讨论课 01

1352839
饶伊文
项目 git 链接：https://github.com/BrunoQin/Open-Reuse

一、讨论课内容要求：

用户登录后始终在线，考虑低宽带/不稳定网络:
- 长连接心跳机制
- 消息不遗漏
- 消息不重复
- 消息压缩

参考业界现有解决方案分析

二、内容分析与业界参考方案：

1、长连接心跳机制

长连接:

在一个 TCP 连接上可以连续发送多个数据包，在 TCP 连接保持期间，如果没有数据包发送，需要双方发检测包以维持此连接；一般需要自己做在线维持。

一般数据库的连接用长连接，如果用短连接频繁的通信会造成 socket 错误，而且频繁的 socket 创建也是对资源的浪费。

优点：可以连续发送多个数据包，多用于操作频繁，点对点的通讯，而且连接数不能太多情况。

操作步骤：连接→数据传输→保持连接(心跳)→数据传输→保持连接(心跳)→……→关闭连接；在没有数据通信时，定时发送数据包(心跳)，以维持连接状态

短连接:

指通信双方有数据交互时，就建立一个 TCP 连接，数据发送完成后，则断开此 TCP 连接；

一般银行都使用短连接。

优点：管理起来比较简单，存在的连接都是有用的连接，不需要额外的控制手段。

操作步骤：连接→数据传输→关闭连接；在没有数据传输时直接关闭

移动通讯推送参考方案：

智能手机使用的是移动无线网络，大部分的移动无线网络运营商为了减少网关 NAT 映射表的负荷，如果一个链路有一段时间没有通信时就会删除其对应表，造成链路中断，正是这种刻意缩短空闲连接的释放超时，原本是想节省信道资源的作用，没想到让互联网的应用不得以远高于正常频率发送心跳来维护推送的长连接。这类的应用发送心跳的频率是很短的，既造成了信道资源的浪费，也造成了手机电量的快速消耗。

推送是由服务器主动向客户端发送消息，如果客户端和服务器之间不存在一个长连接那么服务器是无法来主动连接客户端的。因而推送功能都是基于长连接的基础上的。

IOS 长连接是由系统来维护的，也就是说苹果的 IOS 系统在系统级别维护了一个客户端和苹果服务器的长链接，IOS 上的所有应用上的推送都是先将消息推送到苹果的服务器然后将苹果服务器通过这个系统级别的长链接推送到手机终端上，这样的的几个好处为：1.在手机终端始终只要维护一个长连接即可，而且由于这个长链接是系统级别的不会出现被杀死而无法推送的情况。2.省电，不会出现每个应用都各自维护一个自己的长连接。3.安全，只有在苹果注册的开发者才能够进行推送，等等。

Android 的长连接是由每个应用各自维护的，但是 Google 也推出了和苹果技术架构相似的推送框架，C2DM,云端推送功能，但是由于 Google 的服务器不在中国境内，所以导致这个推送无法使用，Android 的开发者不得不自己去维护一个长链接，于是每个应用如果都 24 小时在线，那么都得各自维护一个长连接，这种电量和流量的消耗是可想而知的。虽然国内也出现了各种推送平台，但是都无法达到只维护一个长连接这种消耗的级别。

2、消息不遗漏、不重复，消息压缩

消息队列：

对任何架构或应用来说，消息队列都是一个至关重要的组件，具备的以下特性使得消息队列成为在进程或应用之间进行通信的最好形式。

解耦：在项目启动之初来预测将来项目会碰到什么需求，是极其困难的。消息队列在处理过程中间插入了一个隐含的、基于数据的接口层，两边的处理过程都要实现这一接口。这允许你独立的扩展或修改两边的处理过程，只要确保它们遵守同样的接口约束。

可恢复性：当体系的一部分组件失效，不会影响到整个系统。消息队列降低了进程间的耦合度，所以即使一个处理消息的进程挂掉，加入队列中的消息仍然可以在系统恢复后被处理。而这种允许重试或者延后处理请求的能力通常是造就一个略感不便的用户和一个沮丧透顶的用户之间的区别。

送达保证：消息队列提供的冗余机制保证了消息能被实际的处理，只要一个进程读取了该队列即可。在此基础上，IronMQ 提供了一个"只送达一次"保证。无论有多少进程在从队列中领取数据，每一个消息只能被处理一次。这之所以成为可能，是因为获取一个消息只是"预定"了这个消息，暂时把它移出了队列。除非客户端明确的表示已经处

理完了这个消息，否则这个消息会被放回队列中去，在一段可配置的时间之后可再次被处理。

异步通信：很多时候，你不想也不需要立即处理消息。消息队列提供了异步处理机制，允许你把一个消息放入队列，但并不立即处理它。你想向队列中放入多少消息就放多少，然后在你乐意的时候再去处理它们。

TCP SSL 安全套接字：

TCP socket 与 UDP socket 在传送数据时的特性：Stream (TCP) Socket 提供双向、可靠、有次序、不重复的资料传送。Datagram(UDP) Socket 虽然提供双向的通信，但没有可靠、有次序、不重复的保证，所以 UDP 传送数据可能会收到无次序、重复的资料，甚至资料在传输过程中出现遗漏。 由于 UDP Socket 在传送资料时，并不保证资料能完整地送达对方，所以绝大多数应用程序都是采用 TCP 处理 Socket，以保证资料的正确性。一般情况下 TCP Socket 的数据发送和接收是调用 send() 及 recv() 这两个函数来达成，而 UDP Socket 则是用 sendto() 及 recvfrom() 这两个函数，这两个函数调用成功发挥发送或接收的资料的长度，否则返回 SOCKET_ERROR。

安全套接字层 (SSL) 是一种协议，支持服务通过网络进行通信而不损害安全性。

SSL 由两层组成，分别是握手协议层和记录协议层。握手协议建立在记录协议之上，此外，还有警告协议、更改密码说明协议和应用数据协议等对话协议和管理提供支持的子协议。SSL 发出消息是将数据分为可管理的块、压缩、使用 MAC 和加密并发出加密的结果。接受消息需要解密、验证、解压和重组，再把结果发往更高一层的客户。

为了避免网络中传输的数据被非法篡改，SSL 利用基于 MD5 或 SHA 的 MAC 算法来保证消息的完整性。MAC 算法是在密钥参与下的数据摘要算法，能将密钥和任意长度的数据转换为固定长度的数据。利用 MAC 算法验证消息完整性的过程如图 2 所示。发送者在密钥的参与下，利用 MAC 算法计算出消息的 MAC 值，并将其加在消息之后发送给接收者。接收者利用同样的密钥和 MAC 算法计算出消息的 MAC 值，并与接收到的 MAC 值比较。如果二者相同，则报文没有改变；否则，报文在传输过程中被修改，接收者将丢弃该报文。

分布式高可靠消息中间件－Hippo：

Hippo 系统存在四种角色，分别为生产者（producer）、 消费者（consumer）、存储层（broker）、中心控制节点（controller）

消息存储可靠性：WAL+持久化；数据存储多副本；存储节点自动 failover；

消息传输可靠性：ACK 机制；数据 CRC 校验；

消息投递可靠性：producer->broker 数据存储后才返回成功确认;broker->consumer 数据处理完成后需进行确认

服务（Qos）级别：不能丢消息；At-least-once 可能会有重复*；极端情况下通过客户端进行数据去重。

三、本组实现分析：

1、长连接心跳机制

我们的项目通过 Netty 实现服务端与客户端的长连接通讯，及心跳检测。

Netty 服务端通过 Map 保存所有连接上来的客户端 SocketChannel，客户端的 Id 作为 Map 的 key。每次服务器端如果要向某个客户端发送消息，只需根据 ClientId 取出对应的 SocketChannel，往里面写入 message 即可。心跳检测通过 IdleEvent 事件，定时向服务端放

送 Ping 消息，检测 SocketChannel 是否终断。

Server 端维护了三个 map：Global Count Map（记录全局的登录注册消息数量）, Global Socket Conn Map（管理连接到 server 的客户端的数量）, Global Rout Map（控制 message 被分配到某个 route）。

2、消息不遗漏、不重复，消息压缩

我们的项目通过消息队列对消息进行处理，基本可实现消息的完整性、不遗漏，并通过 JSON 进行对消息的序列化处理和解析，使用固定的消息格式对消息进行封装压缩。Queue 与 Delayqueue 的使用把生产者和消费者的代码相互解耦合，满足项目需求与解耦程度。

阻塞队列（Blocking queue）提供了可阻塞的 put 和 take 方法，它们与可定时的 offer 和 poll 是等价的。如果 Queue 已经满了，put 方法会被阻塞直到有空间可用；如果 Queue 是空的，那么 take 方法会被阻塞，直到有元素可用。Queue 的长度可以有限，也可以无限；无限的 Queue 永远不会充满，所以它的 put 方法永远不会阻塞。DelayQueue 是一个 BlockingQueue，其特化的参数是 Delayed。Delayed 扩展了 Comparable 接口，比较的基准为延时的时间值，Delayed 接口的实现类 getDelay 的返回值应为固定值（final）。DelayQueue 内部是使用 PriorityQueue 实现的。

四、参考资料：

1. socket 中的短连接与长连接,心跳包示例详解
http://blog.chinaunix.net/uid-26000296-id-3758651.html
2. Netty 实现服务端客户端长连接通讯及心跳检测
http://www.open-open.com/lib/view/open1428890187783.html
3. 互联网推送服务原理：长连接+心跳机制(MQTT 协议)
http://blog.csdn.net/clh604/article/details/20167263
4. 基于 Winsock API 的 VC 网络编程实战
http://blog.chinaunix.net/uid-20672559-id-1579693.html
5. 安全套接字层
http://baike.baidu.com/view/6241732.htm
6. 安全套接层 SSL 协议简介
http://www.360doc.com/content/10/0401/23/633992_21238689.shtml
7. SSL 技术白皮书
http://www.h3c.com.cn/Products___Technology/Technology/Security_Encrypt/Other_technology/Technology_book/200812/622834_30003_0.htm#_Toc212542719
8. 分布式高可靠消息中间件－Hippo
http://www.wtoutiao.com/p/j2c8C6.html
9. 阻塞队列和生产者-消费者模式、DelayQueue
http://my.oschina.net/coolfire368/blog/295637
10. 精巧好用的 DelayQueue
http://www.cnblogs.com/jobs/archive/2007/04/27/730255.html
11. 使用消息队列的 10 个理由
http://www.oschina.net/translate/top-10-uses-for-message-queue