软件复用第一次讨论课方案

1252874 陈薇伊

一、长连接心跳机制

1.背景：

心跳机制是定时发送一个自定义的结构体(心跳包)，让对方知道自己还活着，以确保连接的有效性的机制。但是，目前移动通信网络中由于用户众多、资源稀缺，每个用户都是动态占用资源，比如IP地址以及无线信道。每次发送心跳包，都需要移动通信网络为用户分配资源，分配的过程体现在信令的发送和接收上。一次心跳包的发送过程，牵涉的信令多达几十条。所以，在基本的心跳机制上，考虑不同通信网络的情况，能优化心跳。

2.方案设计：

目的：在尽量不影响用户收消息及时性的前提下，根据网络类型自适应的找出保活信令TCP连接的尽可能大的心跳间隔，从而达到减少心跳引起的空中信道资源消耗，减少心跳Server的负载，以及减少部分因心跳引起的耗电。

方案：自适应心跳间隔优化

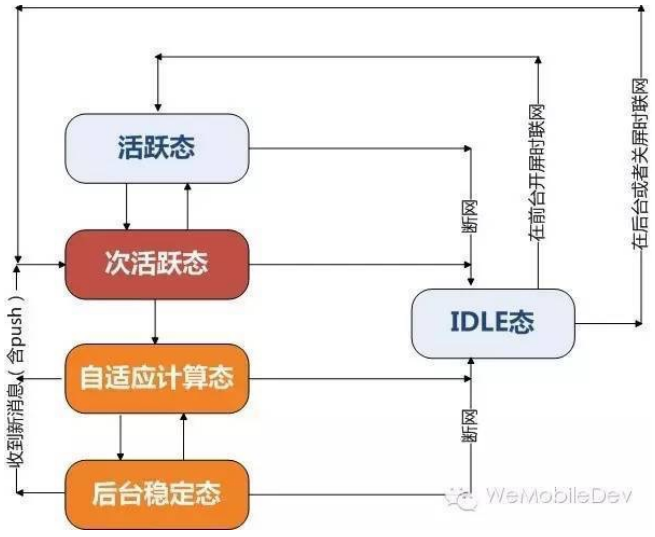
1、前后台区分处理：

为了保证收消息及时性的体验，当处于前台活跃状态时，使用固定心跳。进入后台（或者前台关屏）时，先用几次最小心跳维持长链接。然后进入后台自适应心跳计算。这样做的目的是尽量选择用户不活跃的时间段，来减少心跳计算可能产生的消息不及时收取影响。

2、后台自适应心跳选择区间：

可根据自身产品的特点选择合适的心跳范围。

3.状态转换图如下：

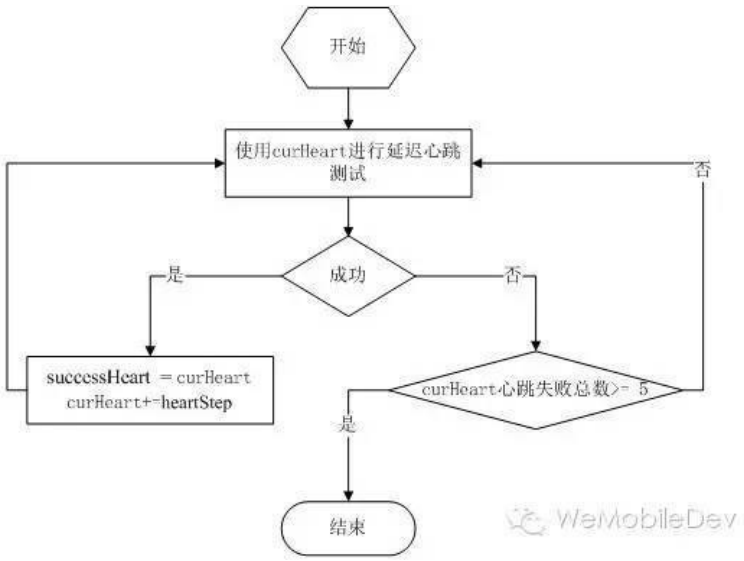


4.按网络类型区分计算：

因为每个网络的NAT时间可能不一致。所以需要区分计算，数据网络按subType做关键字，WIFI按WIFI名做关键字。

对稳定的网络，因为NAT老化时间的存在，在自适应计算态的时候，暂设计以下步骤在当前心跳区间逼近出最大可用的心跳。

变量说明：



[MinHeart，MaxHeart]——心跳可选区间。

successHeart——当前成功心跳，初始为MinHeart

curHeart——当前心跳初始值为successHeart

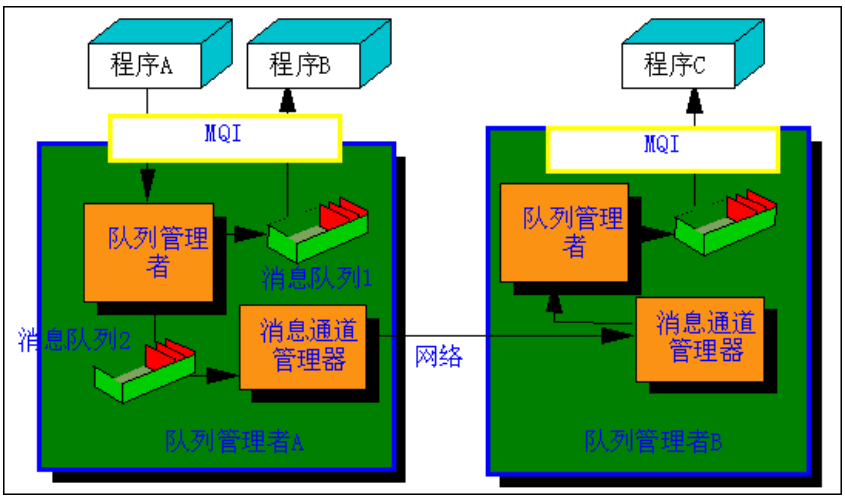
heartStep——心跳增加步长

successStep——稳定期后的探测步长

经过该流程，会找到必然使心跳失败的curHeart（或者MaxHeart），为了保险起见，可以选择比前一个成功值稍微小一点的值作为后台稳定期的心跳间隔。但是，影响网络测试的因素太多，为了尽量保证测试结果的可靠性，使用延迟心跳测试法。在重新建立TCP连接后，先使用短心跳连续成功三次，才认为网络相对稳定，可以使用curHeart进行一次心跳测试。在没有达到稳定网络环境时，会一直使用固定短心跳直到满足三次连续短心跳成功。使用延迟心跳测试的好处是，可以剔除偶然失败，和网络变化较大的情况（如地铁），使测试结果相对可靠（五次延迟测试确定结论）。同时在网络波动较大的情况，使用短心跳，保证收取消息相对及时。

二、消息不遗漏

消息队列技术是分布式应用间交换信息的一种技术。消息队列可驻留在内存或磁盘上,队列存储消息直到它们被应用程序读走。通过消息队列，应用程序可独立地执行—它们不需要知道彼此的位置、或在继续执行前不需要等待接收程序接收此消息。



如图，首先来看本地通讯的情况，应用程序A和应用程序B运行于同一系统A，它们之间可以借助消息队列技术进行彼此的通讯：应用程序A向队列1发送一条信息，而当应用程序B需要时就可以得到该信息。其次是远程通讯的情况，如果信息传输的目标改为在系统B上的应用程序C，这种变化不会对应用程序A产生影响，应用程序A向队列2发送一条信息，系统A的MQ发现Q2所指向的目的队列实际上位于系统B，它将信息放到本地的一个特殊队列－传输队列(Transmission Queue)。我们建立一条从系统A到系统B的消息通道，消息通道代理将从传输队列中读取消息，并传递这条信息到系统B，然后等待确认。只有MQ接到系统B成功收到信息的确认之后，它才从传输队列中真正将该信息删除。如果通讯线路不通，或系统B不在运行，信息会留在传输队列中，直到被成功地传送到目的地。这是MQ最基本而最重要的技术--确保信息传输，并且是一次且仅一次(once-and-only-once)的传递。

三、消息不重复

由于客户端不能避免重复收到信息，那么，可以选择给信息做上标记，对标记进行维护，到客户端接收到信息后，先查看这条信息的标记是否是标记为已处理，若是，则直接丢弃信息，反之则推送到客户端，同时将该标记标记为已处理。

四、消息压缩

压缩就是一个消除冗余的过程，相当于用一种更精简的形式，表达相同的内容。可以想象，压缩过一次以后，文件中的重复字符串将大幅减少。好的压缩算法，可以将冗余降到最低，以至于再也没有办法进一步压缩。

Huffman于1952年提出一种编码方法，该方法完全依据字符出现概率来构造异字头的平均长度最短的码字，是可变字长编码(VLC)的一种，该方法又称为霍夫曼编码

在计算机信息处理中，“哈夫曼编码”是一种一致性编码法（又称“熵编码法”），用于数据的无损耗压缩。这一术语是指使用一张特殊的编码表将源字符（例如某文件中的一个符号）进行编码。这张编码表的特殊之处在于，它是根据每一个源字符出现的估算概率而建立起来的（出现概率高的字符使用较短的编码，反之出现概率低的则使用较长的编码，这便使编码之后的字符串的平均期望长度降低，从而达到无损压缩数据的目的）。这种方法是由David.A.Huffman发展起来的。 例如，在英文中，e的出现概率很高，而z的出现概率则最低。当利用哈夫曼编码对一篇英文进行压缩时，e极有可能用一个位(bit)来表示，而z则可能花去25个位（不是26）。用普通的表示方法时，每个英文字母均占用一个字节（byte），即8个位。二者相比，e使用了一般编码的1/8的长度，z则使用了3倍多。若能实现对于英文中各个字母出现概率的较准确的估算，就可以大幅度提高无损压缩的比例