**BlotMQ网络文档说明**

针对版本V1.0.0

**©成都基础平台架构**

**2017/11/21**

Smartgo-broker修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 修订内容 | 作者 | 审核 | 修订日期 |
| V1.0.0 | 初始版本 | 罗继 | 基础平台架构组 | 2017/11/22 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1 概述 4](#_Toc499132555)

[2 背景 4](#_Toc499132556)

[3 专业术语 5](#_Toc499132557)

[4 网络实现原理 5](#_Toc499132558)

[4.1 设计与交互 5](#_Toc499132559)

[4.2 连接管理 6](#_Toc499132560)

[4.3 事件通知 6](#_Toc499132561)

[4.4 粘包 7](#_Toc499132562)

[4.5 通讯包结构 7](#_Toc499132563)

[4.5.1 RemotingCommand结构 8](#_Toc499132564)

[4.6 报文压缩 8](#_Toc499132565)

[4.7 心跳处理 8](#_Toc499132566)

[附件一 Smartgo开发者联系方式 9](#_Toc499132567)

# 概述

本章将从网络层面讲解Smartgo，它采用的IO模型（epoll），如何实现的事情通知？粘包是怎么实现的？以及消息交互报文的协议，将一一进行说明。 Smartgo在网络层面使用TCP长连接作为通讯方式，RocketMQ使用Netty库为基础网络开发库，netty是事件驱动的网络编程框架和工具，它的强大毋庸置疑。而Golang目前还没有和Netty类似的实现库，所有需要构建一个性能优异的网络基础库。

# 背景

Smartgo在网络层面使用TCP长连接作为通讯方式，RocketMQ使用Netty库为基础网络开发库，netty是事件驱动的网络编程框架和工具，它的强大毋庸置疑。而Golang目前还没有和Netty类似的实现库，所有需要构建一个性能优异的网络基础库。

**IO模型**

Netty是基于NIO(Non-blocking I/O)的实现，NIO的多路复用select/epoll默认使用epoll，可以在不同操作系统有不同选择。Smartgo同样选择了epoll，Golang的net包标准库底层使用了epoll，在runtime层面，是用epoll/kqueue实现的非阻塞io，为性能提供了保障。不同的是开发者层面任然是阻塞的，配合Golang的线程模型CSP能达到高性能。

**连接管理**

Smartgo的netm包提供的统一的连接管理功能，将所有连接统一管理，简化使用者维护连接。该功能讲会在之后去除，由事情通知功能代替，连接维护交由使用者维护。

**事件通知**

提供类似于Netty的事情通知功能，但目前只支持少数几类事情，提供代码的重用行。

**粘包**

这里的粘包是业务粘包，标准net包在底层提供了粘包保证了报文的正确性。业务报文是否完整，将进行粘包处理。

**报文协议**

Smartgo中报文格式定义分为header和body，这两部分都定义了格式来进行通信。这部分将介绍具体的格式以及含义。

# 专业术语

# 网络实现原理

## 设计与交互



图中是整个网络层的设计以及报文的处理流程。分为client和server端，client负责创建连接，报文的封包和拆包。服务器除此之外还要维护客户端连接，保证连接能接收数据。缓存队列可以缓存突发流程的，保证程序的可靠性。

客户端首先创建连接，连接创建成功，发送消息并等待响应消息。发送消息前会将报文进行编码，接收到消息后也会将消息进行解码。

服务器端会启动端口监听，接收来自客户端的连接。当有连接连上的时候，服务端用一个新的Goroutine接收客户端连接。当接收到一个客户端发送的消息时，会将消息发到队列中，然后会从队列中取出消息进行粘包。最后将完整的报文交给业务进行处理并响应。这里说明一点的是，队列和粘包都针对的单个连接，减少资源的竞争。

## 连接管理

当客户端创建连接或者服务端接收一个连接时，将会把连接放入到一个map中，连接地址作为key。同时将新建Goroutine接收连接所接收到的信息。该功能会在之后删除，由事件通知替代。

## 事件通知

当连接状态发送变化时，将该事件通知给用户。支持事件:

* Active: 当接收到一个新创建的连接时，被动接收通常作为服务端。
* Connect: 当新创建一个新连接时，通常主动创建连接的客户端。
* Disconnet: 当连接时断开时，通常被动断开。
* Closed: 当连接时断开时，通常主动关闭。
* Error: 当连接使用中发生错误时。

## 粘包

粘包指的是业务粘包，标准net包在底层提供了粘包保证了报文的正确性。业务报文是否完整，将进行粘包处理。Smartgo采用length-field的方式传输报文。length占用4个字节，存储之后的报文长度。粘包就是将接收的报文进行验证，先验证length域，在根据length域的值取得field域。如果length长度不够，会将报文进行缓存，等待下一个报文的到来。粘包必须是针对单个连接进行，保证传输报文的不乱序。



* length域: 报文的长度。
* field域: 报文内容。

## 通讯包结构



* length域: 报文的长度。
* header length域: 报文头部长度。
* header域: 报文头部。
* body域: 报文内容。

header和body域的数据解析后是RemotingCommand这个结构，通信时将RemotingCommand序列化成byte[]字节数组通信层进行传输。

### RemotingCommand结构

|  |  |  |
| --- | --- | --- |
| **字段名** | **请求** | **响应** |
| **code** | 请求操作代码，请求接收方根据不同的代码做不同操作 | 应答结果代码，0表示成功，非0表示各种错误 |
| **Language** | 请求发起方实现语言 | 响应方实现语言 |
| **Version** | 请求方程序版本 | 响应方程序版本 |
| **Opaque** | 请求标识代码，多线程，连接复用使用 | 应答方不做修改，直接返回 |
| **Flag** | 通信层的标志位 | 通信层的标志位 |
| **Remark** | 传输自定文本信息 | 错诨详细描述信息 |
| **ExtFields** | 请求自定义字段 | 响应自定义字段 |
| **CustomHeader** | 自定义结构，传输时将其转换为extFields型数据 | 自定义结构，传输时将其转换为extFields型数据 |
| **Body** | 请求body | 响应body |

## 报文压缩

报文达到一定长度后，提供报文压缩功能。压缩算法使用zip。RemotingCommand的SysFlag标识的第二位标识报文是否为压缩。

## 心跳处理

通信组件本身不处理心跳，由上层进行心跳处理。

# 附件一 Smartgo开发者联系方式

|  |  |  |
| --- | --- | --- |
| 姓名 | 联系方式 | 更新日期 |
| 谭特贤 | tantexian@qq.com | 2017/09/11 |
| 郜焱磊 | gyl\_adaihao@163.com | 2017/11/21 |
| 田玉粮 |  | 2017/11/21 |
| 尹同强 |  | 2017/11/21 |
| 罗继 | gunsluo@gmail.com | 2017/11/21 |
| 周飞 |  |  |
| 戎志宏 |  |  |