第六章节：WebSocket断线自动重连

前言

上一章节我们完成了一个单机IM服务该具备的基本功能，但是还存在一些问题，比如当server因为其他原因重启或者宕机时，client端还持有一个已经断开的连接，这个连接无法收发消息，所以client端需要检测当前连接是否可用，当发现不可用时需要重连。

检查连接

在`gorilla-websocket`提供的`websocket.Conn`相关`ReadXX`方法中，当连接被关闭时会返回`websocket.CloseError`这个错误，所以client可以通过返回的错误是`websocket.CloseError`时进行重连。

首先将连接WS server的方法抽取出来, 在`client.go`中添加connect方法

```
// 连接到WS server，并带有retry
func connect(url string, header http.Header) (*websocket.Conn, error) {
    var (
        conn *websocket.Conn
        err  error
    )
    maxRetries := 5
    for attempt := 0; attempt < maxRetries; attempt++ {
        conn, _, err = websocket.DefaultDialer.Dial(url, header)
        if err == nil {
            log.Println("connected to server ...")
            return conn, nil
        }
        if attempt < maxRetries {
            // 失败后的延迟
            time.Sleep(time.Second)
        }
    }
    return nil, err
```

}

```

在`client.go`中添加判断连接断开

```
func isClosed(err error) bool {
    _, ok := err.(*websocket.CloseError)
    return ok
}
```

修改`client.go`的main方法

```
func main() {
    ... // 省略
    u := model.User{
        ID: userID,
        Name: args[1],
    }
    token := u.GenToken()
    header := http.Header{
        "X-TOKEN": []string{token},
    }
    conn, err := connect(uri, header)
    if err != nil {
        log.Fatal("dial failed:", err)
    }
    defer conn.Close()

    go func() {
        for {
            m := &model.Message{}
            err = conn.ReadJSON(m)
            if err != nil {
                // server shutdown or socket closed
                if isClosed(err) {
                    // reconnect
                    log.Println("Trying reconnect ...")
                    conn, err = connect(uri, header)
                    if err != nil {
```
```

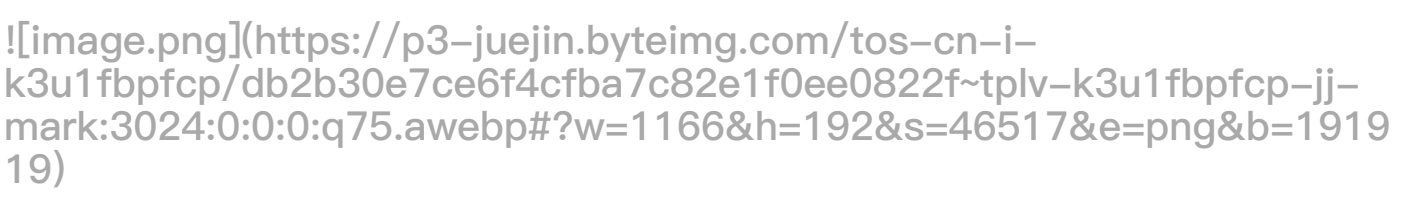
```
                    log.Fatalln("Reconnect Failed...", err)
                    return
                }
                continue
            }
            log.Println("Read WS message failed:", err)
            continue
        }
        log.Printf("Received Message %v From %v \n", m.Data,
m.FromUserID)
        err = markMsgRead(*m, token)
        if err != nil {
            log.Println("MarkRead failed:", err)
        }
    }
}()
... // 省略
}

```

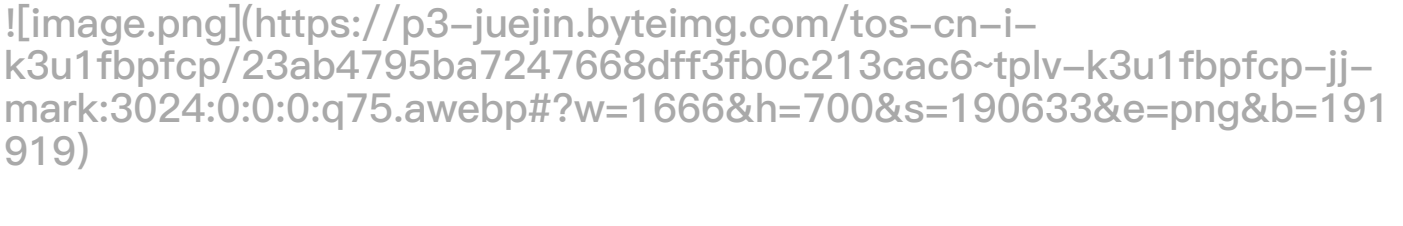
调试
--

首先分别在2个Terminal中启动server和client, 然后停止server，观察client端的日志，5秒内立即重启server，观察server和client的日志，可以在server端看到client连接的日志，然后再新开一个Terminal用另一个用户登录，可以看到用户间的收发消息功能正常。

Client日志



Server日志



小结

本章节中我们在客户端通过识别错误类型判断连接是否可用，并在连接不可用时进行重连，这样当server端短暂重启或因网络波动导致连接断开时可以自动重新进行连接，提高了服务的可靠性。但是随着客户端（用户）变多，要提高服务的高可用性，服务端必须采用多实例部署方案，也就是分布式系统，下一章节我们将探讨服务端如何在分布式环境下共享连接状态、通信。