滴滴基于Go实现的大规模服务发现系统分享

滴滴/基础架构部 张慧

自我介绍

16 年之前，在腾讯用 C 写游戏，折腾过协程；

16 年之后，在创业公司用 Go 写物联网云平台，用 Go 实现了一系列基础组件，包括 MQTT 和 CoAP 这两个协议；

19 年 7 月，加入滴滴，负责服务发现和 RPC 框架的开发和维护，主导了滴滴服务发现 v4 版本的开发。

目录

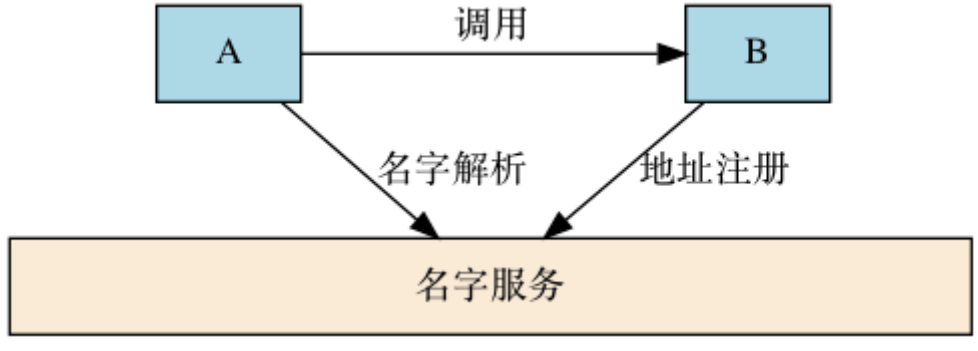
- 服务发现概念
- 滴滴服务发现系统的主要功能及落地挑战
- 滴滴服务发现系统的架构设计及特色功能
- 稳定性保障经验
- 性能和未来规划

什么是服务发现

命名服务支持的主要操作是名字解析，即根据一个给定的名字查询相应的属性。

-------- 《分布式系统概念与设计》

更通俗地讲，根据一个名字获取一组 IP 或 IP:PORT，如 DNS 便是当前应用最广泛、规模最庞大的服务发现系统。



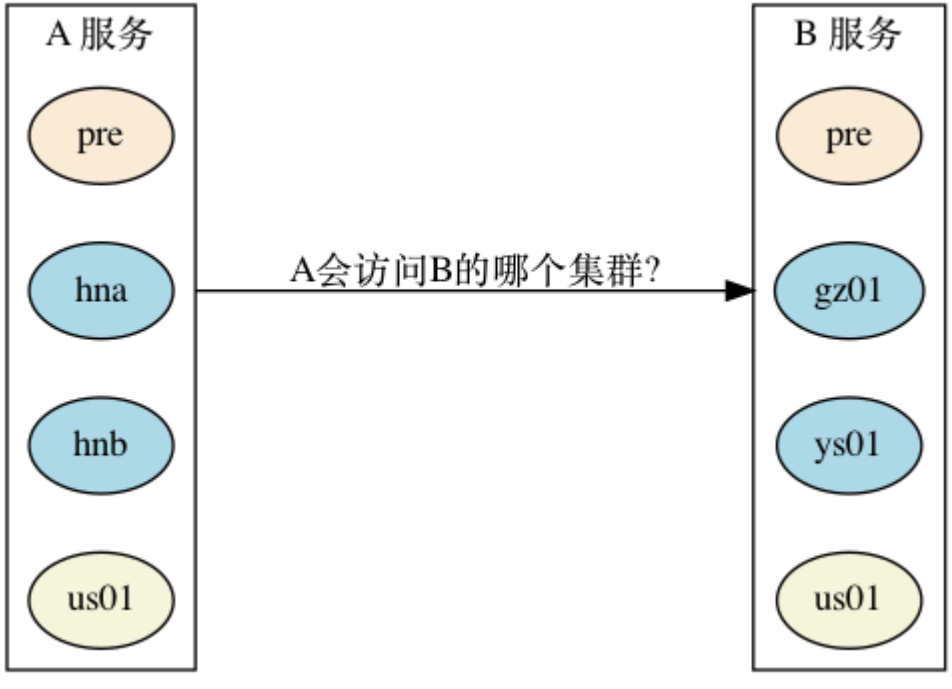
滴滴服务发现系统简介

DISF(didi service found/framework)

- 服务数量：3K+
- 覆盖率：40.2%
- 集群数量：2W+
- 地址数量：12W+
- 支持语言：go, php, java, c++
- 时效性：国内 < 5s，国际 < 10s

滴滴服务发现系统的核心功能

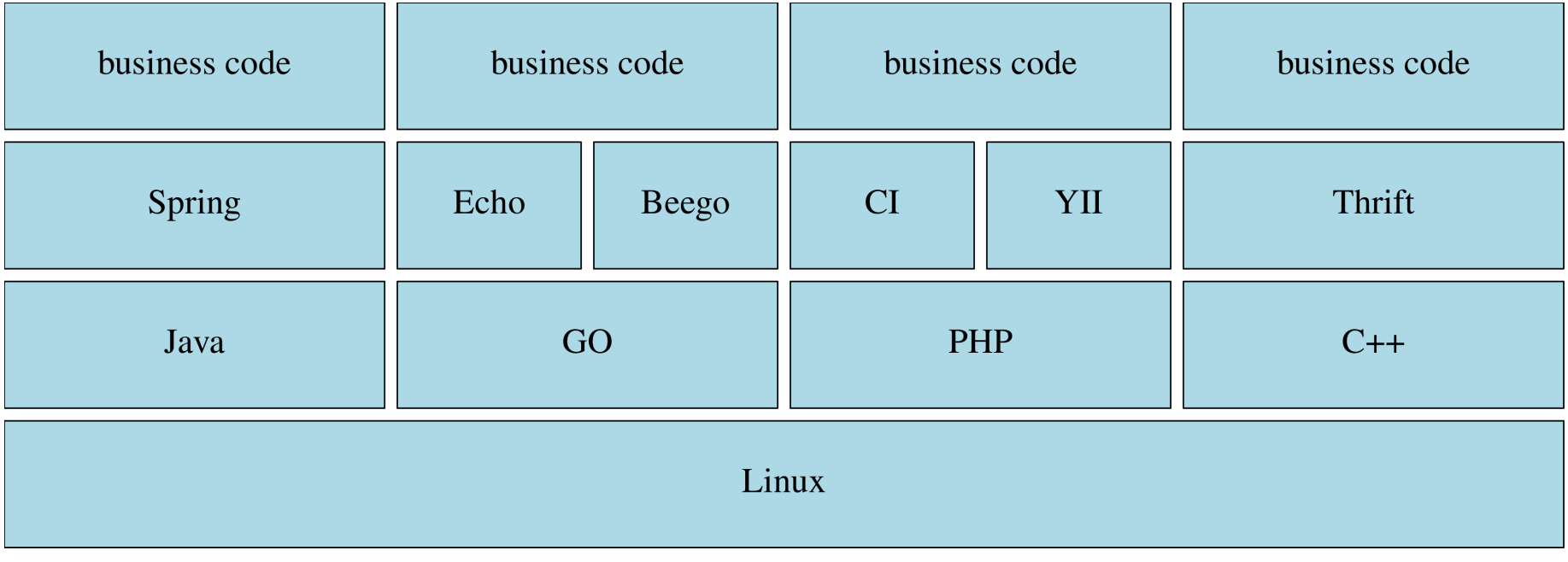
- 名字寻址：解耦上游访问下游时对 IP 的依赖，以此为基础实现快速扩缩容、故障摘除、优雅发布等功能赋能业务
- 路由管理：路由关系平台化，在此基础上提供多活、名字层切流等支持



滴滴服务发现系统的主要挑战

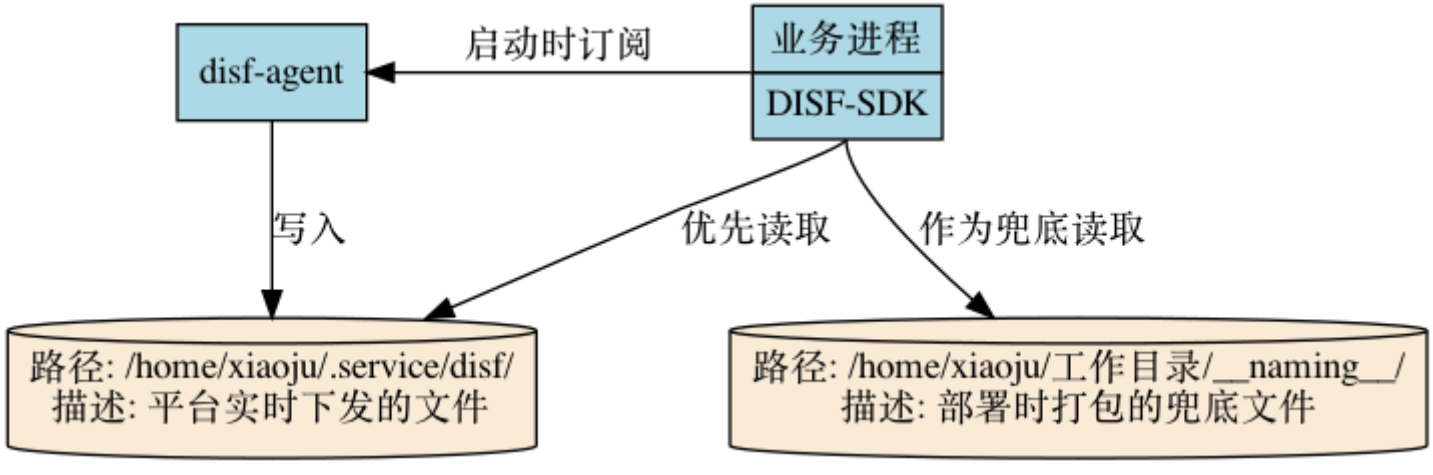
- 不同业务使用不同的语言、协议、框架
- 难以强制业务升级到统一的框架

框架语言五花八门，唯一的共同点是都跑在 linux centos 上



滴滴异构场景下的服务发现方案

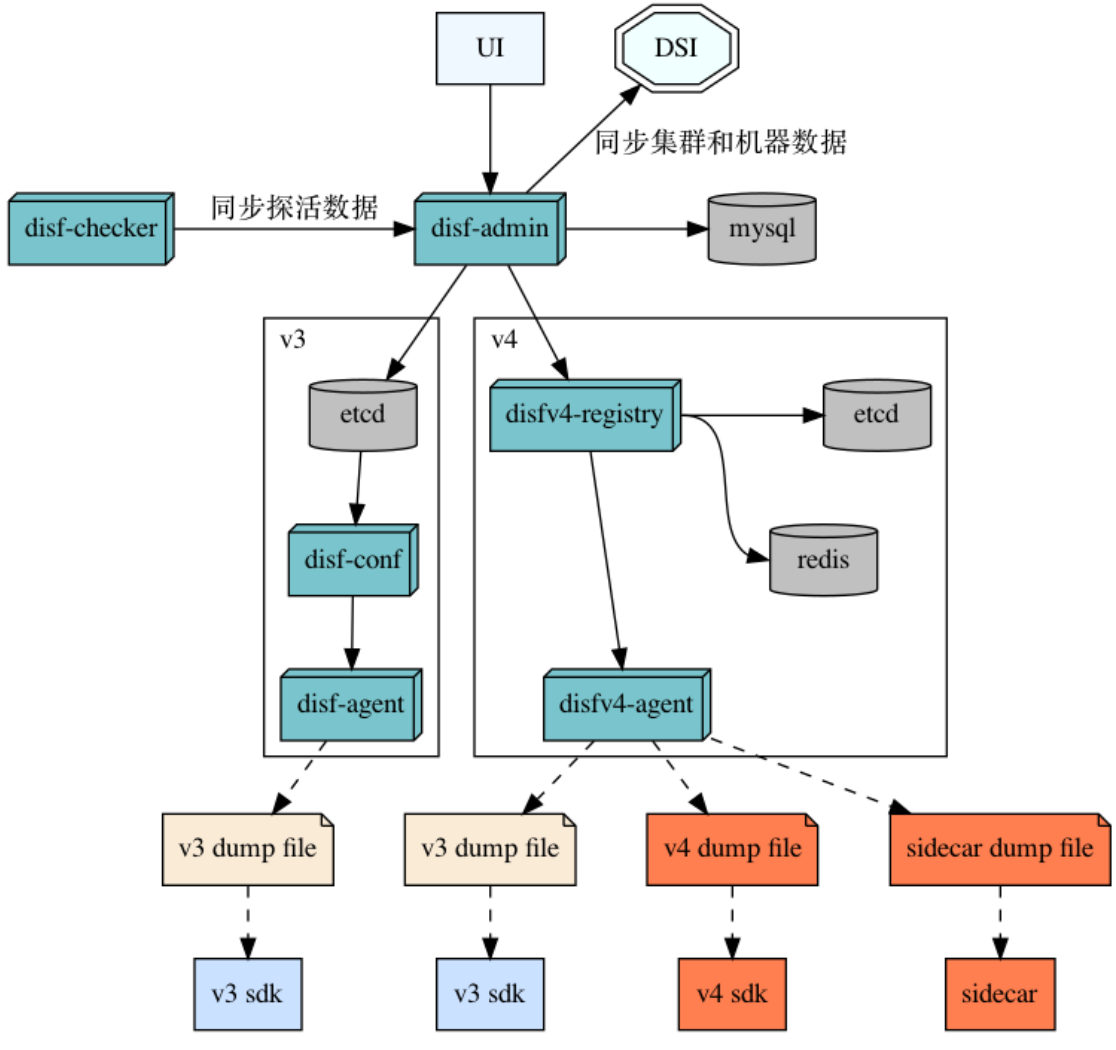
- 最小入侵性接入：不与 RPC 框架强绑定，服务提供方不需要接入任何 SDK，服务消费方提供类似 DNS 名字解析的接口获取地址
- 多语言 SDK 支持：支持 go，php，java，c++ 这四种滴滴主流开发语言
- 低成本轻量 SDK 实现：通过文件加载服务发现数据



滴滴服务发现发展历程

时间轴	版本	关键事项
2017.03	v1	基于开源的 consul，快速打通服务发现基本流程
2017.06	v2	在 v1 的基础上提供集群分组、到达监控等功能
2018.05	v3	引入 db 层，引入 etcd 作为通道，轻量化 agent，支持任务灰度执行
2020.10	v4	引入路由规则匹配机制，实现通用的支持灰度的注册中心

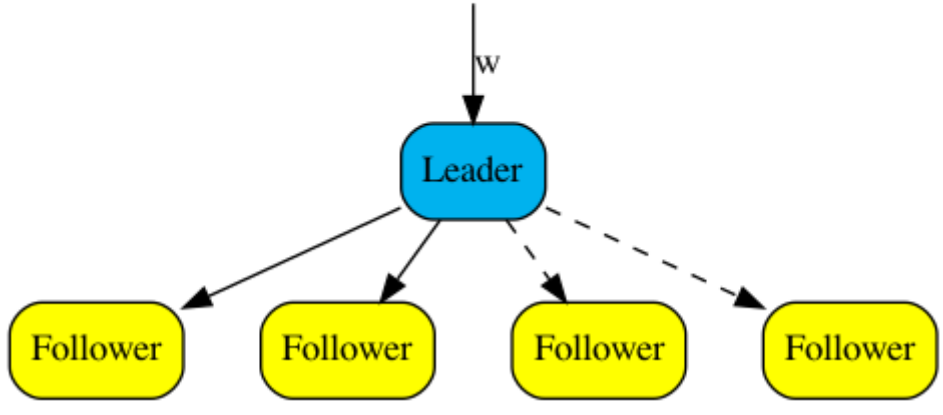
最新架构设计


UI
DSI
同步集群和机器数据
disf-checker
同步探活数据
disf-admin
mysql
v3
v4
etcd
disfv4-registry
etcd
disf-conf
redis
disf-agent
disfv4-agent
v3 dump file
v3 dump file
v4 dump file
sidecar dump file
v3 sdk
v3 sdk
v4 sdk
sidecar

agent 为什么不直连 etcd？

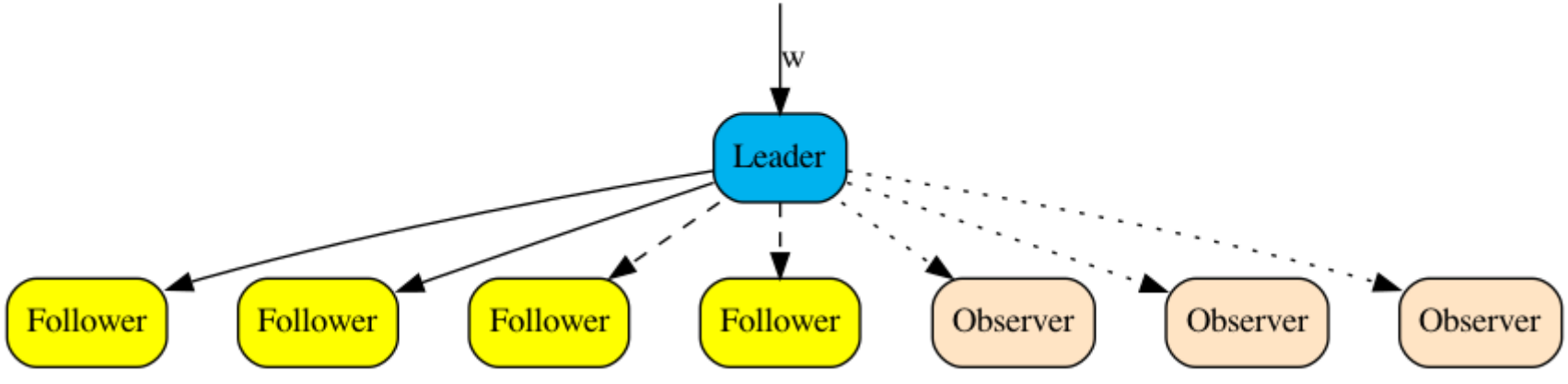
单个 etcd 节点能承载的订阅数量有限，增加 etcd 节点的数量，又会影响写入和选主

raft 日志复制原理: 半数以上节点写入成功才算成功



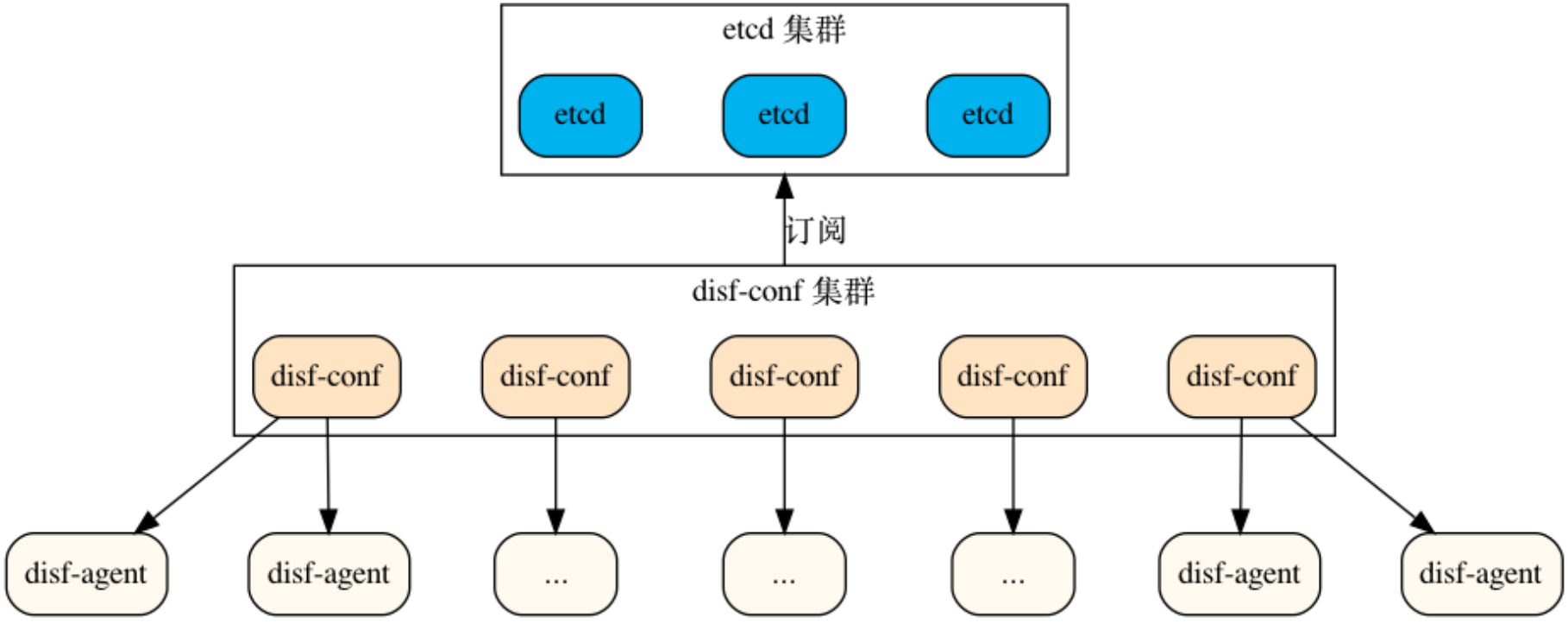
agent 为什么不直连 etcd？

增加一种角色 observer，只同步数据，不产参与选举



agent 为什么不直连 etcd？

disf-conf/disfv4-registry 承担了 observer 的角色，并扩展了一些自己的职责



路由机制

v4 基于规则匹配的路由模型

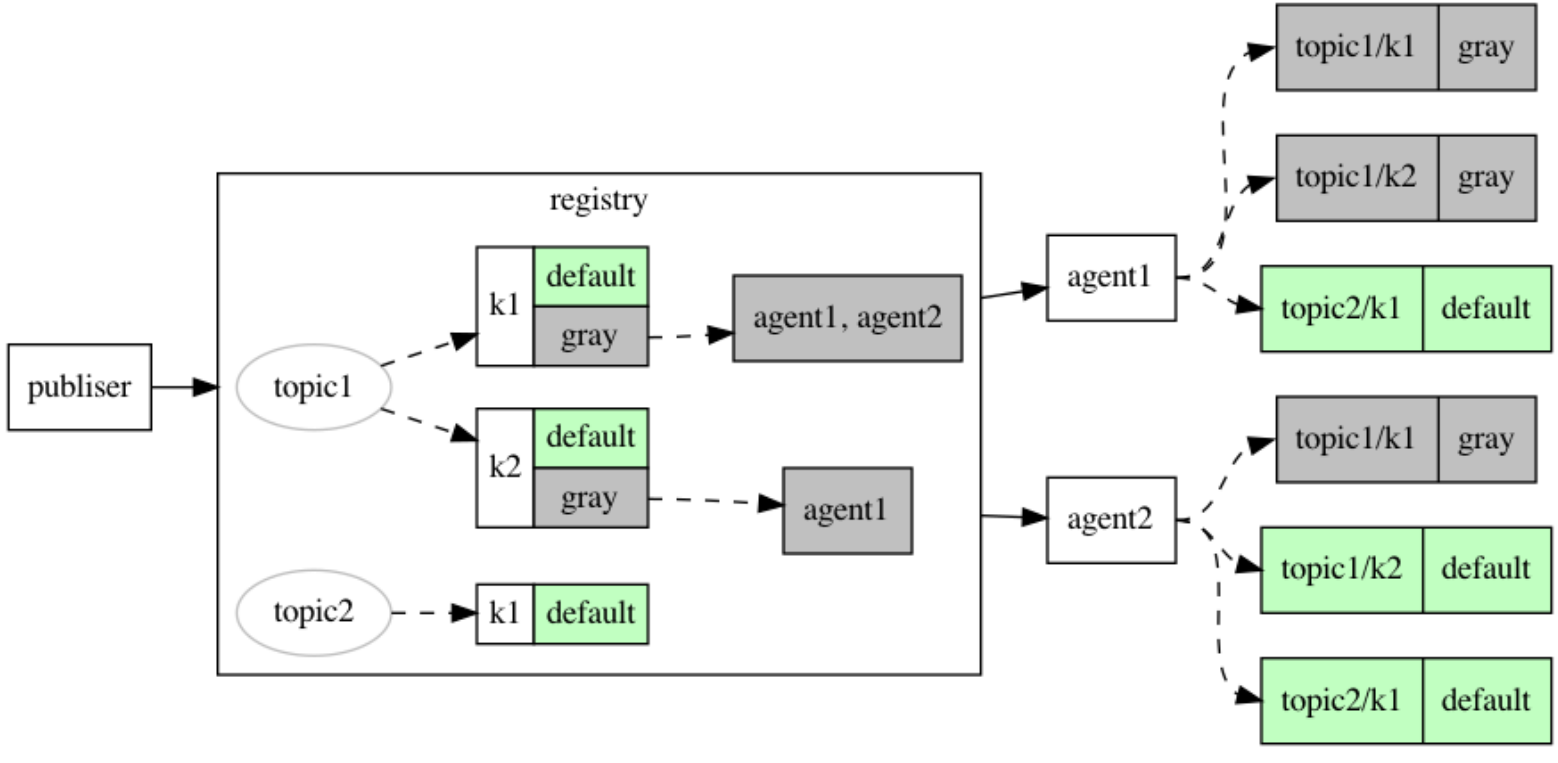
```
{
        "RouteRules": [
                {
                        "MatchRules": [
                                {
                                        "Not": false,
                                        "Op": "regexp",
                                        "Key": "Consumer-Cluster",
                                        "Value": "hn[a|b|c].*"
                                }
                        ],
                        "Destinations": [
                                {
                                        "Cluster": "hna",
                                        "Percent": 0.9
                                },
                                {
                                        "Cluster": "hna-v",
                                        "Percent": 0.1
                                }
                        ]
                }
        ]
}
```

配置了该规则的服务，如果上游服务所在的集群名可以被 ”hn[a|b|c].*” 正则匹配，则有 90% 的几率将请求转发到 hna，10% 的几率将请求转发到 hna-v

灰度机制

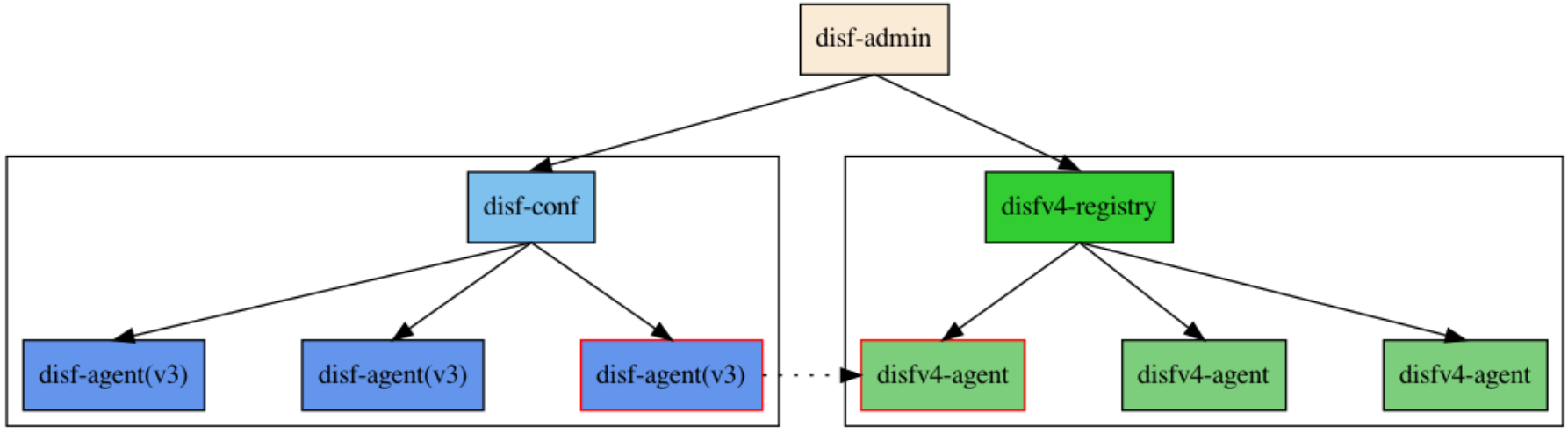
disfv4-registry 的两个特性：

- 发布订阅：订阅端可以订阅多个 topic，topic 下任何 key 的变化都会通知到订阅端
- 灰度发布：发布端设置数据时，可以指定版本号，并可通过设置版本分布调整订阅端实际接收的版本



稳定性保障经验

- 基于文件的服务发现
- 灰度升级，逐步替换
- 健全的监控报警机制
- 最小系统闭环测试
- 全链路系统测试

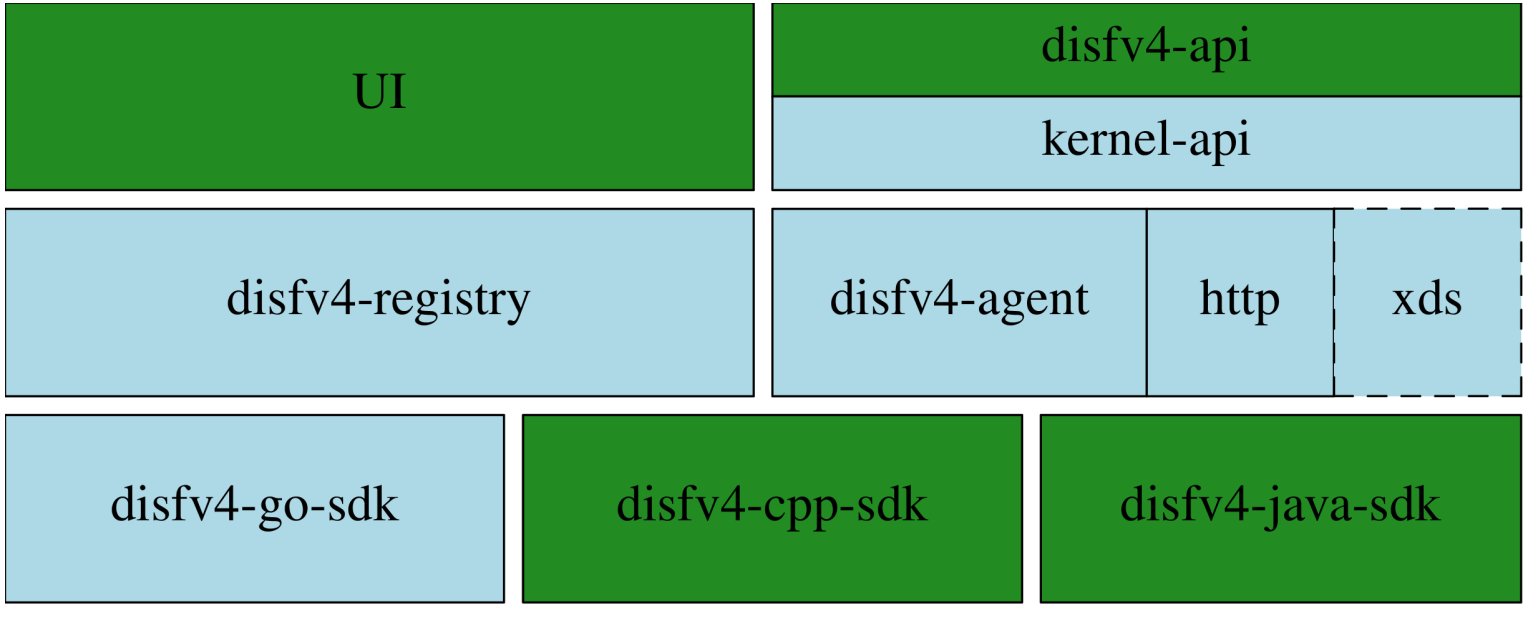


未来规划

- 开源

  我们的特色：

  - Go 生态的服务发现系统
  - 多语言异构场景
  - 强大灵活的路由
  - 支持灰度变更



致谢

如果你对服务治理感兴趣，欢迎合作交流。

iron

上海 虹口

扫一扫上面的二维码图案，加我微信