讨论课1

> 2016.4.06
>
> Created by Fowafolo

讨论内容

> 用户登录后始终在线，考虑低带宽、不稳定网络：
>
> 1. 长连接心跳机制
> 2. 消息不遗漏
> 3. 消息不重复
> 4. 消息压缩
>
> 可以参考业界现有的解决方案

虽然我们的项目使用了ActiveMQ——内部实现了讨论内容中的部分要求，但是我还是首先查阅了一些资料，对相关概念和业界处理办法进行了了解。

学习过程

查阅资料

我查阅了 `长连接心跳机制` 这一似懂非懂的概念。

长连接心跳机制

长连接定义

长连接多用于操作频繁，点对点的通讯，而且连接数不能太多情况。

长连接的例子

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。

心跳机制的定义

根据资料，所谓“心跳”就是定时发送一个自定义的结构体（心跳包或心跳帧），让对方知道自己“在线”。 以确保链接的有效性。所谓的心跳包就是客户端定时发送简单的信息给服务器端告诉它我还在而已。代码就是每隔几分钟发送一个固定信息给服务端，服务端收到后回复一个固定信息如果服务端几分钟内没有收到客户端信息则视客户端断开。

心跳机制的作用

比如有些通信软件长时间不使用，要想知道它的状态是在线还是离线就需要心跳包，定时发包收包。发包方：可以是客户也可以是服务端，看哪边实现方便合理。一般是客户端。服务器也可以定时轮询发心跳下去。心跳包之所以叫心跳包是因为：它像心跳一样每隔固定时间发一次，以此来告诉服务器，这个客户端还活着。事实上这是为了保持长连接，至于这个包的内容，是没有什么特别规定的，不过一般都是很小的包，或者只包含包头的一个空包。

心跳机制的实现实例

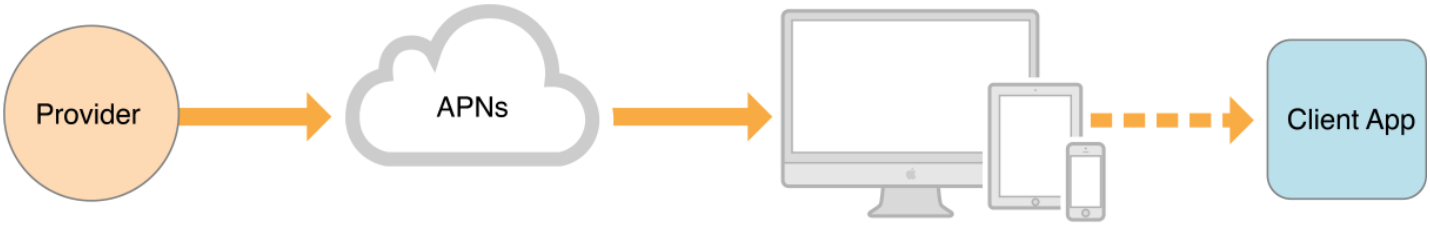
在TCP的机制里面，本身是存在有心跳包的机制的，也就是TCP的选项。系统默认是设置的是2小时的心跳频率。但是它检查不到机器断电、网线拔出、防火墙这些断线。而且逻辑层处理断线可能也不是那么好处理。一般，如果只是用于保活还是可以的。心跳包一般来说都是在逻辑层发送空的包来实现的。下一个定时器，在一定时间间隔下发送一个空包给客户端，然后客户端反馈一个同样的空包回来，服务器如果在一定时间内收不到客户端发送过来的反馈包，那就只有认定说掉线了。只需要send或者recv一下，如果结果为零，则为掉线。

思考

查阅了这点之后，我想到了手机上每日收到的APP的推送应该是维持了一个长连接，并查阅了相关资料。

> 据了解，iOS系统和Android系统的推送服务机制并不一样，由于我的手机系统是iOS，所以查阅了苹果的 `APNS` （苹果推送通知服务）资料。

下面的图就是iOS系统远程推送大致的流程图，其中Provider是指某个iOS app的Push服务器



首先，APNS会对用户进行物理连接认证，和设备令牌认证（简言之就类似本项目的身份验证）；

然后，将服务器的信息接收并且保存在APNS当中，APNS从其中注册的列表中查找该IOS设备（设备可以为iPhone、iPad、iPod Touch，版本是iOS3.0及以上）并将信息发送到该设备； 最后，设备接收到数据信息给相应的APP，并按照设定弹出Push信息。

这一过程如果采用短连接Socket，将会消耗极大的资源，因此必须采用长连接方式。

另外，大部分资料显示，APNS的心跳间隔是15分钟，即15分钟进行一次心跳检查。

本项目解决方案

在本项目中，我们使用了ActiveMQ作为消息中间件，客户端和服务器之间建立了一个连接用来发送专用通道的消息。而ActiveMQ内部实现了断线重连机制，所以我们要做的心跳检测其实基本上可以省略了。

> 想要MQ断开后自动重连，很简单，只用在 tcp://localhost:61616前面加上 failover:

这里简介一下ActiveMQ内部实现心跳的做法。

根据Apache官方指南（链接附在参考资料），ActiveMQ通过 `InactivityMonitor` 来进行连接的心跳检测。

> The ActiveMQ InactivityMonitor is an active thread that checks the connection is still active and if it suspects the connection is not functioning correctly, it closes the connection.

每个连接都有两个相关的 `InactivityMonitors` ，在连接两端各有一个。在指定时间段内， `InactivityMonitors` 将会收到数据来keep alive，如果超过了这个时间段，则会自动断开。

但是如果服务器宕机，client并不能发送消息。解决方案是：

ActiveMQ提供了消息传输监听（transportListener），可以对ActiveMQConnectionFactory添加一个Activemq的消息传输监听，该监听实现 Activemq的TransportListener接口。该接口实现的监听方法有onCommand（），onException（）， transportResumed () ，transportInterupted()等监听方法。拥有这些方法就足以实时感知ActiveMQ服务器的状态了，当发现服务器无法连接时，就采取相应措施，如把消息存储在本地，当服务器恢复时再进行发送。

```java
public class ActiveMQTransportListener implements TransportListener{

    protected final Logger logger = LoggerFactory.getLogger(ActiveMQTransportListener.

    /**
     * 对消息传输命令进行监控
     * @param command
     */
    @Override
    public void onCommand(Object o) {
    }

    /**
     * 对监控到的异常进行触发
     * @param error
     */
    @Override
    public void onException(IOException error) {
        logger.error("onException -> 消息服务器连接错误......");
    }

    /**
     * 当failover时触发
     */
    @Override
    public void transportInterupted() {
        logger.error("transportInterupted -> 消息服务器连接发生中断......");
        //这里就可以状态进行标识了
    }

    /**
     * 监控到failover恢复后进行触发
     */
    @Override
    public void transportResumed() {
        logger.info("transportResumed -> 消息服务器连接已恢复......");
        //这里就可以进行状态标识了
    }
}
```

所以我提出的最终解决方案是：

1. 连接建立好以后，非宕机、断线情况，使用ActiveMQ内部实现的断线重连机制。
2. 断线宕机情况：在客户端和服务器建立connect之后，每次相互发送消息之前先检测connect是否有效，如果出现一方断线、或者ActiveMQ服务器宕机的情况，发送消息方都会得到通知。

3. 如果是消息发送方断开连接，则请求重连。如果请求失败，则是ActiveMQ宕机或者本地网络故障，或者是另一端同时断开了连接。所以要设置一个最大尝试次数。

消息方面

项目要求做到消息压缩、消息不遗漏、消息不重复。

根据项目，我认为考虑这三者的顺序应该是，首先做到不遗漏，其次是不重复，最后是压缩。

消息不遗漏

在本项目中，登录成功后的client每次发送一条消息，都要经过server端license的验证，合法消息才被发到topic中。保证不遗漏，只需要做到两方面：

1. client发的消息都能到达server：

   client发之前要验证queue是否可用，如果已经断开则会提示用户，如果可用，则可以发送到指定与server建立的管道。几乎不可能断掉，因此这方面在之前的工作中已经做到。

2. server转发的合法消息所有subscriber都能收到：

   ActiveMQ的topic连接中有参数可以设置，能够保证监听topic的client都收到合法消息。

ActiveMQ消息丢失解决方案

（1）、没发送到broker上就丢失了：在发送broker之前，将消息本地化存储。收到发送成功的反馈后移除本地文件。实时检索本地文件，发现超过一定时间没有发送直接重新发送。

（2）、在broker上丢失：将broker的持久化改成数据库，将数据库做主从处理

（3）、从broker到订阅者丢失：在没有收到订阅者成功接收的时候，不移除broker的持久化数据，其实这个貌似有事务支持

（4）、集群的failover失败：建立一个测试队列，定时发送消息测试每个节点是否服务正常。只能通过之类心跳类监控手段来确认服务器是否正常。

消息不重复

上面已经提到，消息是被server转到特定topic，而MQ内部实现了一个消息被一个subscriber接收后不被重复接收的机制。因此我们的项目中不用再次重写。

ActiveMQ中消息重复的可能性

1. AUTO_ACKNOWLEDGE : 自动确认,这就意味着消息的确认时机将有consumer择机确认."择机确认"似乎充满了不确定性,这也意味着,开发者必须明确知道"择机确认"的具体时机,否则将有可能导致消息的丢

失,或者消息的重复接收.

2. 当我们使用messageListener方式消费消息时，通常建议在onMessage方法中使用try-catch,这样可以在处理消息出错时记录一些信息，而不是让consumer不断去重发消息；如果你没有使用try-catch,就有可能会因为异常而导致消息重复接收的问题,需要注意你的onMessage方法中逻辑是否能够兼容对重复消息的判断。

消息压缩

消息主体包含了消息的核心数据。

JMS 定义了5中消息类型： TextMessage、MapMessage、BytesMessage、StreamMessage和ObjectMessage

其中BytesMessage（字节消息）是指将字节流存放在消息主体中。刚好适合于：必须压缩发送的大量数据。压缩前需要与现有消息格式保持一致。

另外，ActiveMQ官方提供了设置允许消息压缩的属性：

Option Name	Default Value	Default Value
useCompression	false	Enables the use of compression of the message bodies

参考资料

参考资料名称	链接地址
如何实现支持数亿用户的长连消息系统	http://chuansong.me/n/1641640
ActiveMQ InactivityMonitor	http://activemq.apache.org/activemq-inactivitymonitor.html
Connection Configuration URI	http://activemq.apache.org/connection-configuration-uri.html