基于WebSocket通信方案（调研）

架构设计文档

互联网事业群/商业化中心/移动商业化运营部

移动商业化化技术组

目录

[一、 简介 1](#_Toc35268642)

[1. 业务目标 1](#_Toc35268643)

[2. 考虑要点 1](#_Toc35268644)

[3. 技术选型 1](#_Toc35268645)

[4. 方案概要 1](#_Toc35268646)

[二、 Swoole进程模型 2](#_Toc35268647)

[三、 技术设计方案 4](#_Toc35268648)

[1. 核心流程图 4](#_Toc35268649)

[2. 服务拓扑图 4](#_Toc35268650)

[3. 通信协议 5](#_Toc35268651)

[4. 数据结构 6](#_Toc35268652)

[四、 实现 7](#_Toc35268653)

[五、 方案验证 7](#_Toc35268654)

[1. TCP通讯包 7](#_Toc35268655)

[2. 文档详细 7](#_Toc35268656)

[六、 可能风险 7](#_Toc35268657)

[七、 结论 7](#_Toc35268658)

1. 简介
   1. 业务目标
2. 作为移动端与服务器端通信方案
   1. 考虑要点

本方案对以下技术要点做了重点考虑：

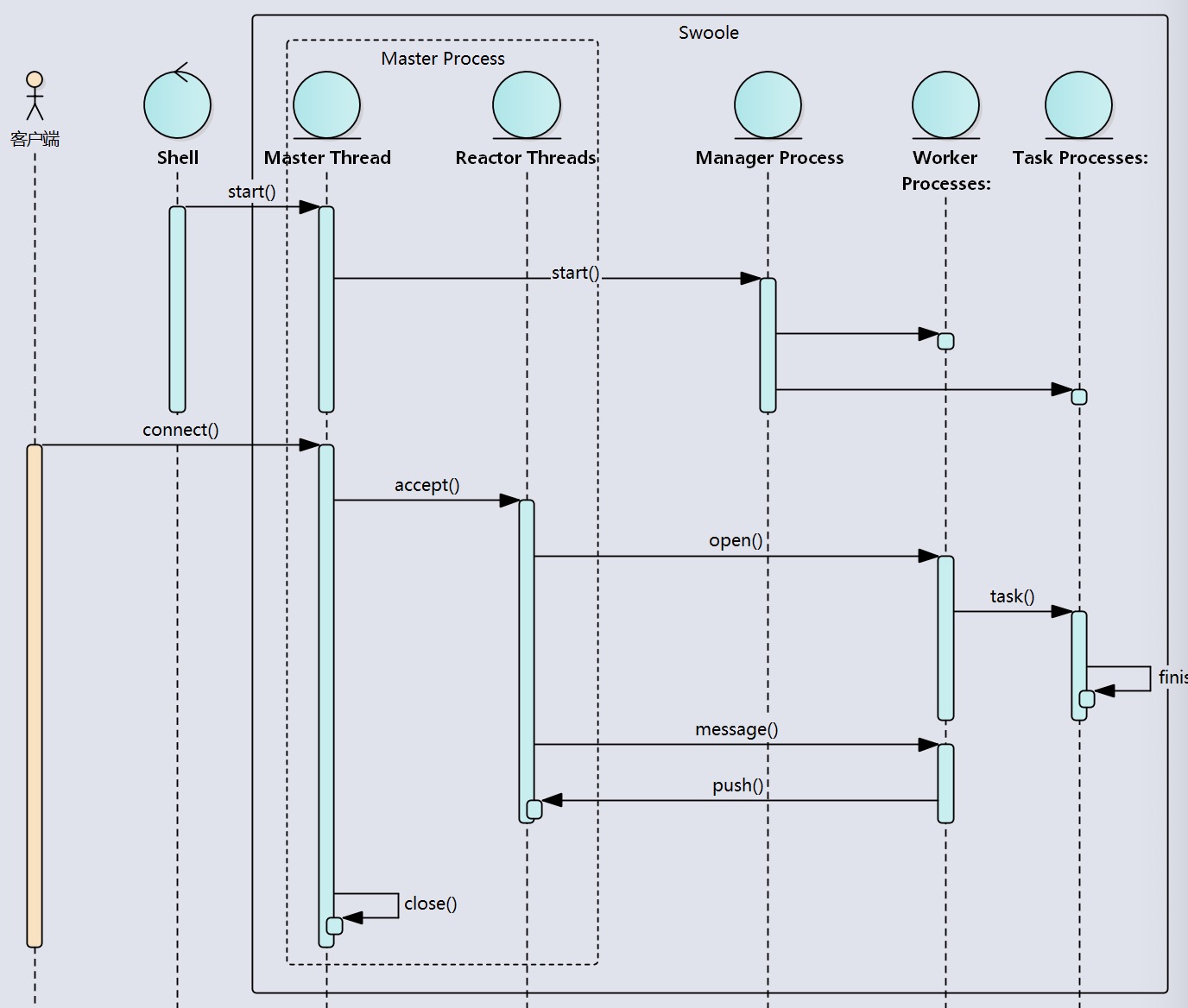
1. 并发问题：包含两方面，既能够满足高并发要求，同时还能够避免因并发连接过多而导致整个服务器不可用问题；
2. 超时问题：能够处理连接超时问题，避免长时间闲置的连接对服务器资源的占用；
3. 主动消息推送问题：能够满足服务器端向客户端主动推送数据的需要；
4. 平滑关闭及重启问题：服务器能够相对友好地关闭、重启，将服务器下线、重启等操作对客户端的影响最小化；
5. 运维及监控问题：服务器运行状况可以被监控，异常状态可以预警，最好可以通过监控平台干预服务器的运行；

此外，还考虑了应用程序的灵活性、可扩展性、可维护性

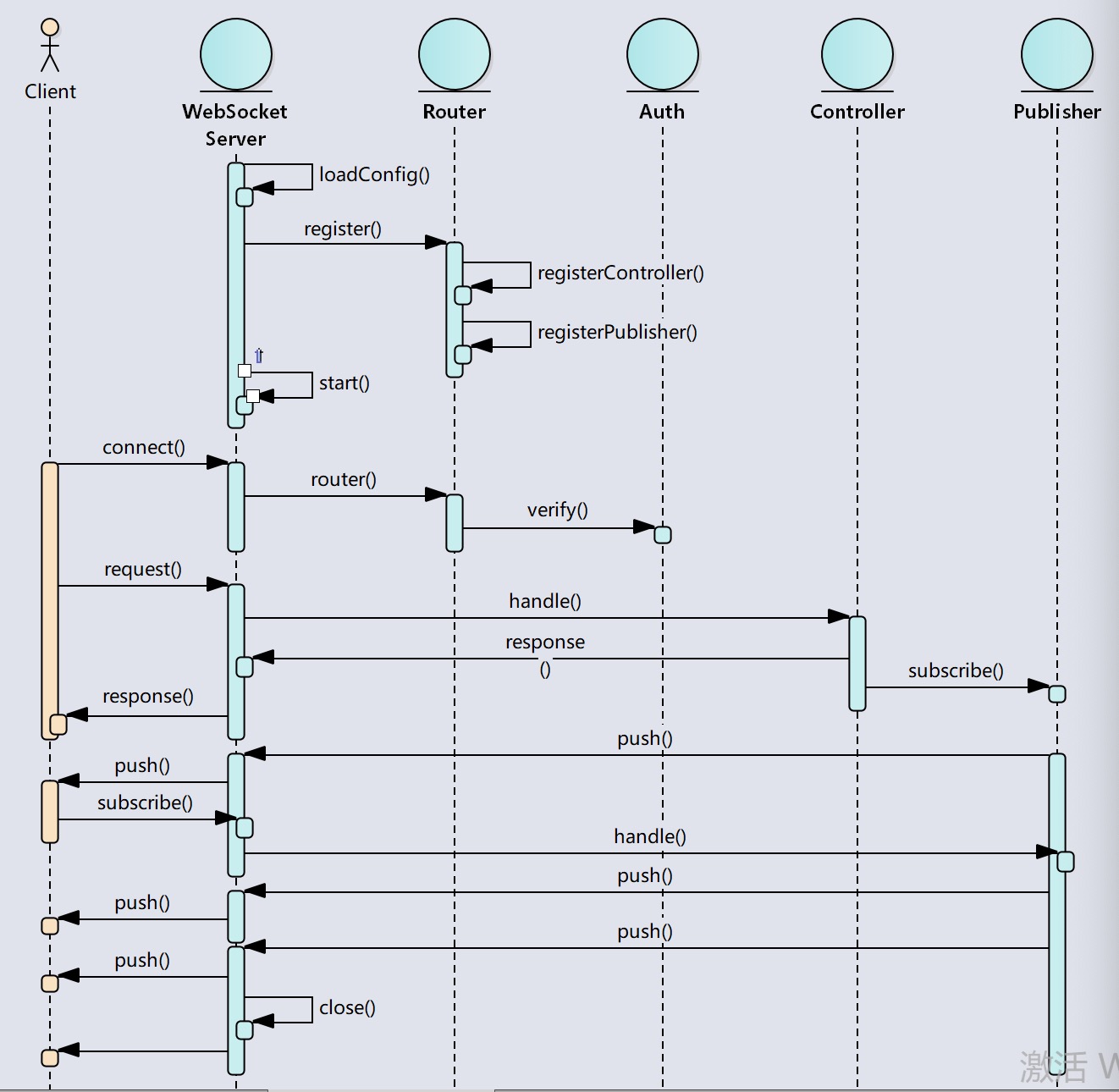
* 1. 技术选型

1. Swoole 作为PHP社区为数不多的网络通信基础库，经过多年实践检验，可信可靠；
2. Swoole能够有效支持高并发，理论上单机能够承载50万甚至数百万的并发连接，同时能够方便地对最大并发数进行限制；
3. Redis除了作为高性能缓存和内存数据库外，还支持消息队列，可以作为系统与系统之间的数据交互管道
4. 本方案使用最新稳定版的Swoole(4.4.16)，最新稳定版PHP(7.4.3)：
   1. 方案概要
5. 高并发以及并发控制，由Swoole本身可靠的进程模型、网络模型保障；
6. 针对超时问题：每个WebSocket连接抵达Server之后，首先根据token对其进行鉴权，鉴权通过之后，在该连接上设置超时定时器，当该连接在固定时间内没有收发数据，则由Server主动断开，释放服务器资源；当客户端连接断开时，应尝试重新发起连接请求；
7. 主动推送消息问题：方案将客户端发送给Server的消息分为两大类，一类为请求，一类为订阅；前者要求Server立即给予应答，后者要求持续性关注某类消息，不要求立即回应，即当消息就绪时主动推送该消息。具体实现上，在每个WebSocket连接的路由上，注册一个Publisher来响应处理订阅类消息的逻辑，一个Controller处理请求类的消息逻辑。Publisher和Controller都分别运行在自己的协程中，当一个订阅类消息抵达Server时，该消息将交给订阅Publisher处理，在Publisher内部可以多次向客户端推送消息；此外，Controller也可以向Publisher发布订阅消息，Publisher收到后处理逻辑相似，根据情况一次或多次向客户端推送消息；
8. 平滑关闭及重启问题：Swoole主进程注册了SIGTERM信号处理逻辑，当其收到一个SIGTERM信号时，主进程将会将Server运行状态设为0，Acceptor线程、Reactor线程将会停止工作，然后Server退出(该行为的细节需要进一步确认，在退出之前是否会通知Worker进程，即我们的应用逻辑是否能够知晓，并做相应的处理；会不会出现应用逻辑认为数据已经反馈给客户端，而此时Swoole没有发送数据就已经退出)
9. 运维及监控问题：在Server启动时，设置定时器，定时上报运行状况到Redis集群(写入一个list)，由监控系统订阅该list搜集(或者由另外的脚本程序消费该list，然后上报到监控系统)；(监控系统是否有需求，要干预Server的运行，如果有，可以让Server同时订阅一个list，监控系统可以将干预指令发送至该list，同时指定Server响应消息存储位置，监控系统通过该位置获得指令执行结果)
10. Swoole进程模型

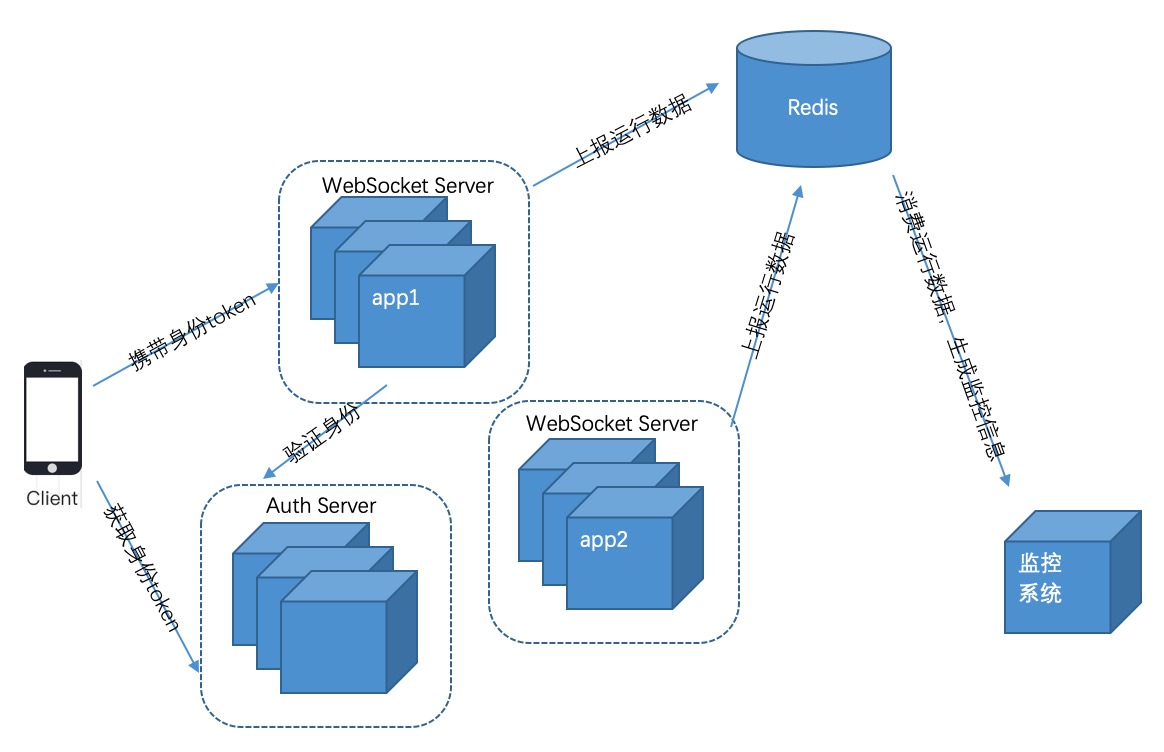
Swoole使用Reactor网络模型，较好地保障了并发性。他同时使用多线程+多进程，将连接的建立、断开，网络数据的收发，与数据的处理分离，一方面增加了并发性，同时也增加了程序的健壮性。



1. 技术设计方案
   1. 核心流程图
2. Web Socket Server启动
3. 客户端建立连接
4. 客户端发送请求类消息
5. 客户端发送订阅类消息



* 1. 服务拓扑图



客户端在申请与WebSocket Server建立连接时，应将身份token放入Header的Token字段，Server收到请求之后，首先从Header中拿到token，通过Auth Server进行校验，如果校验失败，则直接断开连接，如果校验通过则升级为WebSocket协议。

* 1. 通信协议

客户端与Web Socket Server之间使用json格式收发数据；

客户端向Server发送数据包含如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 是否必须 | 备注 |
| type | string | 是 | 请求类型，只能为request或者subscribe |
| param | string | 是 | 为json字符串加密后的结果 |

Server向客户端发送数据格式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 是否必须 | 备注 |
| code | int | 是 | 错误码，0表示正常，其它表示异常或错误 |
| message | string | 是 | 错误原因 |
| data | string | 是 | 为json字符串加密后的结果 |
| type | string | 是 | 响应类型，只能为response或者publish |

* 1. 数据结构

Web Socket Server 向Redis监控的消息list内上报的数据为json字符串，每三秒上传一次，如果三秒内某一台服务器没有上报信息，可以认为网络故障或者该Server已经宕机。

具体上报数据包含以下信息：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 是否必须 | 备注说明 |
| start\_time | int | 是 | 开始时间 |
| worker\_num | int | 是 | 工作进程数 |
| idle\_worker\_num | int | 是 | 空闲进程数 |
| tasking\_num | int | 是 | 任务进程数 |
| task\_idle\_worker\_num | int | 是 | 空闲任务进程数 |
| coroutine\_num | int | 是 | 协程数 |
| connection\_num | int | 是 | 连接数 |
| request\_count | int | 是 | 请求数 |
| accept\_count | int | 是 | 接受连接数 |
| close\_count | int | 是 | 关闭连接数 |
| worker\_request\_count | int | 是 |  |
| worker\_dispatch\_count | int | 是 |  |
| app\_name | string | 是 | 项目唯一标示 |
| hostname | string | 是 | 主机名 |
| ip | string | 是 | Server机器IP |
| port | string | 是 | Server监听端口 |
| datetime | string | 是 | 信息上报时间 |

1. 实现
   1. 调研Hyperf等框架，结合我们自己的关注点适当进行修改定制(首选)
   2. 基于社区已有组件自行DIY(正在实现demo用于演示)
2. 方案验证
3. 可能风险

由于该方案依赖Swoole的协程特性，对开发者要求增加，需要深入理解协程原理

1. 结论

暂无