双通道机制详解

目录

问题由来

早期的通讯方式，都是一方发送，另一方接收，网络上的串流，就是一个流向，这是一种将网络程序通讯简化出来的机制。这是种机制，就是让编程变得直观，容易理解。我们常用的 ftp，http，telnet 都是这种机制。我将这种机制，称之为单向通讯。

因为单向通讯，这种概念，无法满足交互的要求，于是，才有了双通道的概念。

双通道的概念模型其实就是早期的电话线，有两个铜缆负责传输信号，一个是发送，一个是接收，两个通道均有各自的串流方向，这正是交互所需要的。我们实现交互的通讯功能时，双通道的概念，就是首选，我们对服务器，创建两个连接，一个负责接收，一个负责发送，这样干，一针见血，直接就解决了交互通讯这种机制上的问题。

为什么建立两个连接？

因为两个连接，可以避开复杂的数据结构实现，注意力更容易集中于实现通讯协议本身。

试想：如果你坚持要在一个连接中以双工方式实现双通道，在处理接收数据上，就需要区分带反馈的请求，和对方主动发出的请求，这些差异，用文字不太好形容，实际做起来，你需要做抽象数据处理，相当于 zs 中 p2pVM 的通讯数据打包模块，毫无疑问，这是一个小坑。处于简而美，所以 zs 选择了两个连接来解决交互问题。

假如，你希望一个连接，就能解决交互问题，请直接使用 zs 中 p2pVM 的解决方案，它一个连接，可以带起 100 万的虚拟隧道，我们说的双通道，在 p2pVM 其实也是两个虚拟隧道，只不过 p2pVM 把这两个抽象的隧道用一个连接来实现了。（注意：p2pVM 的存在并不是解决双通道以单连接工作问题的，p2pVM 的朝向是模拟整个通讯协议栈）

ZS 中的双通道机制

在 ZS 中，凡是带有 DoubleTunnel 字样的库和类，都使用了双通道机制，在里面总是会有 Recv+send 这种概念。

DoubleTunnel 的 Serve 会有 Receive+Send 两个侦听。DoubleTunnel 的 Client 会有 Send+Receive 两个连接

因为用侦听，连接，这些概念来定义开发工艺，非常容易搞混，所以 ZS 把侦听和连接，合并在了一起，叫做通道，在 ZS 中对通道的命名有(IO,Tunnel)两种，它们分辨代表两个级别的开发模式。
IO 表示底层的通讯通道，提供给 ZS 内核使用，属于原型级的定义。
Tunnel 表示高级的通道，提供给 ZS 的调用程序使用，用 Tunnel 的命名在设计上是倾向于对 IO 的管理作用，属于工艺级的定义。

双通道在 Tunnel 的使用层面需要区分

任何时候，一个连接，ZS 内核不会区分它是服务器还是客户端，ZS 内核只会认为，这是一个 IO 通道，在这一层，没有工艺，都是原始数据处理。对这个通道的管理，那就交给 Tunnel 这一层去干，开发工艺都在 Tunnel 层面。而 Tunnel，它就会区分这是 Recv 还是 Send。

对于客户端而言：

客户端在任何时间，主动发起的数据，包括发送带有反馈的请求，都要使用客户端的 Send Tunnel

客户端的 Receive Tunnel 只能是由服务器主动发起一个请求过来才会有用，不可以作死式的使用客户端的 Receive Tunnel 去主动发起数据请求

对服务器而言：概念与客户端相同，我们在使用时，需要区分 Tunnel 的对应关系

- Client 的 Send Tunnel，对应的是 Server 的 Receive Tunnel，我们在使用许多高级 DoubleTunnel 框架的 Login 时，实际上就是从这个 Tunnel 发出去的
- Client 的 Receive Tunnel，对应的是 Server 的 Send Tunnel

ZS 中的 IO 永远和连接及侦听无关

IO 不管你是服务器还是客户端，甚至是串口，USB，蓝牙，https2.0 中的 websocket，只要有连接，就会诞生一个 IO，在 ZS 中，叫做 TpeerIO

换句话说，如果有 10 个连接，就会有个 10 个 IO

IO 是怎样把设备通讯变成设备交互的？

每个 IO 中，都可以使用 p2pVM，这是一种虚拟机，它模拟了整个通讯协议栈，在协议栈里面，我们又可以做 Tunnel，在 Tunnel 里面又可以有更多的 IO，依次类推下去，你不用担心开发工艺，只要是数据传输，IO 都有虚化通讯设备的能力。

2018-10-19
By qq600585