Service Mesh Meetup #3 深圳站

SOFA MESH 的通用协议扩展

邵俊雄（熊啸）

2018.08.25

ServiceMesher | IT大咖说 知识共享平台

蚂蚁金服 ANT FINANCIAL SOFASTACK

AGENDA

- SOFA MESH 介绍
- SERVICE MESH 落地的问题
- SOFA MESH 的通用落地方案
- DNS 服务寻址方案
- X-PROTOCOL 通用协议
- 问答

SOFA MESH

- 从 ISTIO 克隆并保持同步更新
- 使用 SOFA-MOSN 代替 ENVOY 作为数据平面
  - 打包
  - 安装
  - 部署
  - 测试
- 支持主流的微服务框架
  - SOFA
  - HSF
  - DUBBO
  - SPRING CLOUD
  - …
- 控制平面创新的地方
  - MESH OPERATOR
  - RPC SERVICE CONTROLLER
  - …

SERVICE MESH 落地中的问题

常见的 MESH 落地方案

自己做控制平面

在非 KUBERNETES 环境部署 ISTIO

把 KUBERNETES 当作一个更好用的虚拟化方案

KUBERNETES NATIVE 化微服务

PLATFORM ADAPTER

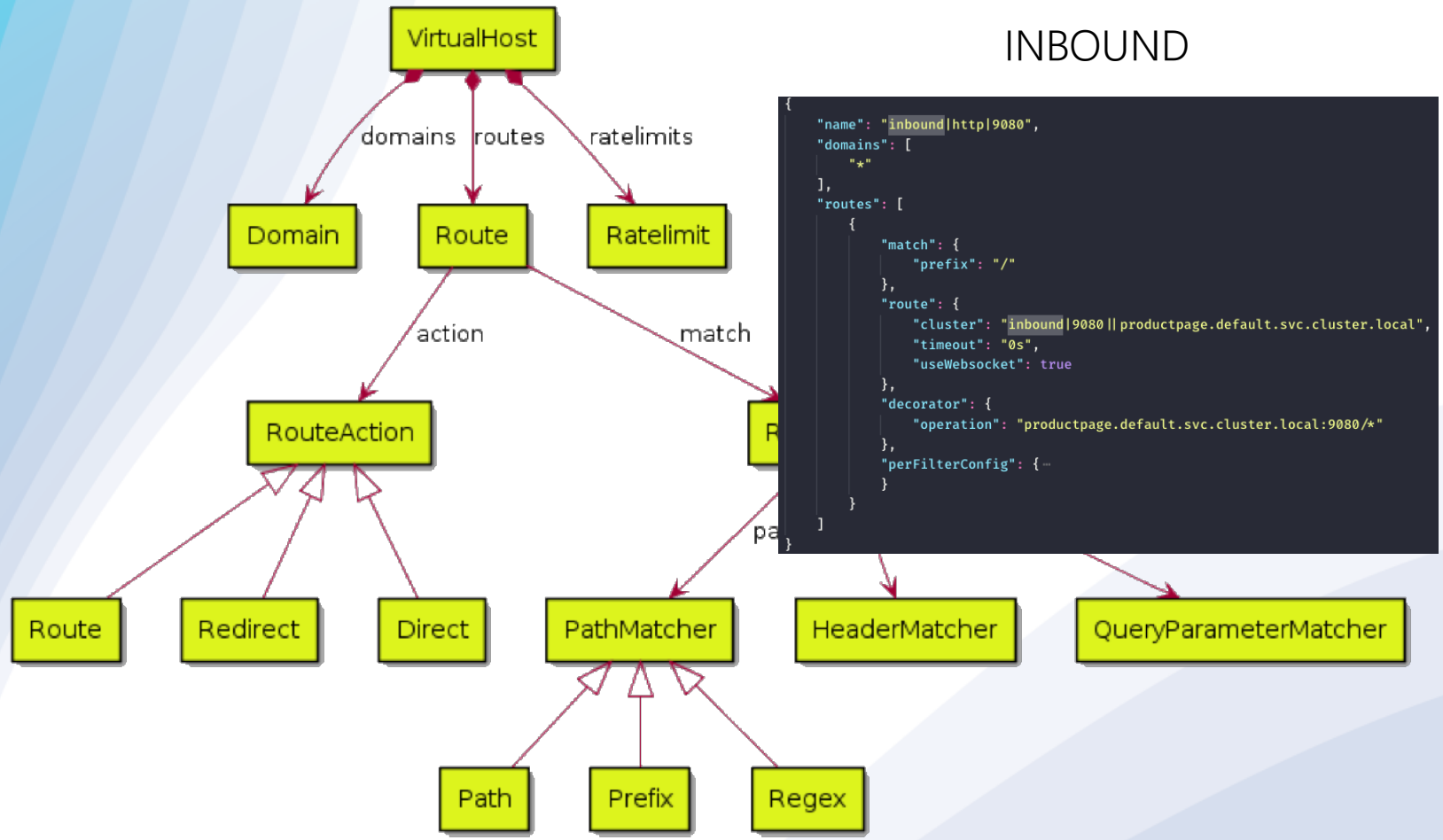
KUBERNETES NATIVE 微服务

- 使用 Kubernetes 作为注册中心
  - Service
  - Endpoint
  - Pod
- 使用 DNS 寻址
- 使用 iptables/ebpf 透明地路由所有网络流量
- 服务治理规则，服务，实例和配置都是 Kubernetes 资源
- 使用 Controller Pattern 通过 CRD 扩展新的能力
- …

MESH 落地碰到的问题

- 客户端服务发现与负载均衡无法与 ISTIO 一起工作
- ENVOY 不支持微服务使用的通信协议
- RPC 服务使用的接口，方法，参数语义无法匹配 ISTIO 的路由模型
- 一个应用上部署了多个 RPC 服务，每个服务有自己的版本
- …

ISTIO 控制平面路由的抽象模型



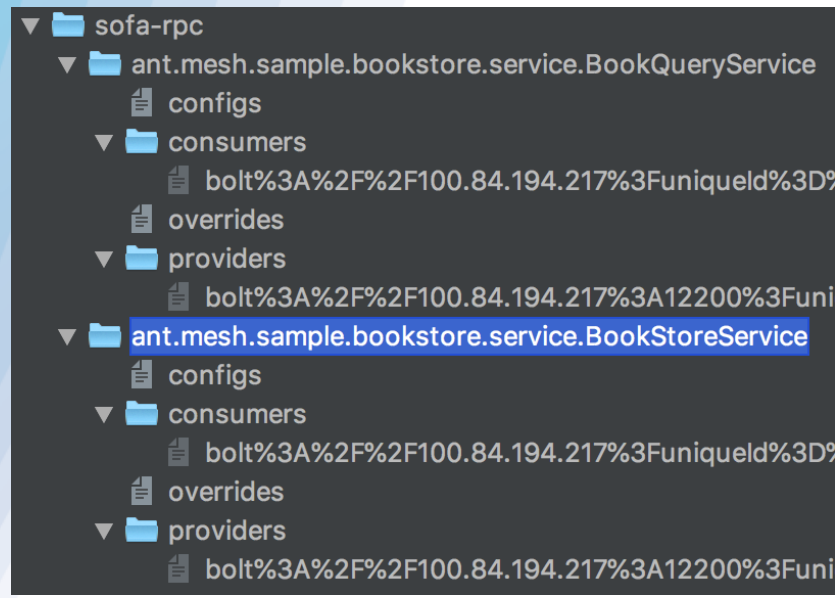
INBOUND

```
{
    "name": "inbound|http|9080",
    "domains": [
        "*"
    ],
    "routes": [
        {
            "match": {
                "prefix": "/"
            },
            "route": {
                "cluster": "inbound|9080||productpage.default.svc.cluster.local",
                "timeout": "0s",
                "useWebsocket": true
            },
            "decorator": {
                "operation": "productpage.default.svc.cluster.local:9080/*"
            },
            "perFilterConfig": {…
            }
        }
    ]
}
```

OUTBOUND

```
{
    "name": "istio-telemetry.istio-system.svc.cluster.local:15004",
    "domains": [
        "istio-telemetry.istio-system.svc.cluster.local",
        "istio-telemetry.istio-system.svc.cluster.local:15004",
        "istio-telemetry.istio-system",
        "istio-telemetry.istio-system:15004",
        "istio-telemetry.istio-system.svc.cluster",
        "istio-telemetry.istio-system.svc.cluster:15004",
        "istio-telemetry.istio-system.svc",
        "istio-telemetry.istio-system.svc:15004",
        "172.21.41.196",
        "172.21.41.196:15004"
    ],
    "routes": [
        {
            "match": {
                "prefix": "/"
            },
            "route": {
                "cluster": "outbound|15004||istio-telemetry.istio-system.svc.cluster.local",
                "timeout": "0s"
            },
            "decorator": {
                "operation": "istio-telemetry.istio-system.svc.cluster.local:15004/*"
            }
        }
    ]
}
```

SOFA 服务注册模型



Provide Service with interface service.BookQueryService through bolt:50400 protocol on app book-store:

service.BookQueryService/providers/bolt://100.84.194.217:12200?uniqueId=&version=1.0&timeout=0&delay=-1&id=bookQueryService&dynamic=true&weight=100&accepts=100000&startTime=1533736070909&appName=book-store&pid=28657&language=java&rpcVer=50400

Consume Service with interface service.BookQueryService through bolt:50400 protocol on app book-store:

service.BookQueryService/consumers/bolt://100.84.194.217?uniqueId=&version=1.0&pid=5075&timeout=3000&id=bookQueryRef&generic=false&appName=book-store&serialization=hessian2&startTime=1533783212295&pid=5075&language=java&rpcVer=50400

落地一个微服务框架需要的工作

- 部署 ZK 集群作为 RPC 框架的注册中心
- 开发 ZK Platform Adapter for DUBBO
- 开发 DUBBO 服务的 XDS 配置下发
- 开发 DUBBO 服务的路由规则 XDS 适配
- 开发 DUBBO 协议支持

SOFA MESH 的统一解决方案

- 采用 Kubernetes Native 方式落地微服务应用
- 使用 INTERFACE 作为 DNS 来寻址服务
- 开发一个通用协议处理框架
  - 避免为不同的微服务框架修改 PILOT 代码
  - 通过插件的方式按需支持新的协议
- 对应用代码无侵入性
  - 为微服务框架提供轻量化客户端

落地一个微服务框架需要的工作

- ~~部署 ZK 集群作为 RPC 框架的注册中心~~
- ~~开发 ZK Platform Adapter for DUBBO~~
- ~~开发 DUBBO 服务的 XDS 配置下发~~
- ~~开发 DUBBO 服务的路由规则 XDS 适配~~
- 开发 DUBBO 协议支持（开箱即用模式下也可以省掉）

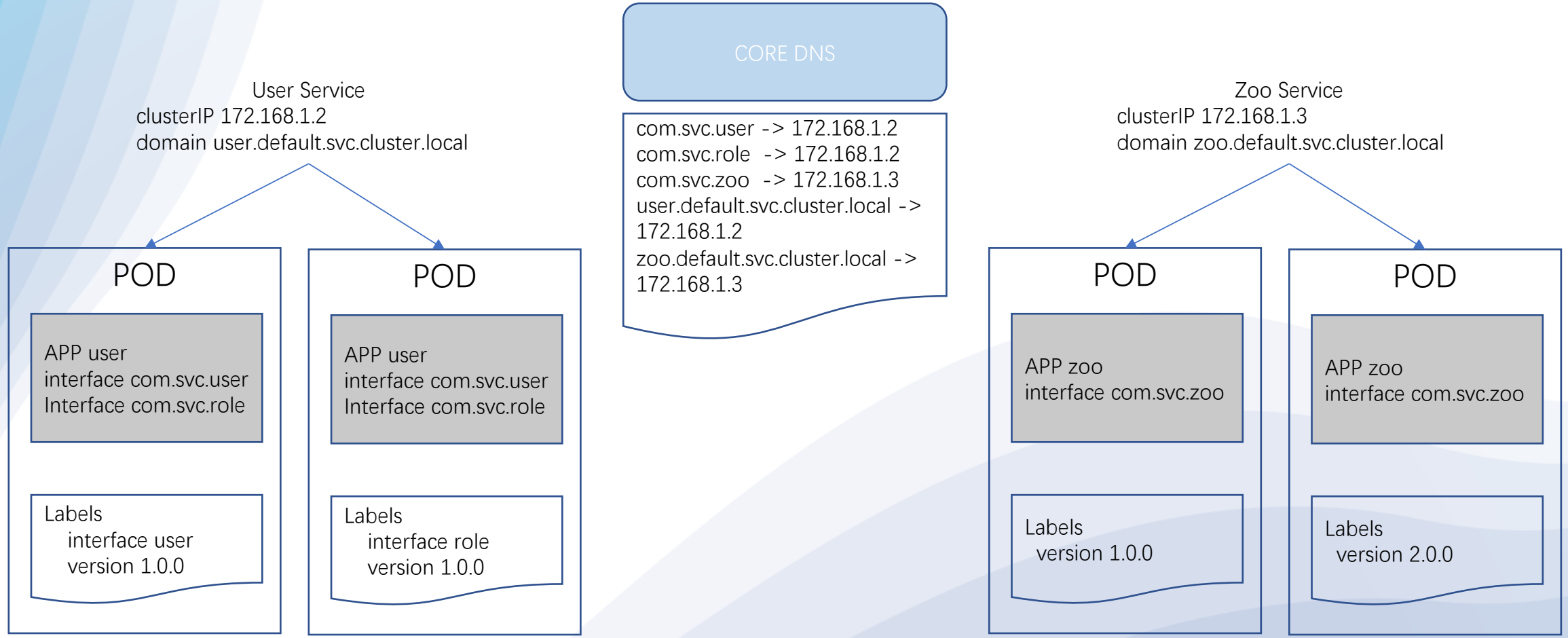
DNS 寻址

目标

bolt://com.yourcompany.youservice:12220

- 允许应用把接口当做域名来访问远端服务
- 支持在 Kubernetes DNS 之上构建更结构化的域名体系
- 支持跨集群的服务寻址
- 支持单应用多服务的部署模型

落地形态 Services, PODS & DNS

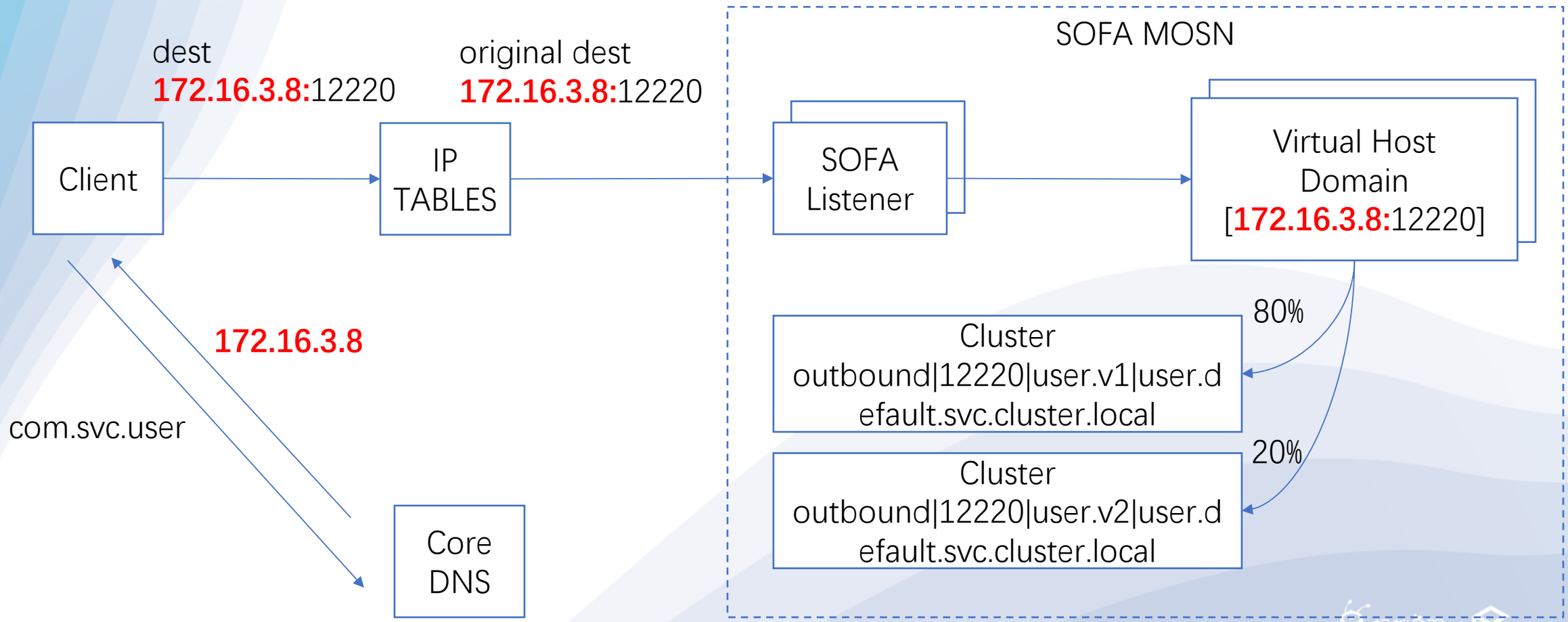


落地形态 Traffic Rules

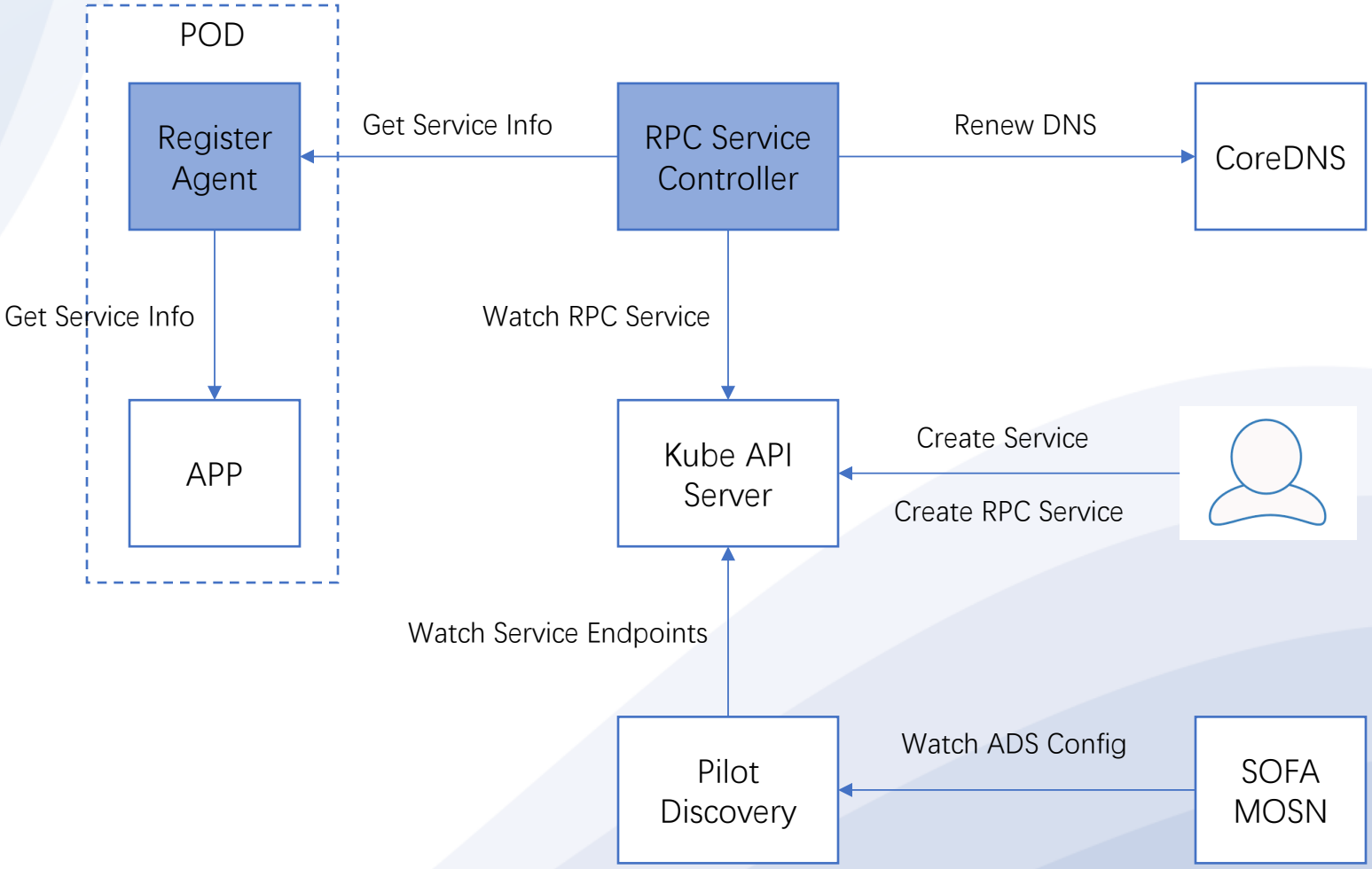
```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: user
spec:
  host: user
  trafficPolicy:
    loadBalancer:
      simple: RANDOM
  subsets:
  - name: user.v1
    labels:
      interface: user
      version: 1.0.0
  - name: user.v2
    labels:
      interface: user
      version: 2.0.0
    trafficPolicy:
      loadBalancer:
        simple: ROUND_ROBIN
```

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: user
spec:
  hosts:
  - user
  http:
  - route:
    - destination:
        host: user
        subset: user.v1
      weight: 80
    - destination:
        host: user
        subset: user.v2
      weight: 20
```

调用流程


dest
172.16.3.8:12220
original dest
172.16.3.8:12220
SOFA MOSN
Client
IP
TABLES
SOFA
Listener
Virtual Host
Domain
[172.16.3.8:12220]
80%
Cluster
outbound|12220|user.v1|user.default.svc.cluster.local
20%
Cluster
outbound|12220|user.v2|user.default.svc.cluster.local
172.16.3.8
com.svc.user
Core
DNS

DNS 寻址方案



CRD 定义

```go
type RpcService struct {
  metav1.TypeMeta   `json:",inline"`
  metav1.ObjectMeta `json:"metadata,omitempty"`

  Spec   RpcServiceSpec   `json:"spec"`
  Status RpcServiceStatus `json:"status"`
}

type RpcServiceSpec struct {
  Selector map[string]string `json:"selector"`
  DomainSuffix string `json:"domainsuffix"`
}

type RpcServiceStatus struct {
}

type RpcServiceList struct {
  metav1.TypeMeta `json:",inline"`
  metav1.ListMeta `json:"metadata"`

  Items []RpcService `json:"items"`
}
```

RPC Service 对应一个微服务，对应 RPC 框架中的接口定义
一个应用可能包含多个 RPC Service
RPC Service 关联到一个 Kubernetes 服务
RPC Service 的域名就是其接口

X-PROTOCOL 通用协议扩展

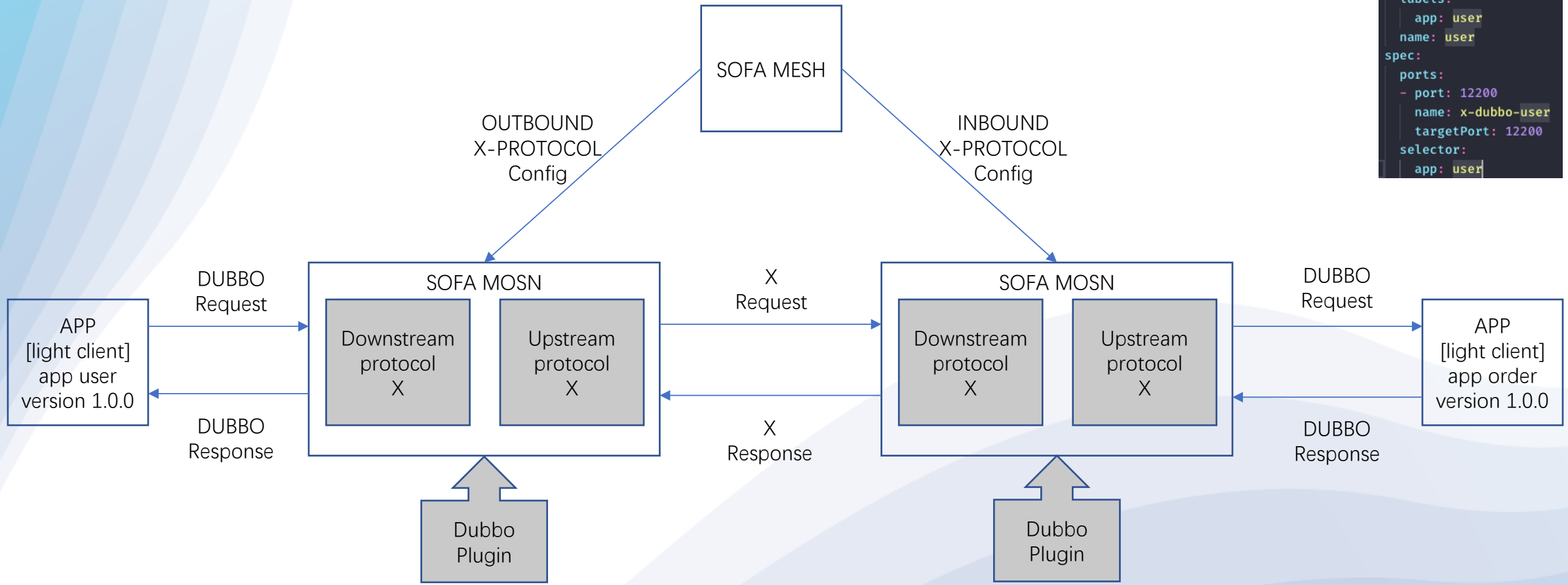
目标

- Kubernetes Native，高性能，低侵入性的通用 Mesh 落地方案
- 支持新 RPC 框架和通信协议低成本接入
- 协议扩展对 Mesh 控制平面透明化
- 允许对协议多层次，插件化的扩展

X-PROTOCOL 配置

```
type XProxy struct {
    // Supplies the x's real protocol.
    XProtocol string `protobuf:"bytes,1,opt,name=x_protocol,json=xProtocol,proto3" json:"x_protocol,omitempty"`
    // The human readable prefix to use when emitting statistics for the
    // connection manager. See the :ref:`statistics documentation <config_http_conn_man_stats>` for
    // more information.
    StatPrefix string `protobuf:"bytes,2,opt,name=stat_prefix,json=statPrefix,proto3" json:"stat_prefix,omitempty"`
    // Types that are valid to be assigned to RouteSpecifier:
    //  *XProxy_Rds
    //  *XProxy_RouteConfig
    RouteSpecifier     isXProxy_RouteSpecifier `protobuf_oneof:"route_specifier"`
    DownstreamProtocol XProxy_Protocol         `protobuf:"varint,5,opt,name=downstream_protocol,json=downstreamProtocol,proto3,enum=envoy.config.fil
    UpstreamProtocol   XProxy_Protocol         `protobuf:"varint,6,opt,name=upstream_protocol,json=upstreamProtocol,proto3,enum=envoy.config.filter.
    // A list of individual HTTP filters that make up the filter chain for
    // requests made to the connection manager. Order matters as the filters are
    // processed sequentially as request events happen.
    StreamFilters []*StreamFilter `protobuf:"bytes,7,rep,name=stream_filters,json=streamFilters" json:"stream_filters,omitempty"`
    // A list of meta used to add to http2 header
    MetasToAddToHeader []string `protobuf:"bytes,8,rep,name=metas_to_add_to_header,json=metasToAddToHeader" json:"metas_to_add_to_header,omitempty"`
    // Presence of the object defines whether the connection manager
    // emits :ref:`tracing <arch_overview_tracing>` data to the :ref:`configured tracing provider
    // <envoy_api_msg_config.trace.v2.Tracing>`.
    Tracing *XProxy_Tracing `protobuf:"bytes,9,opt,name=tracing" json:"tracing,omitempty"`
    // An optional override that the connection manager will write to the server
    // header in responses. If not set, the default is *envoy*.
    ServerName string `protobuf:"bytes,10,opt,name=server_name,json=serverName,proto3" json:"server_name,omitempty"`
    IdleTimeout *time.Duration `protobuf:"bytes,11,opt,name=idle_timeout,json=idleTimeout,stdduration" json:"idle_timeout,omitempty"`
    DrainTimeout *time.Duration `protobuf:"bytes,12,opt,name=drain_timeout,json=drainTimeout,stdduration" json:"drain_timeout,omitempty"`
    AccessLog []*envoy_config_filter_accesslog_v2.AccessLog `protobuf:"bytes,13,rep,name=access_log,json=accessLog" json:"access_log,omitempty"`
}
```

调用流程




```yaml
apiVersion: v1
kind: Service
metadata:
  annotations:
  labels:
    app: user
  name: user
spec:
  ports:
  - port: 12200
    name: x-dubbo-user
    targetPort: 12200
  selector:
    app: user
```

分层扩展接口

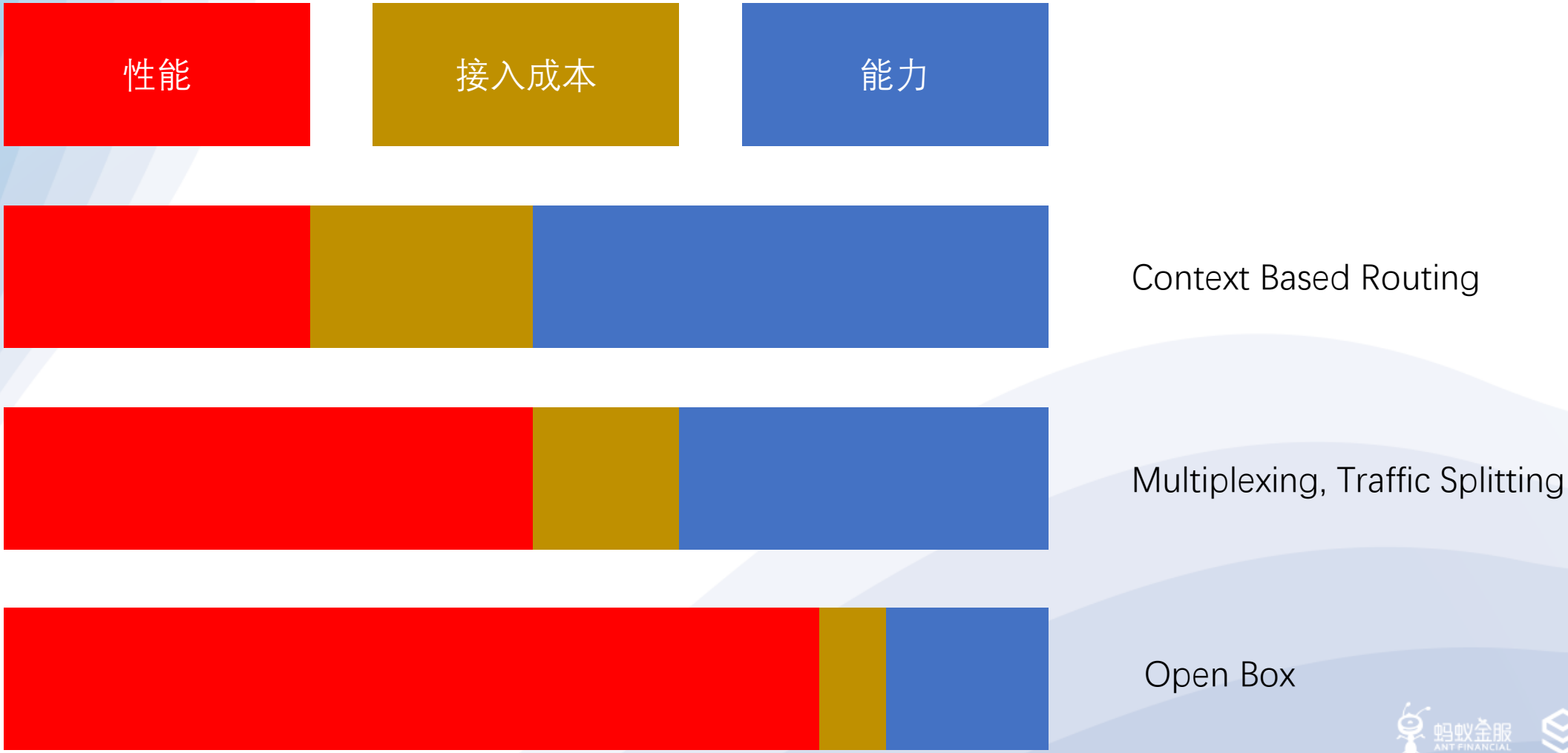
```
// 多路复用，Accesslog，流控，熔断能力
type Multiplexing interface {
    // 将Rpc数据包拆分为多个请求
    SplitRequest(data []byte) [][]byte
    GetStreamId(data []byte) string
    SetStreamId(data *[]byte,streamId string)
}

// Tracing能力
type Tracing interface {
    Multiplexing
    GetServiceName(data []byte) string
    // tracing 是否method粒度?
    GetMethodName(data []byte) string
}

// RequestRouting, RequestAccessControl, RequesstFaultInjection能力
type RequestRouting interface {
    Multiplexing
    GetMetas(data []byte) map[string]string
}

// 协议转换能力
type ProtocolConvertor interface {
    Multiplexing
    Convert(data []byte) (map[string]string, []byte)
}
```

解包是对 SIDECAR 代理性能影响最大的因素

多路复用是基本能力，其他能力需要在多路复用能力上构建

协议装换能力允许使用 HTTP2 作为 SOFA-MOSN 之间的通讯协议

基于能力分层的插件化扩展框架



SOFA MESH

- http://github.com/alipay/sofa-mesh
- 月底发布 0.2.0 版本
- 9月份启动在 UC 的落地
- 蚂蚁主站落地

Q & A

金融级分布式架构

ServiceMesher

欢迎关注微信公众号，获取更多技术干货！

http://www.servicemesher.com

蚂蚁金服 ANT FINANCIAL

SOFASTACK