长连开发经验分享官方

能力更新2018-09-29

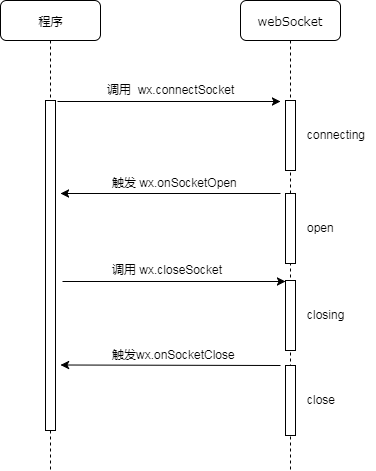
在微信小程序/小游戏的开发中，网络传输主要是依靠http的短连接和webSocket 长连接来完成的。

在一般web服务中，大多使用短连接来向服务器请求资源，与服务器的交互频率低，次数少。而在一些需要与服务器**交互频繁**，需要及时收到服务器推送的场景，比如**直播、多人实时游戏**，更适合使用 webSocket 进行通讯。

这次的小故事主要分享 webSocket 在微信小程序/小游戏开发上的一些经验。

**长连的生命周期介绍**

**webSocket的生命周期一共有4个状态：connecting、open、closing、closed。**我们可以通过 socketTask 的 readyState 属性来获取当前 webSocket 长连的状态。webSocket 的生命周期过程和 API 间的调用关系可以简单的入下图所示。



**注意：只有长连处在 open 状态，才能够正常的收发消息，其他状态均会报错。**

**客户端长连的断开机制**

当小游戏进入到后台运行超过5秒时，客户端会禁止小游戏的所有网络连接。这是一个非常频繁的断线逻辑，十分考验程序断线错误处理逻辑。建议大家可以在用户点击右上角按钮退出小程序/小游戏时，**主动帮用户断线，待用户切回时再重接上去。**

当 webSocket 长连超过一段时间没有任何网络传输时，客户端会主动关闭这条长连，以节省资源。开发者可以设置业务心跳，每隔一段时间与后台进行一次通讯，维持长连。

**如何选择长连的接口**

API接口主要有两类，一类是前缀为 “wx” 的接口，一类 “socketTask” 的接口。举例，同样是连接长连后发送一条消息，两种写法区别如下。



最初小游戏只允许存在1个 webSocket 连接时，并没开放 socketTask 的管理方式。随着小游戏的能力提升，**可支持同时存在的 webSocket 连接个数变多**，在使用 wx.connectSocket 创建 webSocket 连接时会返回 socketTask 任务对象，便于去管理每一条连接链路。

**推荐开发者尽量使用 socketTask 的方式去管理 webSocket 链接，每一条链路的生命周期都更加可控。**同时存在多个 webSocket 的链接的情况下使用 wx 前缀的方法可能会带来一些和预期不一致的情况。例如：**当存在多条连接时，wx.onSocketOpen、wx.sendSocketMessage、wx.onSocketMessage 等接口会只作用于第一条连接的长连。**且wx.onSocketOpen 接口不能多次注册 webSocket 长连的回调函数，仅最后一次生效。使用 socketTask 任务的方式则不会出现上述问题。

**开发与调试的建议**

01

微信提供了 webSocket 最基础的接口能力，开发者可以**在其基础上进行封装，根据业务需要扩展能力**。比如封装一个 offSocketOpen 的方法来取消注册 socketOpen 的回调函数。

02

长连并没有像短连那样“一问一答”的交互形式。在某些场景下，开发者需要这种与服务器的交互。**建议前端与后台协议，每条客户端上行的信息，服务器都下发一个对应的回包，去“模拟短连”。**比如开发者向服务器询问1+1和1+2等于多少，服务器返回了3和2，便可清晰知道哪一个数字对应着哪一个请求的答案。此外，还可以设置业务超时逻辑，便于判断上传是否丢包

03

**在 webSocket 发送数据时，数据格式可以选择string或者ArrayBuffer**。这里要注意的是，由于小游戏禁止了 Function() 和 eval()语法。所以像 protobufjs 这类用了这些语法的库是不能直接拿来用的。

04

在测试调试长连的时，目前开发者工具不支持通过设置 offline 模拟长连断网的情况（短连是支持的），所以在测试断线重连的一些情况时，可以**辅助一些第三方工具，或者用真机调试以及“拔网线”的方式来测试**。

05

在进行多人游戏测试时，在开发者工具中熟练使用**“自定义编译条件”**，以及**“多账号调试”**这两个功能可以极大的提升开发测试效率。

06

长连占用的系统资源，会导致手机发热比较明显。所以在**不需要使用 webSocket 的场景下，建议及早断开长连，需要时再连接。**

**异常断线的监控**

监控长连是否异常断线，在长连的使用中，尤其在小游戏多人对战中是尤为重要的。**socketError 事件并不能认为是异常断线。**

首先 socketError 事件并不一定会导致断线，其次若是由客户端机制断开的长连，是不会触发 socketError 事件的。

最简单的方式可以通过 onClose 回调函数触发时系统传入的 code 是否为1000来判断。当然开发者自身也可以通过代码判断是否是自身调用的 close 函数触发的 onclose 事件，监控异常断线。

希望大家在实际应用中能帮助到到大家。