为什么要由TCP协议负责数据传输的可靠性？

原创：车小胖谈网络 车小胖谈网络 前天

如果TCP不负责数据传输的可靠性，让谁来负责呢？

这不是抬杠，真理是不怕辩论的，从某种角度来说，真理是越辩越明朗！

按照TCP/IP模型五层模型，从上层到下层一个个看过来，看谁最帅，最适合做数据传输的可靠性！

应用层

应用程序，直接和用户的数据打交道，做数据传输的可靠性毫无压力。不就是TCP那个套路吗？咱应用程序也学得会，不就是那个确认机制吗？

我方发数据，你方确认收到数据；你方发数据，我方确认收到数据！

多么朴素无华的道理！

还有如果丢包，超时重传。如果超时重传多次没有成功，Reset双方的通信会话。

还有，为了不把接收方的接收缓冲区（仓库）占满并溢出，需要对方实时通告对方仓库的剩余空间，这个就是“Window”。

如果每次发数据，对方能及时确认，那就越发越快，直到占满对方仓库为止。

如果发数据，对方不能及时确认，那就悠着点发，咱别给互联网添堵。

实现完以上代码，对照TCP代码一看，我去，这不就是TCP代码吗？

通过上文的讨论，得出第一个观点：

第一个观点：无论哪层来实现数据的可靠传输，实现代码和**TCP**代码应该是高度相似的！

假设这段伪**TCP**代码10K 行，有100K字节大小。如果用户开N个浏览器窗口，同时有N个伪**TCP**代码在运行，占用的内存空间是**N*100 K**字节。其它的应用程序，也还是要实现自己的伪TCP代码。

读者会说，直接把浏览器的伪**TCP**代码拿来用就好了吗？其它程序为什么还要从无到有实现伪**TCP**代码？

浏览器的伪**TCP**代码为何要给你用？哦，你也是Microsoft操作系统旗下的产品，Okay，拿去用吧，不谢！

当新的应用程序每运行一个实例，内存空间就会多一块100K字节的伪**TCP**代码。

如果这段“伪**TCP**代码”有什么bug需要修复，所有使用“伪**TCP**代码”应用程序都要修改，这是多么的悲惨的场景啊！一大批程序员撅着屁股在那儿狂改bug。。。

那些独立开发应用程序的小公司，和人家Microsoft又不认识，人家的伪**TCP**代码也不会给你用，那只有硬着头皮写自己的伪**TCP**代码。限于开发水平，一大群程序员花了三个月的时间写是写出来了，但是错误百出，需要经常修Bug。。。

统计表明，每100行代码平均会有1个bug出现，10000行代码里至少埋藏着100个Bug！

小公司的老总硬着头皮联系Microsoft技术支持，能否把贵司的伪**TCP**代码通过接口函数开放出来？

很快，Microsoft 将伪**TCP**代码，用接口函数开放了出来，包括但不限于：

CreateSocket()
Connect()
Send()
Receive()
Close()
Shutdown()

用户看不到这些函数的内部实现，但是只要在应用程序里使用这些接口函数，如同使用伪**TCP**代码。

为了最大限度降低伪**TCP**代码的数量，操作系统将伪**TCP**代码从浏览器里剥离，伪**TCP**代码被集成到操作系统内核，所有应用程序调用接口函数，就如同使用伪**TCP**代码本身！

这样，整个操作系统就有且仅有一个伪**TCP**代码在运行，即使有N多个程序在同时运行。

当前操作系统就是这么来实现的，**TCP**代码，可以供所有的应用程序共享使用，提高代码的重用，避免代码的无谓重复！

这段神奇的代码名字是“**TCP**代码”，用于完全实现TCP/IP协议栈的**TCP**协议！

第二个论点：**TCP**的存在，是为了避免相同的代码出现不同的地方，从而提高代码的使用效率！

网络层

一个IP报文从源主机到达目的主机的路径上，会经过N多个路由器。为了让IP层来实现可靠传输，需要和相邻的路由器建立可靠传输，问题是IP协议头也没有什么字段可以保证可靠传输的，比如如何建立连接？如何字节流编号？如何确认？这些还不是致命的！

要命的是，路由器收到一个IP报文，并不知道它的邻居是谁，因为IP报文只携带源主机、目的主机的IP，却独独没有邻居的IP地址，和谁建立可靠连接？

有读者会说，通过以太帧头的源MAC可以知道邻居路由器，然后根据ARP表可以反向解析得到邻居的IP地址，自然就可以建立可靠连接了。

好吧，还需要扩展IP头协议字段以实现可靠传输！等扩展完才发现，扩展出来的协议字段和TCP没有什么两样！

问题又来了，核心路由器一秒钟高达几千万、甚至上亿次**IP**报文的转发，这些完全依靠硬件转发。

一旦要IP层建立可靠连接，维护连接状态、以及处理IP报文，就不能依靠硬件了，硬件处理不了那么复杂的逻辑。而是需要软件（**CPU**）来实现，而软件压根完成不了一秒几千万次的转发！这样就会严重影响路由器的转发速率！

至于数据链路层，就更不提了，数据链路层越简单越好，数据的转发效率才会高！

通过以上论述，才发现操作系统选择TCP来实现可靠传输是多么天经地义！TCP是纯软件，可以处理复杂的逻辑判断。客户端电脑不会有太多进程在运行，所以即使纯软件运行，也不会有太大压力！

服务器端会有一些压力，每秒要处理几十万次、甚至千万次的连接，这可以通过服务器集

群、CDN加速来实现流量负载的分摊！

当然，用户对自己实现数据传输的可靠性有足够的自信，完全可以在应用程序里实现可靠性传输代码，只需要调用基于UDP的Socket接口函数即可！

精选留言　　写留言

佳烁

所以QUIC就是最后一段话说的那种吗？

作者　1

很强的归纳能力，对的，因为传统的TCP流量调度算法有点落后于网络速率的升级，带宽并不能有效利用。所以一些大牛公司就可以使用UDP，来实现更快的数据传输，从而完全避开操作系统内部的TCP模块！

τβ

有品位的软件（网络）工程设计！

作者

为了增强一点画面感，会穿插一点人物进入，免得读者看睡着了。

kyle奕™

虽然枯燥的文字已经富有一些生命力了　但要是能把文字做成动画就太棒了\手动狗头

点了广告了

作者

谢谢，协议这块用动画来演示，会更清晰。这块动画内容需要太多的时间与精力！