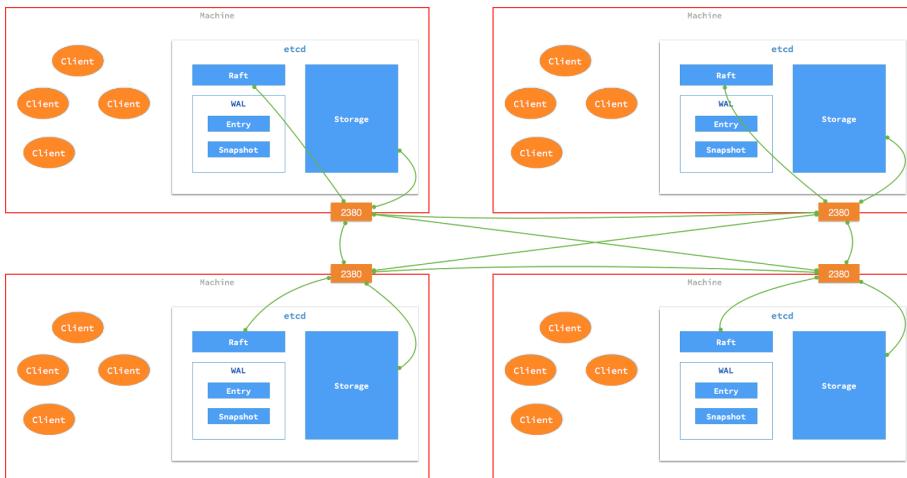
在默认设定下，etcd 通过主机的 2379 端口向 Client 提供服务。如下图：



每个主机上的应用程序都可以通过主机的 2379 以 HTTP + JSON 的方式向 etcd 读写数据。写入的数据会由 etcd 同步到集群的其它节点中。

4.2.2. Peers

在默认设定下，etcd 通过主机的 2380 端口在各个节点中同步 raft 状态及数据。



5. 创建

从方法上来划分，创建 etcd 集群的方式分为两种：Static（通过制定 peers 的 IP 和端口创建）与 Discovery（通过一个发现服务创建）。

Static 方式需要预先知道集群所有节点的 IP，所以适合小规模的集群或者搭建一个临时的开发与测试环境。

Discovery 方式不需要预先了解其他节点的 IP。启动时 etcd 通过访问一个 Discovery URL 来注册自己并获取其他节点的信息。这种方式通常适合将 etcd 部署在某个云服务平台或是一个 DHCP 环境中。其中 Discovery 服务可以使用 CoreOS 提供的一个公共地址 <https://discovery.etcd.io/new> 来申请一个 token，或者自己搭建一个这样的服务并设定一个 token。出于安全的考虑，这个 token 应该只在集群初始引导时短暂存在，因为集群建立之后将不再需要这一地址，而集群中节点的变更可以通过 etcd 运行时重构的能力来进行配置。

6. 运行

下面我们尝试使用 etcd 0.5.0 以 discovery 方式创建一个 CoreOS 集群。当然由于 etcd 0.5.0 尚未正式发布，所以我们目前还无法从官方渠道获得打包 etcd 0.5.0 的 CoreOS 镜像，但是我们可以修改引导文件，在 CoreOS 启动时将 etcd 0.5.0 下载至系统里进行配置并启动。

CoreOS 官方提供了一个使用 vagrant + virtualbox 项目，供用户在本地电脑中创建一个微型 CoreOS 集群。我们可以在这个项目的基础上进行修改来实现我们的需求。

6.1. Clone 项目到本地

```
git clone
https://github.com/coreos/coreos-vagrant.git
cd coreos-vagrant
cp config.rb.sample config.rb
cp user-data.sample user-data
```

6.2. 通过 CoreOS 提供的公共 discovery 服务申请 token

```
curl https://discovery.etcd.io/new?size=3
https://discovery.etcd.io/780456e1317eb2db312b62ba1cb9a4f7
```

size = 3 表示这个集群节点总数为 3 个。

6.3. 修改 config.rb 文件

```
# Size of the CoreOS cluster created by Vagrant
$num_instances=3
```

将启动的 CoreOS 实例数量定为 3 个

6.4. 修改 user-data 文件

6.4.1. 修改 etcd 参数：

```
etcd:
  discovery: https://discovery.etcd.io/780456e1317eb2db312b62ba1cb9a4f7
  advertise-client-urls: http://$public_ipv4:2379
  initial-advertise-peer-urls: http://$public_ipv4:2380
  listen-client-urls: http://$public_ipv4:2379
  listen-peer-urls: http://$public_ipv4:2380
```

6.4.2. 修改 etcd 服务内容

```
- name: etcd.service
  command: start
  content: |
    [Unit]
    After=network-online.target
    Requires=network-online.target

    [Service]
    ExecStartPre=/usr/bin/wget -N -P /opt/bin https://github.com/coreos/etcd/releases/
download/v0.5.0-alpha.3/etcd-v0.5.0-alpha.3-linux-amd64.tar.gz
    ExecStartPre=/usr/bin/tar -C /opt/bin -xvf /opt/bin/etcd-v0.5.0-alpha.3-linux-amd64.tar.gz
    ExecStartPre=/usr/bin/chmod +x /opt/bin/etcd-v0.5.0-alpha.3-linux-amd64/etcd
    ExecStart=/opt/bin/etcd-v0.5.0-alpha.3-linux-amd64/etcd

    [Install]
    WantedBy=multi-user.target
```

修改的部分让我们重新定制了 etcd 服务的内容。在系统启动时，先将 etcd 0.5.0 的打包文件下载至指定目录并在稍后将其启动。

6.4.3. 使用 vagrant 启动集群

```
vagrant up
```

稍后 vagrant 将帮助我们在 VirtualBox 中创建三个 CoreOS 实例。

6.4.4. 登录到 CoreOS

登录到其中的一个节点

```
vagrant ssh core-01
```

查看一下 etcd.service 的状态，输入：

```
systemctl status etcd.service
```

不出意外的话，可以看到其状态为 active (running)

```
etcd.service
  Loaded: loaded (/etc/systemd/system/etcd.service; disabled)
  Drop-In: /run/systemd/system/etcd.service.d
            └─20-cloudinit.conf
  Active: active (running) since Sun 2014-11-16 13:10:59 UTC; 12min ago
    Process: 894 ExecStartPre=/usr/bin/chmod +x /opt/bin/etcd-v0.5.0-alpha.3-
linux-amd64/etcd (code=exited, status=0/SUCCESS)
    Process: 890 ExecStartPre=/usr/bin/tar -C /opt/bin -xvf /opt/bin/etcd-
v0.5.0-alpha.3-linux-amd64.tar.gz (code=exited, status=0/SUCCESS)
    Process: 857 ExecStartPre=/usr/bin/wget -N -P /opt/bin
https://github.com/coreos/etcd/releases/download/v0.5.0-alpha.3/etcd-v0.5.0-
-alpha.3-linux-amd64.tar.gz (code=exited, status=0/SUCCESS)
   Main PID: 896 (etcd)
```

```
CGroup: /system.slice/etc.service
└─896 /opt/bin/etc-v0.5.0-alpha.3-linux-amd64/etc
```

6.4.5. 通过 etcdctl 查询集群状态

etcdctl 可以帮助我们查询一下集群中节点信息，输入：

```
/opt/bin/etc-v0.5.0-alpha.3-linux-amd64/etcctl --peers 172.17.8.101:2379 member list
```

启动参数 --peers 172.17.8.101:2379 表示通过本节点的 2379 端口访问 etcd 接口，用户可以根据自己的实际情况对 IP 地址进行修正。

正常情况下可看到如下输出：

```
b4f4d25ed56d7a44: name=b74c24773df147e1be8e1e35defaad38 peerURLs=
http://172.17.8.101:2380 clientURLs=http://172.17.8.101:2379
b7404cc414e7affa: name=001db0fc02184af5b293e2cb21c86a11 peerURLs=
http://172.17.8.102:2380 clientURLs=http://172.17.8.102:2379
f609956c55a6809e: name=2840dc331d224360a097b781c876c9e5 peerURLs=
http://172.17.8.103:2380 clientURLs=http://172.17.8.103:2379
```

这样，我们就在本地创建了一个基于 etcd 0.5.0 的 CoreOS 集群。

7. 预告

作为 CoreOS 及管理工具介绍的第三部分，笔者将向大家介绍 CoreOS 集群的重要管理工具 fleet，通过 fleet 用户可以在 CoreOS 实现简单的 Orchestration 功能，敬请期待。

感谢[郭董](#)对本文的策划和审校。

给InfoQ中文站投稿或者参与内容翻译工作，请邮件至editors@cn.infoq.com。也欢迎大家通过新浪微博（[@InfoQ](#)）或者腾讯微博（[@InfoQ](#)）关注我们，并与我们的编辑和其他读者朋友交流。

【QCon北京2016】近日，极客邦科技经常被热门大片儿《QCon的后裔》攻陷，不少Q迷们都直呼膜拜美国硅谷 Airbnb 美女工程师朱赟的“撩汉”技能，女神已放话要美美的来QCon啦！男神阿里巴巴资深总监庄卓然也将在此邂逅。4月21~23日，3天的约会，等你来见证。再不行动，神都帮不了你了。[约约约](#)。

- 领域
- [架构 & 设计](#)
- [语言 & 开发](#)
- 专栏
- [CoreOS实战](#)
- 架构
- [coreos](#)
- [Docker](#)
- [分布式编程](#)
- [分布式开发](#)

您好，朋友！

您需要[注册一个InfoQ账号](#)或者[登录](#)才能进行评论。在您完成注册后还需要进行一些设置。

获得来自InfoQ的更多体验。

告诉我们您的想法

信息

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

当有人回复此评论时请E-mail通知我

社区评论 [Watch Thread](#)

[拼写错误 by Wo HanHor Posted 2014年11月21日 09:57](#)

[Re: 拼写错误 by Wo HanHor Posted 2014年11月21日 10:24](#)

[Re: 拼写错误 by Guo Gary Posted 2014年11月21日 04:26](#)

[Re: 拼写错误 by yong jie Posted 2014年11月24日 10:34](#)

[这个赞 by ma lichao Posted 2014年11月21日 03:58](#)

[赞 by juq kai Posted 2014年12月2日 02:01](#)

[类似zk by 刑天 Posted 2015年2月11日 04:33](#)

[etcd入门 by Mar Xsank Posted 2015年8月17日 11:15](#)

拼写错误 2014年11月21日 09:57 by "Wo HanHor"

`cp user-data.sample user-data.sample` 这个地方目标文件后面应该是没有`.sample`

- [回复](#)
- [回到顶部](#)

Re: 拼写错误 2014年11月21日 10:24 by "Wo HanHor"

还有，`修改 etcd 服务内容` 下面的配置格式乱掉了，容易给别人造成误导

- [回复](#)
- [回到顶部](#)

这个赞 2014年11月21日 03:58 by "ma lichao"

这个赞

- [回复](#)
- [回到顶部](#)

Re: 拼写错误 2014年11月21日 04:26 by "Guo Gary"

已经修正。

- [回复](#)
- [回到顶部](#)

Re: 拼写错误 2014年11月24日 10:34 by "yong jie"

请教下，在“3.4.1. 复制”，当leader 将条目复制给所有的 follower，并且当大多数节点记录此条目之后并ack给 leader。leader 节点认定此条目有效，将此条目设定为已提交并存储于本地磁盘。之后在下一个 heartbeat，leader 通知所有 follower 提交这一日志条目并存储于各自的磁盘内。

如果在下一个heartbeat时，leader 挂了，那么通知所有follower提交这条日志条目的消息就不会被发出，那么之后这条日志条目在下一个candidate选举以后，会被commit吗？（大多数follower都有此条目）。

另外，对于客户端来说，没有接收到前leader的response，是需要重试来检测此条请求是否成功被同步到所有 cluster上吗？如果这样，leader在什么时候返回给client比较合适，是在自己commit之后还是在commit并且下一个heartbeat给所有其他follower之后呢？

谢谢。

- [回复](#)
- [回到顶部](#)

赞 2014年12月2日 02:01 by "juq kai"

写得很详细，谢谢

- [回复](#)
- [回到顶部](#)

类似zk 2015年2月11日 04:33 by "刑天"

看了一下etcd的接口，和zookeeper的接口相似，但zookeeper有成熟的客户端和稳定版本，etcd就得需要时间检验了

- [回复](#)
- [回到顶部](#)

etcd入门 2015年8月17日 11:15 by "Mar Xsank"

不错的文章

- [回复](#)
- [回到顶部](#)

[关闭](#)

by

发布于

- [查看](#)
- [回复](#)
- [回到顶部](#)

[关闭](#)

主题

您的回复

[引用原消息](#)

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

当有人回复此评论时请E-mail通知我

[发送信息](#) [取消](#)

[关闭](#)

主题

您的回复

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

当有人回复此评论时请E-mail通知我

[取消](#)

[关闭](#)

[OK](#)

相关内容



- [CoreOS综合案例之分布式服务的监控](#) 2016年1月11日



- [大型分布式系统设计的一些黄金原则和实例](#) 2016年2月1日



- [分布式监控系统的设计与实现](#) 2015年11月30日

- [设计全球级的分布式、任务关键型应用——从实际项目中得来的教训（上）](#) 2015年11月27日



- [1号店11.11：分布式搜索引擎的架构实践](#) 2015年11月12日

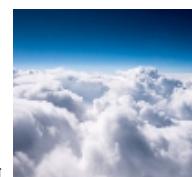


- [解读2015之容器篇：扩张与进化](#) 2016年1月14日



- [如何在云平台构建大规模分布式系统](#) 2016年4月18日

- [Docker推出Mac、Windows的Beta版](#) 2016年3月31日



- [Facebook的分布式图查询引擎：Dragon](#) 2016年3月28日



- [Docker三年回顾：梦想依在，人生正当年](#) 2016年3月22日



- [集群调度框架的架构演讲之路](#) 2016年3月17日

赞助商内容



- [GMTC全球移动技术大会, 6折优惠, 最后一周!](#)



United Stack 有云

- [通过demo学习OpenStack开发所需的基础知识 — 软件包管理](#)

相关内容

- 肖德时：[如何用Docker来管理微服务架构的乱网](#) 2016年3月15日



- 蚂蚁金服金融云PaaS docker实践 2016年3月15日



- 王振威：[Docker的坑与编排系统](#) 2016年3月14日



- 为什么我要选择使用Yarn来做Docker的调度引擎 2016年2月26日



- 百度超大规模分布式安全系统实践 2016年2月26日



- 洪强宁：[宜信的PaaS平台基于Calico的容器网络实践](#) 2016年2月25日



- Apache Ignite(四)：[基于Ignite的分布式ID生成器](#) 2016年2月15日



- [微众银行基于自主可控技术的分布式架构实践](#) 2016年2月9日



- [10个精选的容器应用案例](#) 2016年2月4日



- [Docker架构下私有云的机遇与挑战](#) 2016年2月2日



- [高性能分布式PaaS解决方案](#) 2016年1月24日

赞助商内容



全球软件开发大会

北京·国际会议中心/2016年4月21-23日

[北京站]

精彩多多 尽在 QCon!

[豆瓣百万级指标监控实践](#)

QCon
全球软件开发大会

[支撑滴滴高歌猛进的管理基石](#)

相关内容

- [经验分享：大众点评私有云平台之Docker实践](#) 2016年1月22日



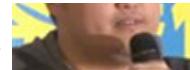
- [使用Apache Mesos伸缩Docker容器](#) 2016年1月22日



- [Docker与UCloud DataBase的融合实践](#) 2016年1月21日



- [Coding 容器化实践分享：Docker 理念解析与技术填坑](#) 2016年1月20日



- [为什么 Kubernetes 不使用 libnetwork](#) 2016年1月15日



- [解读2015之云计算篇：打磨产品服务，迎接市场变局](#) 2016年1月15日



- [Docker Swarm 与 Apache Mesos 的区别](#) 2015年12月28日



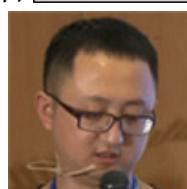
- [详解当当网的分布式作业框架 elastic-job](#) 2015年12月25日



- [Mesos 在去哪儿网的实践之路](#) 2015年12月14日



- [中国顶尖技术团队访谈录·第四季](#) 2015年12月8日



- [Docker 在汽车之家的应用实践](#) 2015年11月26日

赞助商链接

InfoQ每周精要

通过个性化定制的新闻邮件、RSS Feeds和InfoQ业界邮件通知，保持您对感兴趣的社区内容的时刻关注。



语言 & 开发

[专访包建强：为什么我说Android插件化从入门到放弃？](#)

[为什么我不再使用MVC框架](#)

[以持续集成工具实现DevOps之禅](#)

架构 & 设计

[专访包建强：为什么我说Android插件化从入门到放弃？](#)

[为什么我不再使用MVC框架](#)

[以持续集成工具实现DevOps之禅](#)

文化 & 方法

[John Willis谈软件行业的职业倦怠](#)

[中国顶尖技术团队访谈录·第五季](#)

[Hadoop Summit 2016会场回顾（二）](#)

数据科学

[Hadoop Summit 2016会场回顾（二）](#)

[IAP:HTTP的替代者、更快、更丰富](#)

[Facebook想用机器人取代App](#)

[中国顶尖技术团队访谈录·第五季](#)

[微软推出Azure Functions预览版、涉足无服务器应用领域](#)

[JetBrains发布了IntelliJ IDEA 2016.1](#)

- [首页](#)
- [全部话题](#)
- [QCon全球软件开发大会](#)
- [关于我们](#)
- [投稿](#)
- [创建账号](#)
- [登录](#)

- 全球QCon
- [圣保罗 2016年3月28日-4月1日](#)
- [北京 2016年4月21-23日](#)
- [纽约 2016年6月13日-6月17日](#)
- [里约热内卢 2016年10月3-7日](#)
- [上海 2016年10月20-22日](#)
- [旧金山 2016年11月7-11日](#)

InfoQ每周精要

通过个性化定制的新闻邮件、RSS Feeds和InfoQ业界邮件通知，保持您对感兴趣的社区内容的时刻关注。

[点击这里
查看样刊](#)



- [属于您的个性化RSS](#)
- [InfoQ官方微博](#)
- [InfoQ官方微信](#)
- [社区新闻和热点](#)

特别专题

- [活动大本营](#)
- [月刊：《架构师》](#)
- [AWS专区](#)
- [百度技术沙龙专区](#)
- [腾讯云专区](#)
- [七牛专区](#)
- [EGO超级极客邦](#)
- [StuQ提升你技能](#)
- [信息无障碍参考文档](#)

定制您感兴趣的技术领域

- 语言 & 开发
- 架构 & 设计
- 数据科学
- 文化 & 方法
- DevOps

这会影响您在主页和RSS订阅中看到的内容。点击“偏好设置”可选择更多精彩定制内容。

InfoQ.com
及所有内
容，版
权所
有 ©
2006-2016
C4Media
Inc.
InfoQ.com
服务器由

[Contegix](#)提
供，我们最
信赖的ISP
伙伴。
北京创新
网媒广告
有限公司
京ICP备
09022563
号-7 [隐私
政策](#)

提供反馈

feedback@cn.infoq.com

错误报告

bugs@cn.infoq.com

商务合作

comsales@cn.infoq.com

内容合作

comeditors@cn.infoq.com

Marketing

commarketing@infoq.com

[BT](#)