链滴

Http 协议与 TCP 协议简单理解后续

作者：Jiafeimao

原文链接：https://ld246.com/article/1519916076845

来源网站：链滴

写了这么长时间的代码，发现自己对 TCP/IP 了解的并不是很透彻。虽然会用 C# 的 HttpClient 类来行网络编程，也可以使用 Chrome 的开发者工具来检测每一次的 HTTP 请求的报文头与报文体，也道 cookie 的存在方式，但是对于这些数据怎么在网络上传输还是很模糊，数据是怎么从客户端的文或者字符串转换为二进制数并且传送到服务器端的？为了弄明白这些问题，最近大致的读了读《TCP-I 详解 (卷一、二、三)》，也算是比以前清楚多了，下面是读的过程中的一些知识点。

首先，我们要弄明白这个计算机网络分层的概念。下边这个图是一个经典的分层描述，记得大学时候本上的图也跟这个差不多。

但是我更觉得，大家思想上都有一个抽象的概念，就是分层是垂直的，从上到下的。其实，我觉得，准确的说，这个分层应该是水平的，从左到右的，就像车间的生产线，进去一个大的需要处理的原料经过不同的操作台，一层一层的切割，包装，到最后出来的时候就成为了很多精致的小产品。

关于网络层。

网络层有不同的协议，如 IP 与 ICMP，两者的不同就是对于上层传过来的数据根据什么样的格式进行割，然后再次封装时候遵循的准则不同。

ICMP 是 Ping 命令经常用到的协议。Ping 命令不是什么特别神秘的东西，是一个程序员编写的一个 e e 应用程序，你的电脑控制台之所有能够使用这个程序，是因为你的电脑上安装了这个 exe，而且在 p th 里边设置了这个程序的路径。ICMP 全称是报文控制协议。通过上边的图片可以看出，应用层的 Pi g 工具，使用 Ping 协议，直接跳过运输层，调用了网络层的 ICMP 协议。ICMP 数据包里边内容，是关于目的主机的一些信息，因此可以用于远程判断一台主机是否存在于网络上。ping 程序是对两系统连通性进行测试的基本工具。它只利用 ICMP 回显请求和回显应答报文，而不用经过传输层 TCP UDP。Ping 服务器一般在内核中实现 ICMP 的功能。

网络上一台主机的可达性不仅仅取决于 IP 层是否可达，还要取决于使用何种协议以及端口号。就比说，一台主机确实存在于互联网上边，而且一台 Client 向这台主机使用 Ping 工具发起 ICMP 协议包这些数据包也准确到达了主机。主机在接收到这些数据包之后，从链路层传到网络层一层层拆去包装行解析，但是主机的操作系统从网络层再往上解析的时候，发现了 Ping 的端口为 6666（假设该主机闭了该端口），就不会做出反应，而且默默的把这些数据吞了。那么在 Client 看来，发出去的数据包联了，会认为这个主机找不到。

所以，总结一下 Ping 不同可能的原因：主机不在线，比如说关机了或者拔掉网线了。还有就是网络火墙或者 IP 策略，会对 ICMP 报文进行过滤，ping 命令无法回应，还有就是主机本身的一些策略，过滤掉 ICMP 数据包。

(个人感觉操作系统以及网卡是这样工作的，所有的网络数据都是从一个入口进来的，进来之后操作系与网卡相关的部件就开始从最底层开始解析这些二进制的数据包，一层层的拆包，组装，然后分析，到 IP 层的时候，会对 IP 数据包进行分析，然后进行 TCP 层的分析，这时候就发现了端口号这个概，那么会根据端口号的不同，把这些数据存储在不同的缓冲区域，每个缓冲区域属于一个指定的应用序（以端口号作为标识）。最终应用程序会从自己的缓冲区域来进行网络数据的读取。）

关于 TCP 的通信机制。

当 TCP 发出一个段后，它启动一个定时器，等待目的端确认收到这个报文段。如果不能及时收到一确认，将重发这个报文段。TCP 将保持它首部和数据的检验和。这是一个端到端的检验和，目的是检数据在传输过程中的任何变化。如果收到段的检验和有差错，TCP 将丢弃这个报文段和不确认收到此文段（希望发端超时并重发）。既然 TCP 报文段作为 IP 数据报来传输，而 IP 数据报的到达可能会序，因此 TCP 报文段的到达也可能会失序。如果必要， TCP 将对收到的数据进行重新排序，将收到数据以正确的顺序交给应用层。

另外，TCP 对字节流的内容不作任何解释。TCP 不知道传输的数据字节流是二进制数据，还是 ASCII 字符、EBCDIC 字符或者其他类型数据。对字节流的解释由 TCP 连接双方的应用层解释。这种对字节

的处理方式与 Unix 操作系统对文件的处理方式很相似。Unix 的内核对一个应用读或写的内容不作任解释，而是交给应用程序处理。对 Unix 的内核来说，它无法区分一个二进制文件与一个文本文件。

(这里说一句题外话，就是 ASCII 码与二进制文件的问题。最终保存在计算机硬盘上的数据都是二进数据，那么这个二进制数据是怎么来的，这是一个问题。就拿 txt 文本文件来说，其存储方式就是根据 ASCII 码将文本内容转换成相应的数字，然后用二进制的形式保存并且存储。但是对于 word 等文件说，比较复杂，有专门的软件比如说 Office 来处理，并且有一定的算法来生成这些二进制。所以这是为什么 Word 文件必须要用 Office 软件来打开。Notepad 是操作系统自带的，如果用 Notepad 打开 word , 那么 notepad 就会根据 ASCII 码的方式去解析，最终发现要么无法解析出来字符，要么析出来的字符是乱码。)

每个 TCP 段都包含源端和目的端的端口号，用于寻找发端和收端应用进程。这两个值加上 IP 首部中源端 IP 地址和目的端 IP 地址唯一确定一个 TCP 连接。一个 IP 地址和一个端口号也称为一个插口 soc et.

既然一个 TCP 连接是全双工（即数据在两个方向上能同时传递），因此每个方向必须单独地进行关。这原则就是当一方完成它的数据发送任务后就能发送一个 FIN 来终止这个方向连接。当一端收到一 FIN，它必须通知应用层另一端几经终止了那个方向的数据传送。发送 FIN 通常是应用层进行关闭的果。

与 Telnet 类似，FTP 最早的设计是用于两台不同的主机，这两个主机可能运行在不同的操作系统下使用不同的文件结构、并可能使用不同字符集。但不同的是，Telnet 获得异构性是强制两端都采用同个标准：使用 7 比特 ASCII 码的 NVT。而 FTP 是采用另一种方法来处理不同系统间的差异。FTP 支有限数量的文件类型（A S C II，二进制，等等）和文件结构（面向字节流或记录）。

在一次 HTTP 请求中，form 表单的数据与上传的文件数据有什么不同?

表单数据是根据 ASCII 码转换成的二进制，而上传文件的时候，就是直接读取的计算机硬盘上的二进数据。比如说上传一个 Word 文件，服务器端接收到的会是一大段二进制数据。其实文件在客户端存的时候就是一大段二进制码，那么这个二进制码是怎么生成的？那么就要问微软的 Office 客户端了是它根据一定的方式生成的二进制码然后存在了硬盘上。所以，这就是为什么，一个 exe 生成的文件外的 exe 打不开，因为使用的解码方式不一样，不知道怎么去分析这么一大堆的二进制码，然后生成要字符串展现给用户。

端口号，不是说一个真正存在的实体，或者说在网卡上有个端口啥的。其实端口号就是一个简单的数标识，用于区分不同的应用程序，有点类似于应用程序的 ID，因为网络数据到达了一个主机上边，怎知道这个数据是给哪个应用程序的呢，这时候端口号就起作用了。前面已经指出过， TCP 和 UDP 采 16bit 的端口号来识别应用程序。那么这些端口号是如何选择的呢？服务器一般都是通过知名端口号识别的。例如，对于每个 TCP/IP 实现来说，FTP 服务器的 TCP 端口号都是 2 1，每个 Telnet 服务的 TCP 端口号都是 2 3，每个 TFTP (简单文件传送) 服务器的 UDP 端口号都是 69。

客户端通常对它所使用的端口号并不关心，只需保证该端口号在本机上是唯一的就可以了。客户端口又称作临时端口号（即存在时间很短暂）。这是因为它通常只是在用户运行该客户程序时才存在，而务器则只要主机开着的，其服务就运行。

网络层（ IP）提供点到点的服务，而运输层（ T C P 和 U D P）提供端到端的服务。

在 TCP/IP 协议族中，网络层 IP 提供的是一种不可靠的服务。也就是说，它只是尽可能快地把分组从结点送到目的结点，但是并不提供任何可靠性保证。而另一方面， TCP 在不可靠的 IP 层上提供了一可靠的运输层。为了提供这种可靠的服务， TCP 采用了超时重传、发送和接收端到端的确认分组等制。由此可见，运输层和网络层分别负责不同的功能。

以前一直搞不懂，为什么 IP 层是不可靠的，而 TCP 是建立在 IP 的基础上的，却是可靠的呢？因为了一些冗余的操作来保证可靠。Telnet 和 Rlogin 这两个交互应用要求最小的传输时延，因为人们主用它们来传输少量的交互数据。另一方面，FTP 文件传输则要求有最大的吞吐量。

同一个 HTML 页面，从服务器端发送到客户端浏览器，首先是根据 HTTP 协议，组装字符串，组装一次请求回复，这个回复的字符串包括 header，body 等。然后这个字符串会被转成二进制数据，然给 TCP 层去分解，然后 TCP 层交给 IP 层，拆解成多个 IP 数据包。这时候这些包是无序的，不一定个包先到达。最终这些包再组成文件，如 img,css,js 文件。这就是为什么图片渲染出来的顺序不一样。

IP 层的下一层是数据链路层，我们也可以理解为以太网层或者令牌网。当一台主机把以太网数据帧发到位于同一局域网上的另一台主机时，是根据 48bit 的以太网地址来确定目的接口的。设备驱动程序不检查 IP 数据报中的目的 IP 地址。ARP 为 IP 地址到对应的硬件地址之间提供动态映射。我们之所用动态这个词是因为这个过程是自动完成的，一般应用程序用户或系统管理员不必关心。

在硬件层次上进行的数据帧交换必须有正确的接口地址。但是，TCP/IP 有自己的地址：32 bit 的 IP 址。知道主机的 IP 地址并不能让内核发送一帧数据给主机。内核（如以太网驱动程序）必须知道目端的硬件地址才能发送数据。ARP 的功能是在 32bit 的 IP 地址和采用不同网络技术的硬件地址之间供动态映射。

获取字符串的 ASCII 码

string A = ” Hello World”；

byte[] data = Encoding.ASCII.GetBytes(A);

一次 Http 请求，会建立一个 TCP 连接，然后将内容切割，分组打包，最后发送到服务器。

以前有个疑问，就是总觉得进行 TCP 通信的 A 与 B 之间有个管道。如果 A 在发消息的时候，B 也发消息，那么内容在管道之中不就冲突了么。但是这种想法是错误的。A 与 B 之间根本没有管道，是通 IP 层这种路由方式来进行数据包的转换的，发送方与接收方根本都没有指定的路线。发送与接收都在不同的缓冲区，一般发消息的一方会在发送的内容中添加一个标识符，告诉接收方这次这一批的数发送完了，你去处理吧，处理完了给我个回复。

当我们写代码的时候，有个读取网络数据的 read 方法，以前我一直以为是去网络上都数据。这是错的，这个 read 呢，就是去从缓冲区读取已经被操作系统或者网卡拆箱并且还原了的数据，把这个数读取到程序的内存中。

为什么 TCP 建立连接会花费开销?

这里并不是说要占用很多的互联网上的带宽，这里的花销主要是指电脑上的资源消耗。建立 TCP 连的时候，电脑要做很多的准备工作，建立相应的缓冲区域，根据端口号建立存储区域，还有就是 IP 不可靠的，TCP 要想办法找出空间来存储一些额外的东西来保证可靠性，这都是开销。

还是那句话，建立 TCP 通道，其实根本没有通道，走的是 IP 路由，建立通道主要是在电脑内存上开出相应的空间。TCP 连接一直存在，说明那块相应的缓存区域一直没有被回收。

A 与 B 之间是怎么建立起 TCP 连接的?

这个就涉及到了 3 次握手机制。因为 B 机器上有程序在时刻监视着所有的 IP 数据包，一旦检测到数包中含有 3 次握手的内容，便会打开一个连接，然后通过身份验证等机制，最终建立起 TCP 连接