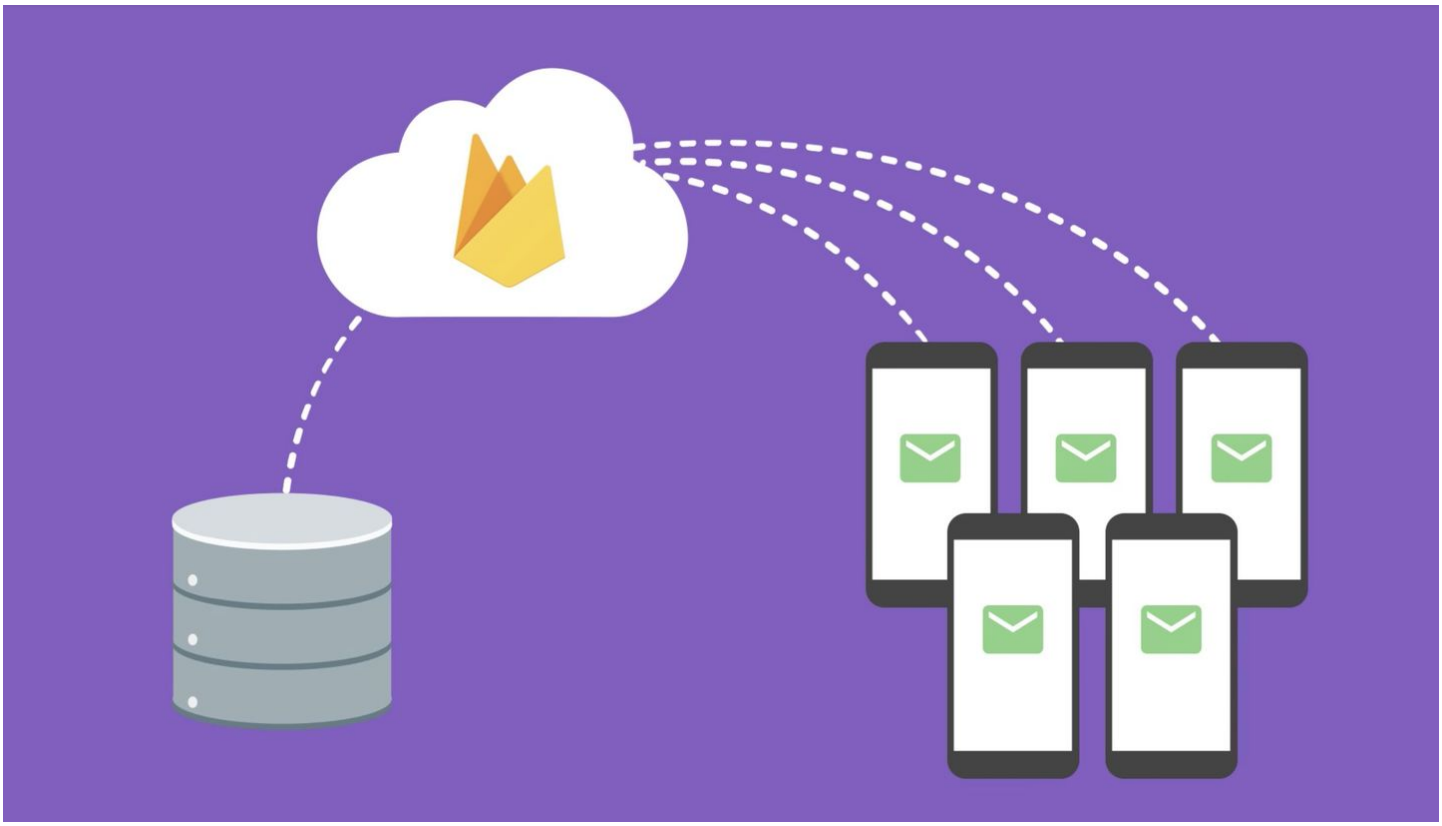
知乎 首发于 嘿嘿的学习日记



温故之消息推送系统浅谈一

忘穿秋裤 关注他

这篇文章仅仅只讨论自建消息推送系统（非IM系统）时需要考虑的问题及其对应的解决建议

目录如下

消息完整性

消息完整性与后面所述的消息的安全不同，这儿主要是说消息传输过程中粘包、半包等问题。

对于这个问题，目前比较主流的做法有以下几种

- 消息定长：例如每个报文的大小固定为1024字节，不足可以用空格补齐；
- 在每个消息尾部增加回车换行符进行分割：比如FTP协议；
- 将消息分为消息头与消息体，其中消息头为固定长度，可包含版本号，包体长度等信息，而消息体长度由所发的消息动态决定
- 使用自定义协议：比如有业务需要能在严苛的网络条件下（弱网、网络经常闪断）正常运行，这

举个使用自定义协议的例子，比如我们的业务场景命令不会超过16个，而消息内容均不会超过4字节，则这种条件下，我们可以设计成5字节，共 40 bit。其中高4位可用于表示命令，低36位表示具体消息内容

这种设计在互联网领域可能较少，但在与嵌入式相关的领域却有着广泛的应用。
我之前帮朋友设计过一个车载GPS的互联网业务系统，这里面就包含远程控制车载设备的消息系统的设计。
之所以会采用自定义二进制协议，就是希望车载设备在2G条件下（比如隧道、停车场等）也可以正常接收命令。而事实证明，这样设计是确实有用且高效的

断线重连

要使消息能推送到客户端，一条能通的链路是少不了的

就比如我们想从成都出发，去北京旅游（假设这两个地点之间只有一条路可走），如果这中间的路断了，我们就不能继续前进了，只有路修好了，我们才能继续前行

服务端与终端（消息接收端）的消息通信，也是这样，只有保证连接畅通，消息才有机会送出去
因此，我们需要在连接断开之后重新向服务器发起连接请求，以保证链路可通

连接保持

还是以上面的例子来说，成都到北京的路是否可走，这是基本条件。但可走并不代表可达，比如路上堵车十分严重，就算我们出发了，也不能说我们能够在指定的时间内到达

终端与服务器断的连接也是这样，连接上了，不代表消息能在指定的时间发送成功。在网络拥堵的情况下（一般是堵在服务器那边），消息也可能无法在指定的时间内发送到终端

因此，我们需要一种机制去保证连接的畅通性——即心跳机制。在客户端，可以定时向服务器发送一个很小的消息，即心跳包

1. 如果在指定的时间内，收到服务器端的响应，则可说明连接是畅通的
2. 如果未收到服务器端的响应，则表示这个终端与服务器之间的连接可能已经瘫痪了

当未收到服务端心跳响应时，我们可以有以下处理方式：

1. 在下一个心跳周期到来时，继续发送，如此反复。尝试指定次数（比如 6 次）之后，如果还是未能收到服务端的响应，则我们可以直接关闭连接，过会再向服务发起连接请求
2. 直接关闭连接，过会再向服务发起连接请求

关于连接的例子，更形象一点如下

- 假定成都到北京只有一条高速公路
- 那么上高速的匝道，则可以表示为服务器端或终端的发送缓存与接收缓存
- 车辆从匝道并入高速主路，则可以表示消息发送到网络链路中去的过程
- 车辆在高速公路上行驶的过程，则表示消息在网络中传输的过程
- 车辆驶下高速进入匝道，则表示消息进入终端接收缓存的过程

希望这样描述可以帮助朋友把服务端与客户端之间的连接理解得更深刻

消息可靠性

可靠性，可以从消息发送和消息接收两个方面来说

- 消息成功发送，并知道客户端已成功消费
- 消息成功接收并成功消费

但这也不能保证消息的可靠性，因此，有以下的解决方式，来尽可能地保证

消息发送

如果客户端处于离线状态，则直接将消息放入离线消息队列，待客户端上线后，服务端再推送消息或由客户端主动拉取消息

如果客户端在线（有路且路没断，但不一定畅通），为了确保客户端能尽可能地收到，发送端在仅收到客户端ACK或超过规定重发次数后才停止继续发送

- 如果消息发送失败：
  A. 如果发送超时（就像车辆堵在匝道一样），放入重发消息队列；
  B. 如果发送异常，比如Socket异常（就像车辆还未上高速就出事了一样），此时需要关闭连接，待客户端重连之后，推送再消息或由客户端主动拉取消息
- 如果收到客户端地ACK响应，则标记该消息已发送且被客户端成功消费
- 如果在指定时间内，未收到客户端地ACK，则将消息放入重发消息队列
- 对于重发消息队列的消息，如果发送指定次数或超过指定时间，仍未收到ACK，则将消息放入DLQ（Dead Letter Queue），并通知应用服务处理此消息
- 收到DLQ消息后，应用服务可以对该消息做一些处理或修复，重新放入发送队列中；也可以直接丢弃

消息接收

- 接收失败：比如客户端异常或App退出。此时需要在App重启并再次连接服务端之后，主动拉取消息或等待服务端推送
- 接收成功，但消费该消息时失败：此时无法向服务器发送ACK消息。在这种情况下，客户端可能会收到多条相同的消息，客户端去重便可
- 为了解决客户端接收了多条消息，在还未消费完时客户端挂掉丢消息的情况。
  可以在客户端加入接收消息队列（可持久化在数据库中），这样，就算客户端消费能力较弱的情况下，也能正确消费完所有消息

消息推送速度

推送速度由网络带宽（服务器和客户端），消息包的大小决定

其中，网络条件多种多样，且优化起来难度很大。因此，优化主要集中在消息包的大小上，如下

1. 协议的优化：
   A. 针对自身业务场景，定制协议；
   B. 使用MQTT协议
2. 实时监控发送性能，对于某一客户端，推送的时间普遍较长，则可以通过两种方式优化：
   A. 通过间隔指定时间（此时间由程序动态调整）推送一次消息，每个消息只包含一条消息；
   B. 消息拆分：对于一次发送多条消息的推送，拆分成单独消息按序发送
3. 减小消息体的大小：通过阈值（可后台配置）来确定消息体是否应该被压缩后再发送（比如当消息长度超过128字节后，就开启压缩）
4. 对每次推送，仅发送消息ID等重要信息

这些方式往往就可以解决大部分弱网条件下消息的传输问题了

消息推送的实时性

消息的实时性，我们可以通过以下方式保证

- 消息推送失败后，立即重发。失败指定次数之后，放入重发消息队列，由其单独调度，如果超过特定时间（比如一个消息）还未发出，则交给应用服务决定如何处理该消息
- 采用离线消息队列：即客户端如果不在线，则将消息存储于离线消息队列。客户端上线之后，从该队列取出与其相关的消息，通过单独的服务推送出去，以避免同其他调度器发生干扰

消息持久化

消息持久化，主要是针对服务端而言的，以保证在服务端宕机重启之后，还能还原重启之前的正常的那个状态。因此，主要原则是——将消息存储与消息推送分离开来。有以下建议

- 可使用阿里云 TableStore 对消息进行存储
- 可使用Redis对消息进行缓存，同时使用 Thrift、gRPC与其他服务交换消息
- 其他NoSQL数据库，但不建议使用关系型数据库，这种场景用它们不太合适
- 自写：这种方式难度最大，但也是最灵活的方式，其可以根据自身业务做到最优。但一般不建议这样做，成本太高了。市面上ActiveMQ、RobbitMQ及RocketMQ等都是采用的这种方式

需要持久化的消息主要包括以下方面

1. 待推送的消息
2. 推送失败，需要重推的消息
3. 推送失败，需要应用服务处理的消息
4. 推送失败，需要人工处理的消息
5. 离线消息
6. 从客户端接收到的，但还没来得及处理，或处理失败的消息

如果服务重启了，我们可以按以下方式处理

1. 在客户端连接之后，由客户端主动拉取
2. 拒绝客户端初次连接时的数据拉取请求，由服务端调度器推送。因为服务重启之后，会有大量的客户端请求连接，如果响应其拉取请求，容易造成网络拥堵

服务高可用

高可用主要是对外而言的，即无论服务器端发生什么情况，都可以对外提供服务

我们可以通过以下一些方式来保证

- 限制服务使用的内存大小，超过该内存之后，服务暂时停止从消息队列拿取消息
- 监控带宽使用情况。比如带宽使用超过80%时，服务暂时停止从消息队列拿取消息
- 单机上使用守护进程，守护进程每隔指定时间，向服务进程索取工作情况报告（各种性能数据）。间隔指定次数后，如果仍未获取，则认为该进程失效。此时需要停止其服务，并另开新服务来响应客户端请求
- 使用集群，负载各个客户端的请求。同时，可在服务端加入哨兵机制（可参考Redis集群的哨兵机制），以保证服务的高可用

消息安全

这里说的安全，主要指的是传输过程中的安全性，需要通过以下两方面来保证

- 消息内容加密：确保其他人不会看到
- 对消息内容进行签名：确保不会被其他人更改

> 加密、签名的顺序，只能先签名后加密。不能对加密后的消息进行签名，这没有意义

状态监控

一个好的框架或服务，少不了对各个模块状态的监控。就像一座城池的稳定，少不了巡逻兵的巡逻一样

关于这一块，一般情况都会监控以下数据。也可以根据需要监控其他数据

- 系统信息：内存使用率、CPU使用率、服务线程数量、系统网络带宽、网卡性能等

- 如果由集群，则需要从大的范围监控集群的性能：吞吐量、并发连接数、负载数据、每一台服务器的吞吐量及并发连接数等等

> 我们应该保证状态监控不会过多地影响服务的吞吐量以及并发性

最后，搭建这样的框架，可用采用那些框架/库呢？

- Netty：用于处理底层消息的通信。其解决了粘包等问题，并且它还自带很多编解码器，包括MQTT
- Redis或TableStore：用于消息的持久化
- Thrift或gRPC：用于不同服务间的通信
- Sqlite：可用于安卓端消息的存储
- Protobuf、MsgPack或Jil：如果需要序列化的话，可参考这三个库

关于自建消息推送系统，我大概就想到这么些了。欢迎关注公众号【嘿嘿的学习日记】，所有的文章，都会在公众号首发；或加个人微信【JameLee6292】共同探讨。Thank you~

发布于 2018-08-22

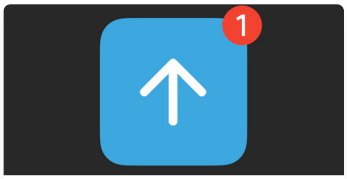
推送 (Push) 推送服务 即时通讯 (IM)

文章被以下专栏收录

嘿嘿的学习日记
温故而知新，里面是我工作学习中对各种技术知识的总结

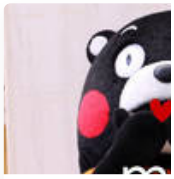
关注专栏

推荐阅读



技术干货：从零开始，教你设计一个百万级的消息推送系统

Jack ... 发表于即时通讯技...



关于消息推送的

熊孩子家的...

还没有评论

写下你的评论...