PRPC

技术：

Binary：实现了数据和字节序列之间的转换，以及 varint 的编解码，提供了 LittleEndian 和 BigEndian 两种方式实现大小端字节序的转换。

雪花算法：Snowflake 雪花算法，由 Twitter 提出并开源，可在分布式环境下用于生成唯一 ID 的算法。该算法生成的是一个 64 位的 ID，第一位是符号位，不能改变，41 位的时间戳，10 位机器 ID，12 位的序列号，其中除了第一位的符号位不可改变，剩下的位数都可以变化。

Reflect：reflect 是 golang 中提供了反射功能的标准库，反射是一种机制，可以在编译时不知道具体类型的情况下，获得变量的类型信息和值信息。使用反射，可以让我们编写出能统一处理所有类型的代码。反射建立在类型系统上，类型分为内置类型和用户自定义类型，每种类型都有自己的类型描述信息，包括类型名称、大小、关联的方法等，叫做类型元数据，全局唯一，类型中存在一个特殊的类型接口类型，成为抽象类型，其他属于具体类型，接口类型由两部分组成动态类型和动态值，空接口的动态类型指向接口的动态类型元数据，动态值指向具体的值，非空接口的动态类型指向一个特殊的数据结构，该结构封装了接口类型元数据和动态值的类型元数据。reflect 可以通过 TypeOf 和 ValueOf 方法获取变量运行时的类型和值信息，具体过程是先将变量转为空接口类型，然后获取对应的动态类型元数据和动态值。

Redis：

幂等性解决方案：token、数据库唯一索引、redis 分布式锁

时间轮算法：把时间分成不同时间段，将定时任务划分到不同的时间段中，当时间跳到某个时间段就批量执行该时间段存储的定时任务。具体是实现是使用一个环形队列存储任务，底层采用数组实现，数组中每个元素代表一个时间槽，维护一个定时任务列表。定时任务列表是一个的双向链表，链表中的每一项代表定时任务项，其中封装了定时任务，每个时间槽代表当前时间轮的基本时间跨度（tickMs）。时间轮有一个表盘指针（currentTime），用来表示时间轮当前所处的时间，currentTime 是 tickMs 的整数倍。currentTime 指向的时间槽是到期的时间槽，需要处理时间槽中存储的定时任务。使用最小堆来实现表盘指针的转动，将堆中元素是维护了定时任务的时间槽，按照槽对应的时间刻度来构建最小堆，时间轮在启动时，启动两个 goroutine，分别用于定时任务监控和执行，监控 goroutine 会不断获取当前时间，从最小堆中获取定时任务，结果有三种，没有时间槽、定时任务未到时间，成功获取定时任务，最小堆同时维护了 sleeping 和 exit 两个通道，一个让监控 goroutine 堆中在没有元素时阻塞，另一个用于监控 goroutine 的退出。当获取结果是没有时间槽时，会进行阻塞，等有新元素加入堆顶时唤醒，定时任务未到时间是会阻塞等待有新元素加入堆顶或时间到期执行，成功时会将定时任务列表交给执行 goroutine 执行。每种情况都会监听 exit 是否有消息传递，便于快速退出 goroutine。执行和监控 goroutine 通过 channel 传递消息，当接收到定时任务时，会先更改表盘指针刻度，然后批量执行任务，期间也会监听 exit 通道。当表盘加入新任务时，会判断任务分到哪个时间槽，多级时间轮的情况会加上位于哪一级时间轮的判断，确定层级后通过取模操作获取槽存储任务，如果是第一个任务会将槽加到最小堆中。

Protobuf 协议：Protocol Buffers 是一种轻便高效的结构化数据存储格式，可以用于结构化数据串行化，或者说序列化。它很适合做数据存储或 RPC 数据交换格式。可用于通讯协议、数据存储等领域的语言无关、平台无关、可扩展的序列化结构数据格式，相对常见的 XML、JSON，描述同样的信息，ProtoBuf 序列化后数据量更小、序列化/反序列化速度更快、更简单。

http 库：http.Hijacker 接口定义的 Hijack 方法，通过该方法可以将从当前连接从 http 服

务器对象管理的 connection 列表中删除，不会再管理这个 TCP 连接的生命周期，转而交给使用者自己来管理。

介绍：

本项目是基于 net/rpc 中定义的编码器接口实现的远程过程调用框架，该框架除了基本的远程过程调用功能外，还新增了超时重传、负载均衡、服务发现、http 协议支持、注册中心等功能，同时基于 ProtoBuf 协议自定义设计编码器，支持多种压缩格式。

在整个框架中，一共定义了四类主要实体，分别是客户端、服务器、注册中心和编码器。

编码器封装了建立的 tcp 连接，net/rpc 的编码器接口接口定义了四个方法，分别是读消息头、读消息体、写消息、关闭函数，基于该接口分别实现了基于 gob 和 protobuf 两种序列化协议的编码器，自定义了传输的消息头格式，在 protobuf 编码器中使用 binary 实现消息头的序列化，将字节数据从主机字节序转为网络字节序。

客户端实现了两种客户端，分别是客户端和集成了服务发现模块的客户端，前者需要提供服务器地址，后者需要提供注册中心地址，对于客户端，需要实现对请求发送、响应接收、超时重传等功能，在实现过程中，客户端将每次调用请求抽象为 Call 类，每个 Call 实例表示一次正在执行的调用请求，包含了方法信息、调用结果、错误、信号 channel 等信息，在客户端进行调用时会生成 Call 并注册到客户端，当调用返回时从客户端删除对应的 Call，并通过信号 channel 向调用者传递结束信号，客户端两个核心函数是 send 和 receive，send 接受 Call 并将 Call 的调用信息封装成 rpc 请求，交给编码器将请求编码发送给远程服务器，receive 作为一个守护协程在客户端建立连接时启动，负责通过编码器接收来自服务器的请求响应，并根据响应序号从客户端注销对应的 Call，传递结束信号。客户端对外提供了两种调用方法 Go 和 Call，分别代表异步式调用和同步式调用，通过选择对信号 channel 的读取时机来实现。集成了服务发现模块的客户端封装了一个服务发现模块和一个客户端集合，客户端集合用 map 实现，用来缓存已经建立的客户端，使用 Call 进行调用时会使用服务发现模块获得服务器地址然后创建对应的 Client 发起调用，在创建过程中会先尝试从缓存中获取，客户端集合中无 Client 可用时创建新客户端执行调用并加入集合，Client 在创建时会发起一个定时 goroutine，定时关闭 Client。

服务器需要实现服务注册和请求响应功能，对于服务注册功能，服务器对外提供方法 Register 和 UpdateToRegistry 进行服务注册和远程服务更新，服务使用结构体定义，结构体关联的方法定义为该服务可以提供的调用方法，将注册的服务和方法抽象成 service 和 methodType 两个类，服务器中每个 service 代表一个注册的服务，service 和 methodType 是一对多的关系，当服务器收到一个服务注册请求后，会利用 reflect 对结构体进行解析，获取结构体的类型信息和值信息，封装到 service 类中，通过类型信息获取结构体拥有的方法信息并封装到 methodType 中，methodType 封装到 service 中，当进行请求调用时，会根据请求的服务名和方法名获取对应的 methType，然后利用 reflect 的 New 方法创建方法对应的入参和返回结果并使用 reflect 库的 Call 调用方法，服务注册以后会使用 UpdateToRegistry 方法更新到注册中心。对于请求响应，服务器将每次请求抽象为 RPCRequest 类，包含请求头、参数反射对象、methodType 和 service 类，服务器有四个核心函数，readRequest、handleRequest 和 sendResponse，readRequest 使用编码器读取请求的消息头和消息体，根据信息确定调用方法获取对应的 methType 实例，生成 RPCRequest 实例，sendResponse 使用编码器将请求头和调用结果发送到客户端，handleRequest 接收 RPCRequest，根据 RPCRequest 实例的信息执行对应的方法，调用 sendResponse 返回调用结果。

注册中心需要实现服务注册、服务订阅、心跳检测功能，注册中心的功能主要基于 http 实现，使用 services 和 servers 两个 map 类型的数据来维护服务器及其提供的服务，server

的 key 是服务器地址，value 是 serverItem 类，serverItem 类是注册服务器的抽象，记录了服务器的地址、提供服务、更新时间等；servers 的 key 是服务名称，value 是一个双向链表，链表节点指向可以提供 key 服务的 serverItem，通过 services 可以快速找到可以提供对应服务的服务器地址，通过 servers 可以实现对特定服务器的查询和更新操作。同时，服务订阅和心跳检测功能基于时间轮算法实现，客户端的服务发现模块在创建时会启动一个订阅协程来向注册中心订阅服务信息和定期接收注册中心的更新消息，注册中心在服务发现模块和服务器第一次请求时，会创建定时任务并加入到时间轮中等待执行，当进行注销时会从时间轮中删除对应的任务。

同时，基于时间轮算法还实现了客户端和服务器的超时处理功能，客户端一侧是超时重传，当客户端注册 Call 实例会将一个定时任务加入到时间轮中，如果未超时会在删除 Call 实例同时删除该定时任务，否定时任务会先判断重试次数，然后重新调用 send 发送调用请求，同时为了保证服务器一侧调用的幂等性，客户端一侧使用雪花算法生成一个全局唯一 seq 服务器采用 redis 对调用请求的 seq 使用 setnx 保存，通过判断是否保存成功来决定是否处理请求。

负载均衡功能实在客户端一侧进行负载均衡，客户端在进行调用时会从服务发现模块获取服务器地址然后建立相应的连接，服务发现模块利用负载均衡算法从维护的服务器地址列表中选出合适的地址返回给客户端，目前实现了随机和轮询两种均衡算法。

Balancer

该项目是基于 golang 的 http 库中的路由机制和 net 库下面的 httputil 包中的 ReverseProxy 实例实现的，ReverseProxy 是一个 HTTP 处理程序，它可以将传入的请求发送到另一个服务器，然后将响应返回给客户端。项目中抽象了一个 HTTP 代理类，代理类封装了两个哈希表 hostMap 和 alive 和一个均衡器实例，hostMap 的 key 是域名地址，vaue 指向该域名地址的 ReverseProxy 实例，alive 记录了每个域名地址对应服务器的可用状态，均衡器是一个自定义接口类型，该接口定义了一组方法，分别是服务器的增删方法、服务器连接数的加减方法，还有请求处理服务器的获取方法，按照不同的负载均衡算法定义了多个实现了该接口的均衡器类，以及对应的实例创建函数，并将该函数和算法名称以键值对的形式保存在一个全局哈希表中，创建 HTTP 代理类时根据选择的算法获得对应的均衡器实例。服务器在接收到请求时，首先会进入路由 (router)，路由根据请求路径找到对应的处理器，处理器对接收到的请求进行相应处理后构建响应并返回给客户端。Golang 的 http 库里路由需要实现一个 ServeHTTP 方法，HTTP 代理类充当是的这样一个路由的角色，可以在服务器运行时将 HTTP 代理类作为一个路由注册到 server 类中并设置好匹配的请求路径，就可以实现对 http 请求的一个负载均衡功能。

介绍 TCP、UDP 区别

连接性：TCP 是面向连接的协议，而 UDP 是无连接的协议。TCP 在传输数据之前需要先建立连接，而 UDP 则不需要。

可靠性：TCP 是一种可靠的协议，它提供了多种机制确保数据的完整性和可靠性。而 UDP 则不提供这种保证，因此在传输过程中可能会出现数据丢失或错误。

服务对象：TCP 是一对一的两点服务，即一条连接只有两个端点，UDP 支持一对一、一对多、多对多的交互通信。

传输方式：TCP 是面向字节流的，没有边界，而 UPD 是面向报文的，以报文段形式发送。

速度：由于 TCP 需要建立连接和保证数据可靠性，因此它的传输速度相对较慢。而 UDP

则没有这些限制，因此它的传输速度相对较快。

用途：TCP 主要用于需要确保数据可靠性的应用程序，如文件传输、电子邮件、网页浏览等。而 UDP 则主要用于需要快速传输数据的应用程序，如实时视频、音频、游戏等。

进程间通信方式

管道、消息队列、共享内存、信号量、信号、socket

网络层有哪些路由协议

IP（Internet Protocol，网际协议）：TCP/IP 协议中最重要的协议之一，主要作用是定义数据包的格式、对数据包进行路由和寻址，以便它们可以跨网络传播并到达正确的目的地。目前 IP 协议主要分为两种，一种是过去的 IPv4，另一种是较新的 IPv6，目前这两种协议都在使用，但后者已经被提议来取代前者。

ARP（Address Resolution Protocol，地址解析协议）：ARP 协议解决的是网络层地址和链路层地址之间的转换问题。因为一个 IP 数据报在物理上传输的过程中，总是需要知道下一跳（物理上的下一个目的地）该去往何处，但 IP 地址属于逻辑地址，而 MAC 地址才是物理地址，ARP 协议解决了 IP 地址转 MAC 地址的一些问题。

ICMP（Internet Control Message Protocol，互联网控制报文协议）：一种用于传输网络状态和错误消息的协议，常用于网络诊断和故障排除。例如，Ping 工具就使用了 ICMP 协议来测试网络连通性。

NAT（Network Address Translation，网络地址转换协议）：NAT 协议的应用场景如同它的名称——网络地址转换，应用于内部网到外部网的地址转换过程中。具体地说，在一个小的子网（局域网，LAN）内，各主机使用的是同一个 LAN 下的 IP 地址，但在该 LAN 以外，在广域网（WAN）中，需要一个统一的 IP 地址来标识该 LAN 在整个 Internet 上的位置。

OSPF（Open Shortest Path First，开放式最短路径优先）：一种内部网关协议（Interior Gateway Protocol，IGP），也是广泛使用的一种动态路由协议，基于链路状态算法，考虑了链路的带宽、延迟等因素来选择最佳路径。

RIP(Routing Information Protocol，路由信息协议）：一种内部网关协议（Interior Gateway Protocol，IGP），也是一种动态路由协议，基于距离向量算法，使用固定的跳数作为度量标准，选择跳数最少的路径作为最佳路径。

BGP（Border Gateway Protocol，边界网关协议）：一种用来在路由选择域之间交换网络层可达性信息（Network Layer Reachability Information，NLRI）的路由选择协议，具有高度的灵活性和可扩展性。

讲讲 ARP

cookie 和 session 的区别

Cookie 和 Session 都是用来跟踪浏览器用户身份的会话方式。

Cookie 一般用来保存用户信息，服务端给特定的用户创建特定的 Session 之后就可以标识这个用户并且跟踪这个用户了。

Cookie 数据保存在客户端(浏览器端)，Session 数据保存在服务器端。

相对来说 Session 安全性更高。如果要在 Cookie 中存储一些敏感信息，不要直接写入 Cookie 中，最好能将 Cookie 信息加密然后使用到的时候再去服务器端解密。

session 的运行依赖 session id，而 session id 是存在 cookie 中的，也就是说，如果浏览器禁用了 cookie ，同时 session 也会失效。

为什么要用 redis 为什么要用缓存？

redis 和 memcached 的区别？

redis 常见数据结构以及使用场景分析
redis 内存淘汰机制（MySQL 里有 2000w 数据，Redis 中只存 20w 的数据，如何保证 Redis 中的数据都是热点数据？）
redis 持久化机制（怎么保证 redis 挂掉之后再重启数据可以进行恢复）？
缓存雪崩和缓存穿透问题解决方案？
如何解决 Redis 的并发竞争 Key 问题？
如何保证缓存与数据库双写时的数据一致性？

网络传输时为什么要定义消息格式

在网络传输中，定义消息格式非常重要。这是因为在网络中，数据是通过不同的计算机和设备进行传输的，这些计算机和设备可能使用不同的编程语言、操作系统和硬件。因此，为了确保数据能够正确地传输和解释，需要定义一种标准的消息格式。

消息格式定义了消息的结构和内容，包括消息头、消息体和消息尾等部分。这有助于接收方正确地解析和处理消息，避免出现解析错误或数据丢失等问题。消息格式还可以帮助开发人员更好地组织和管理数据，提高代码的可读性和可维护性。

此外，定义消息格式还可以提高网络传输的效率。通过使用紧凑的消息格式，可以减少数据的传输量，从而提高传输速度和降低网络延迟。因此，在网络传输中，定义消息格式是非常必要的。

既然有 HTTP 协议，为什么还要有 RPC?

HTTP 主要用于 B/S 架构，适用于需要频繁与不同服务器进行资源交流和通信的应用场景，通用性更强，但需要在报文头花费更多空间记录双方信息。而 RPC 更多用于 C/S 架构，适用于需要进行高速、大规模数据传输的应用场景，定制化程度更高，双方的一些信息可以直接进行配置，不需要在通信过程中记录，报文头更小，传输效率更高。

反向代理

反向代理是充当服务器网关的代理服务器。当您将请求发送到使用反向代理的服务器时，他们将先转到反向代理服务器，由该代理将确定是将其路由到服务器还是将其阻止。相当于"代理服务器"代理了"目标服务器"，去和"客户端"进行交互。通过反向代理可以避免请求客户端与服务器进行直接通信，通过负载平衡和缓存，可以保护服务器避免遭受攻击。

反向代理的用途：1.隐藏服务器真实 IP，使用反向代理，可以对客户端隐藏服务器的 IP 地址。2.负载均衡，反向代理服务器可以做负载均衡，根据所有真实服务器的负载情况，将客户端请求分发到不同的真实服务器上。3.提高访问速度，反向代理服务器可以对于静态内容及短时间内有大量访问请求的动态内容提供缓存服务，提高访问速度。4.提供安全保障，反向代理服务器可以作为应用层防火墙，为网站提供对基于 Web 的攻击行为（例如 DoS/DDoS）的防护，更容易排查恶意软件等。还可以为后端服务器统一提供加密和 SSL 加速（如 SSL 终端代理），提供 HTTP 访问认证等。

正向代理:是一个位于客户端和原始服务器(origin server)之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定原始服务器，然后代理向原始服务器转交请求并将获得的内容返回给客户端。相当于"代理服务器"代理了"客户端"，去和"目标服务器"进行交互。（翻墙软件）

正向代理的用途：1.突破访问限制，通过代理服务器，可以突破自身 IP 访问限制，访问

国外网站，教育网等。2.提高访问速度，通常代理服务器都设置一个较大的硬盘缓冲区，会将部分请求的响应保存到缓冲区中，当其他用户再访问相同的信息时，则直接由缓冲区中取出信息，传给用户，以提高访问速度。3.隐藏客户端真实 IP，上网者也可以通过这种方法隐藏自己的 IP，免受攻击。