# 主Loop

要实现一套异步事件循环系统，我们对于主Loop的需求主要是两个:

1. 独立的循环。
2. 高精度定时器。不会因为系统的时间同步、人工时间调整而导致定时器功能紊乱。

目前不考虑兼容其他系统,只考虑linux,因此只需要支持epoll.目前可用的Loop: libEv/libEvent/libUV/Redisae

前三者都包含了定时器,Redisae也包含定时器,但是Redisae是一个链表,对于我们显然并不适合。我考虑的是将Redisae抽出来将定时器的数据结构修改为最小堆。原因在于Redisae足够简单。实际上我们也只需要一个足够简单的主Loop。

另外一个考虑在于,框架需要考虑插件化,势必要自行管理Loop,前三者功能已经非常齐全,但是从新进行改造成本太高而且也比较麻烦。

## 需求分析思考

对于一套证券交易系统,我们对通信主要有以下考虑:

1. 动态路由

动态路由的目的在于能够使系统进行动态的请求分发以及负载均衡处理。一旦出现机器繁忙或者节点不够用的情况下能够通过手工甚至于自动的方式进行资源扩展

同时路由就意味着发起的请求能够正常将应答返回。一种情况原路返回,这是最简单的情况,带来的问题很明显,一旦某一个节点挂掉了,就会导致应答异常。还有一种通过路由的动态寻址协议,但是这种协议实现起来要求较高。

这里我们换一种方法,我们的目的在于实现交易系统的功能而不是实现通用路由功能。因此采用服务发现网络。也就是:

1. 正常情况下,通过中心gateway#1 #2进行注册发现服务。
2. 当业务A初始需要与业务B建立连接时,先从中心gateway#1 #2获得B的服务信息。B是一类服务,我们假定A并不知道B的地址。实际上在证券交易系统里面所有业务的地址基本上都是固定的,只是需要在系统搭建时通过人工配置,这里面我们采用注册制目标也在于降低人工运维,让服务能够自动发现。



1. 高可靠

在通信过程要求数据不允许出现丢失,同时在链路断线之后能够自动进行重连尝试,使得业务能够尽快恢复

1. 节点状态的动态监控

在证券交易系统里面,除了业务本身可靠之外,机构有固定的运维人员对系统进行维护,那么此时就需要提供一种监控方法能够实时发现问题、上报问题、解决问题

1. 并发处理

证券交易系统里面实际上存在一个矛盾问题,对于系统来说实际上总是希望容量越大越好,但是实际上由于交易系统和钱息息相关,那么这里面必然涉及到一个串行数据处理以确保数据准确的问题。最常见的例如可用的控制,对于同一个账户的可用明显是不允许并发处理的,这样可能导致可用被穿透,带来灾难性的影响

在一套证券交易系统里面,我们考虑的最小交易单位是账户(子账户也认为是独立账户，只是与母账户有关联关系而已)。在这种考虑之下,我们对于系统并发度的设计是非关联账户并发操作,关联账户串行操作。例如一个机构内部的账户尽可能放到一起,跨机构的账户就可以完全独立。

## 进程间通信

## 线程间通信

## 协议

## 定时器

## 无锁编程

## 热内存

## 负载均衡与热备

## 日志

# 插件

# 业务体系

## 账户体系

## 用户体系

## 算法

## 订单

## 接口适配器

## 风控

## 清算结算

## 安全体系

# 模块细分

## 算法服务

## 风控服务

## 基础管理

## 清算结算

## 接口适配器

# 具体流程