讨论课内容

1352961 秦博

基础内容

长连接与短连接

长连接: 指在一个 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。

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。 它的优点是：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 。比如 http 的，只是连接、请求、关闭，过程时间较短,服务器若是一段时间内没有收到请求即可关闭连接。

其实长连接是相对于通常的短连接而说的，也就是长时间保持客户端与服务端的连接状态。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。每个 TCP 连接都需要三步握手，这需要时间，如果每个操作都是先连接，再操作的话那么处理速度会降低很多，所以每个操作完后都不断开，下次处理时直接发送数据包就可以了，不用建立 TCP 连接。

长连接应用场景

长连接应用的场景之一就是推送。iOS 系统中所有的应用都维护同一个长连接，所以成本比较低。但是 Android 系统中，应用要维护各自的长连接，所以要解决这个问题比较复杂。几种常见的解决方案有：

1）轮询(Pull)方式：应用程序应当阶段性的与服务器进行连接并查询是否有新的消息到达，你必须自己实现与服务器之间的通信，例如消息排队等。而且你还要考虑轮询的频率，如果太慢可能导致某些消息的延迟，如果太快，则会大量消耗网络带宽和电池。

2）SMS(Push)方式：在 Android 平台上，你可以通过拦截 SMS 消息并且解析消息内容来了解服务器的意图，并获取其显示内容进行处理。这是一个不错的想法。这个方案的好处是，可以实现完全的实时操作。但是问题是这个方案的成本相对比较高，我们需要向移动公司缴纳相应的费用。我们目前很难找到免费的短消息发送网关来实现这种方案。

3）持久连接(Push)方式：这个方案可以解决由轮询带来的性能问题，但是还是会消耗手机的电池。IOS 平台的推送服务之所以工作的很好，是因为每一台手机仅仅保持一个与服务器之间的连接，事实上 C2DM 也是这么工作的。不过刚才也讲了，这个方案存在着很多的不足之处，就是我们很难在手机上实现一个可靠

的服务，目前也无法与 IOS 平台的推送功能相比。

对于 Android 系统，我们可以通过消息队列来模拟长连接。典型的解决方案就是 MQTT。

MQTT 协议是为大量计算能力有限，且工作在低带宽、不可靠的网络的远程传感器和控制设备通讯而设计的协议，它具有以下主要的几项特性：

1、使用发布/订阅消息模式，提供一对多的消息发布，解除应用程序耦合；

2、对负载内容屏蔽的消息传输；

3、使用 TCP/IP 提供网络连接；

4、有三种消息发布服务质量：

1）“至多一次”，消息发布完全依赖底层 TCP/IP 网络。会发生消息丢失或重复。这一级别可用于如下情况，环境传感器数据，丢失一次读记录无所谓，因为不久后还会有第二次发送。

2）“至少一次”，确保消息到达，但消息重复可能会发生。

3）“只有一次”，确保消息到达一次。这一级别可用于如下情况，在计费系统中，消息重复或丢失会导致不正确的结果。

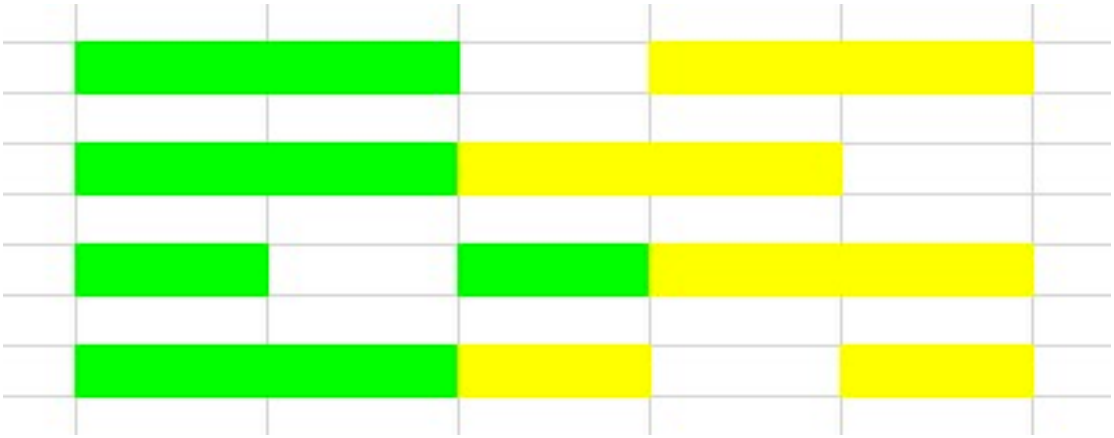
5、小型传输，开销很小（固定长度的头部是 2 字节），协议交换最小化，以降低网络流量；

消息发送

对于消息的发送，首先将消息转化成二进制流，客户端进行发送，服务器进行接收。其中包含缓冲区，以防止消息过长，分次发送，接收不完整。这个缓冲区在网络不稳定时会发挥作用。

对于 NIO 的 SocketChannel，在非阻塞模式下，它会直接返回连接结果，如果没有连接成功，也没有发生 IO 异常，则需要将 SocketChannel 注册到 Selector 上监听连接结果。所以，异步连接的超时无法在 API 层面直接设置，而是需要通过定时器来主动监测。创建连接超时定时任务之后，会由 NioEventLoop 负责执行。如果已经连接超时，但是服务端仍然没有返回 TCP 握手应答，则关闭连接。

TCP 是个"流"协议，所谓流，就是没有界限没有分割的一串数据。TCP 会根据缓冲区的实际情况进行包划分，一个完整的包可能会拆分成多个包进行发送，也用可能把多个小包封装成一个大的数据包发送。这就是 TCP 粘包/拆包。



ProtocolBuffer 是一种语言无关、平台无关、扩展性好的用于通信协议、数据存储的结构化数据串行化方法。用于结构化数据串行化的灵活、高效、自动的方法，有如 XML，不过它更小、更快、也更简单。你可以定义自己的数据结构，然后使用代码生成器生成的代码来读写这个数据结构。你甚至可以在无需重新部署程序的情况下更新数据结构。

随着系统的演化，protocolBuffer 有一些其他的功能：

1）自动生成编码和解码代码，而无需自己编写解析器。

2）除了用于简短的 RPC(Remote Procedure Call)请求，人们使用 ProtocolBuffer 来做数据存储格式(例如 BitTable)。

3）RPC 服务器接口可以作为 .proto 文件来描述，而通过 ProtocolBuffer 的编译器生成存根(stub)类供用户实现服务器接口。