术语介绍

tRPC知识库

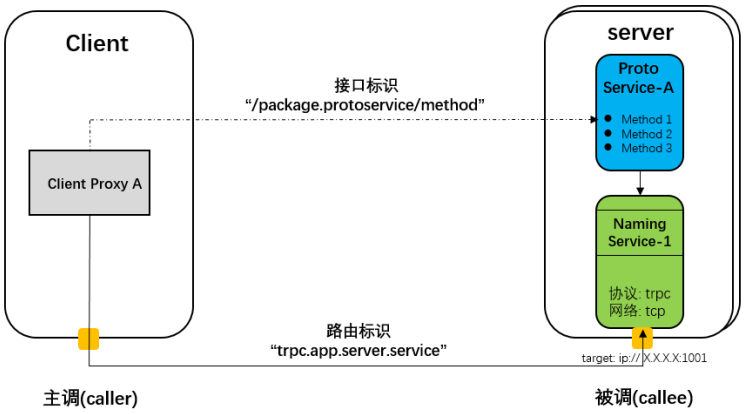
Exported on 06/29/2021

Table of Contents

目录

1 1 服务标识

在tRPC的世界里, 服务是如何被命名的? 服务间的函数调用是如何被识别的? 这里会涉及到很多的术语, 我们可以通过下面这张图来理解这些术语, 通过一个RPC的调用过程来把这些术语串联起来,进而解释这些术语之间的关系.



从RPC调用关系来看,分为客户端和服务端. 客户端也称为**"主调(caller)"**, 它是RPC调用的发起者. 服务端也称为**"被调(callee)"**, 它是服务的提供者.

1.1 1.1 路由标识

RPC调用的本质是一次网络通信的过程。我们从服务寻址的角度来看, 服务需要一个标识, 通过查询此标识就能获取服务的地址,也就是IP+端口+协议. 路由标识一般通过名字服务进行管理，标识必须全局唯一, 我们也把它称为 服务名 或 `Naming Service`。客户端通过路由标识向名字服务查询获取服务的地址。

在tRPC的设计中，并没有对Naming Service的命名格式做要求（字符串）。通常路由标识的命名格式和其所对接的名字系统或服务运营系统相关，例如在PCG 123系统中，路由标识是由.号分割的四段字符串：**"trpc.{app}.{server}.{service}"**来表示。在后续tRPC文档中，大部分示例都推荐采用**"trpc.{app}.{server}.{service}"**的命名格式来定义服务名。其中app， server， service的含义为：

- **app:** 应用名. 标识一组服务的一个小集合, 开发者可以根据需要自己定义, 通常表示实现某个业务系统名称
- **server**: 提供服务的进程名称, 一个 server 必须属于某个app, app 下的 server 名称都具备唯一性.一个server代表一个独立的程序, 绑定至少一个 ip, 实现至少一个service
- **service**: 服务提供者，提供接口规范供客户端调用。从网络通信的角度讲, 每个service对应一个IP + 端口 + 协议。

1.2 1.2 接口标识

在解决了服务寻址后, 我们再看看在通信报文中如何来识别一个RPC调用. RPC调用接口也需要一个全局唯一标识， 我们把它称为接口标识 或 `RPCName`, 它由 **"/{package}.{proto service}/{method}"** 三段组成, 各个字段的含义如下：

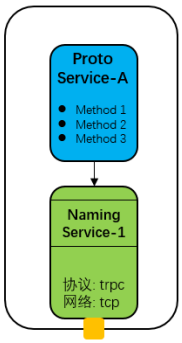
- **package:** 包名，tRPC对于服务的定义使用了protobuf的语法和概念. 这里的"package"对应于protobuf中的package, 可以理解为一个支持多级的名字空间, 用来防止接口命名冲突

- **proto service:** 协议服务名，对应于protobuf中的service, 是一组RPC方法的集合，它是对接口的一个逻辑分组。一个server内允许定义多个proto service
- **method:** 方法或接口，对应于protobuf中的rpc方法, 包括方法的名称，请求和响应报文的参数类型, 每个method必须且只能属于一个proto service. 一个proto service允许定义多个method

1.3 1.3 接口映射

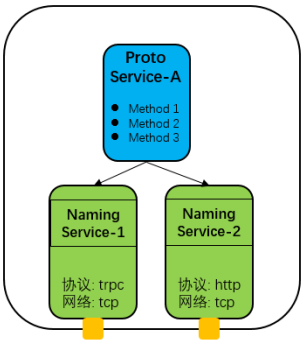
proto service负责定义RPC的接口的定义和归类， naming service负责网络通信连接的建立，协议的处理，它们是两个独立的概念。 我们需要把两者关联起来，通过把proto service注册到naming service来完成服务的组装。

基于微服务的“单一职责”原则， tRPC建议一个服务端程序只提供一个proto service，对应一个naming service，通过一个端口对外提供接口服务。服务组装模式如图所示:



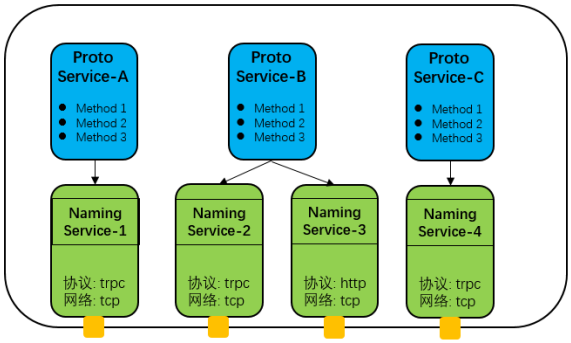
单服务模式
(推荐)

对于个别场景，业务需要对一个服务通过多种协议提供服务，例如: 一个认证服务需要通过http协议对外部系统提供认证服务，同时也需要通过tRPC协议对内部其它应用程序提供认证服务。对于这种场景，tRPC提供了单服务多协议的服务组装模式, 具体如图所示:



单服务多协议模式

系统同时也支持多服务模型， 即一个server有多个proto service，每个proto service对应一个或者多个naming service，多个proto service之间不共享naming service。对于这种模型的使用，开发人员可能审视微服务的设计是否可以合理，是否需要做拆分，以满足“单一职责”原则。服务组合模式如图所示：



多服务多协议模式

tRPC不支持一个端口提供多个Naming Service的功能。

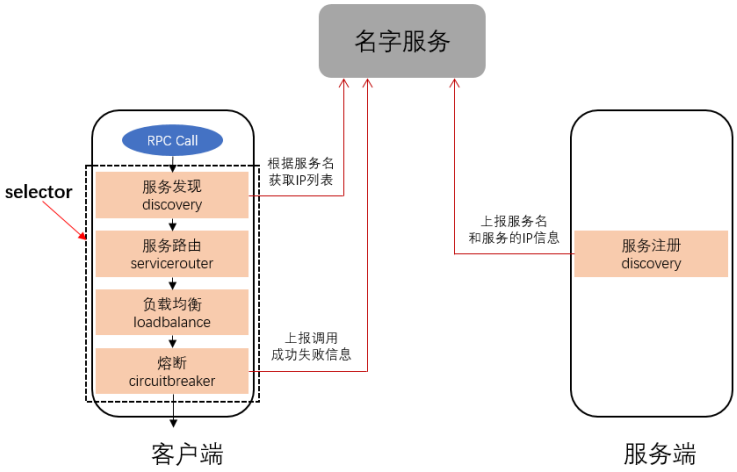
1.4 1.4 命名推荐

为了便于服务的管理,让服务路由名和接口有简洁明了的映射关系. 建议 package 的命名方式为："trpc.{app}.{server}", 这样上游服务只需要通过pb文件就能获知服务的路由名,而不需要询问下游的服务名字。

微服务理念推荐一个服务只做一件事，所以一般一个server最好就只有一个service, 建议 proto service的命名和service的命名保持一致

2 2路由寻址

名字服务是微服务治理中非常核心的一个环节, 在tRPC后续文档中会频繁提到. tRPC框架并不实现名字服务，但是会和名字服务密切交互，其中涉及到的术语非常多, 包括服务发现, 服务注册, 负载均衡, 熔断, 服务路由, 健康检查等等.下面我们会从一次RPC调用的过程来解释这些术语



服务端提供服务时, 首先需要告诉名字服务自己提供服务的服务名是什么, 服务端的地址(IP+端口+协议等)。同时还需要提供健康检查的机制(一般说是心跳检测)来让名字服务监控服务的存活情况的.

客户端调用服务的流程会经过服务发现, 服务路由, 负载均衡和熔断四个模块, 其流程如下:

1. 服务发现
   首先客户端会向名字服务询问指定服务名的地址, 服务端返回所有可提供服务的服务端地址列表.
2. 服务路由
   客户端基于预先设定的策略,对名字服务返回的地址列表进行筛选, 生成新的、满足策略的服务端地址列表. 常见的策略有: 就近路由, set路由, 元数据路由等. 策略的支持和名字服务能提供的功能是密切相关的. 常见策略可以参考<<北极星使用指南[1]>>
3. 负载均衡
   负载均衡模块支持从满足本次转发要求的服务实例集中，通过一定的负载均衡策略，选取一个实例供客户端进行服务请求发送. 常见的负载均衡策略有:random, round_robin, weight_round_robin, consistent_hash等.
4. 熔断
   对每个RPC请求的请求结果都会进行上报处理，熔断器会根据上报的情况，如果触发熔断，则会对服务器节点进行熔断处理。

在tRPC的设计中, 我们把客户端的服务发现, 服务路由, 负载均衡和熔断的组合称之为**"Selector"**.用户可以选择selector和指定的名字服务对接,使用名字服务所提供的服务路由, 负载均衡, 熔断等能力.

1 https://iwiki.woa.com/pages/viewpage.action?pageId=89656466

3 3 IDL语言

IDL(Interface description language)是用来描述软件组件接口的一种计算机语言。IDL通过一种中立的方式来描述接口，使得在不同平台上运行的对象和用不同语言编写的程序可以相互通信交流.

tRPC服务是采用protobuf IDL语言来定义服务的. tars服务采用的是jce IDL语言来定义tars服务的.通常情况下, 我们可以使用工具把IDL协议文件转换成指定编程语言的桩代码.

4 4 OWNER

##terryjzhang