链滴

计算机网络知识点

作者：douniwan

原文链接：https://ld246.com/article/1584097377740

来源网站：链滴

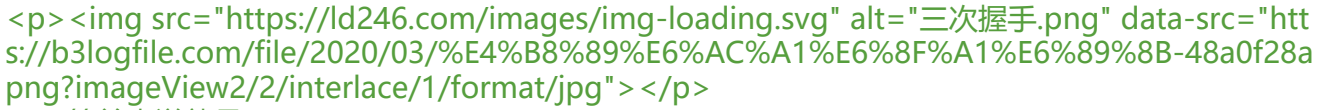
<h3 id="计算机网络模型">计算机网络模型</h3>
计算机网络模型有 OSI 的七层模型，TCP/IP 五层模型以及简化的四层结构，如下所示：
<table>
<tbody><tr>
<td width="104">OSI体系结构</td>
<td width="126">TCP/IP体系结构</td>
<td width="106">五层体系结构</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 应用层</td>
<td rowspan="3">4 应用层</td>
<td rowspan="3">5 应用层</td>
</tr>
<tr>
<td>6 表示层</td>
</tr>
<tr>
<td>5 会话层</td>
</tr>
<tr>
<td>4 传输层</td>
<td>3 传输层</td>
<td>4 传输层</td>
</tr>
<tr>
<td>3 网络层 </td>
<td>2 网络层</td>
<td>3 网络层</td>
</tr>
<tr>
<td>2 数据链路层</td>
<td rowspan="2">1 网络接口层</td>
<td>2 数据链路层</td>
</tr>
<tr>
<td>1 物理层</td>
<td>1 物理层</td>
</tr>
</tbody></table>
<h3 id="协议与各层关系">协议与各层关系</h3>

以 TCP/IP 五层协议为例，常用的协议与层级的对应关系有
应用层：DNS 协议，HTTP 协议，FTP 协议，SMTP 协议
运输层：TCP 协议，UDP 协议
网络层：IP 协议
数据链路层：封装的 IP 报文的数据帧

<h3 id="TCP">TCP</h3>
<blockquote>
TCP 把连接作为最基本的对象，每一条 TCP 连接都有两个端点，这种断点我们叫作套接字（socket），它的定义为端口号拼接到 IP 地址即构成了套接字，例如，若 IP 地址为 192.3.4.16 而端口号为 0，那么得到的套接字为 192.3.4.16:80。
</blockquote>
<h4 id="特点">特点</h4>
TCP 是面向连接的，可靠的协议。

面向连接：使用 TCP 协议通信的双方必须先建立连接，然后才能开始数据的读写，TCP 连接是双工的，即双方的数据读写可以通过一个连接进行。完成数据交换之后，通信双方都必须断开连接以放资源。TCP 协议的这种连接是一对一的，所以基于广播和多播（目标是多个主机地址）的应用程序能使用 TCP
可靠的

TCP 协议采用发送应答机制，即发送端发送的每个 TCP 报文段都必须得到接收方的应答，才能为这个 TCP 报文段传输成功。
TCP 协议采用超时重传机制，发送端在发送出一个 TCP 报文段之后启动定时器，如果在定时时内未收到应答，它将重新发送该报文段。
由于 TCP 报文段最终是以 IP 数据报发送的，而 IP 数据报到达接收端可能乱序、重复、所以 TCP 协议还会将接收到的 TCP 报文段重排、整理、再交付给应用层。

<h4 id="三次握手">三次握手</h4>

简单来说就是：

发送端–发送带有 SYN 标志的数据包–一次握手–接收端
接收端–发送带有 SYN/ACK 标志的数据包–二次握手–发送端
发送端–发送带有带有 ACK 标志的数据包–三次握手–接收端

<h4 id="为什么需要三次握手">为什么需要三次握手</h4>
<blockquote>
三次握手的目的是建立可靠的通信信道，说到通讯，简单来说就是数据的发送与接收，而三次握最主要的目的就是双方确认自己与对方的发送与接收是正常的。
</blockquote>
第一次握手：发送端什么都不能确认；接收端确认自接收正常，发送端发送正常。
第二次握手：发送端确认自己发送，接收正常，接收端接收，发送正常；接收端确认自己接收正常，发送端发送正常。
第三次握手：发送端确认自己发送，接收正常，接收端接收，发送正常；接收端确认自己发送，接收正常，发送端接收，发送正常。
<h4 id="为什么最后发送端还要确认一次呢-为什么不是两次握手-">为什么最后发送端还要确认一次（为什么不是两次握手）</h4>
这里主要有两种情况：

防止已经失效的连接请求报文突然又传送到了服务器，从而产生错误。
如果使用的是两次握手建立连接，假设有这样一种场景，客户端发送了第一个请求连接并且没有失，只是因为在网络结点中滞留的时间太长了，由于 TCP 的客户端迟迟没有收到确认报文，以为服器没有收到，此时重新向服务器发送这条报文，此后客户端和服务器经过两次握手完成连接，传输数

，然后关闭连接。此时此前滞留的那一次请求连接，网络通畅了到达了服务器，这个报文本该是失效，但是，两次握手的机制将会让客户端和服务器再次建立连接，这将导致不必要的错误和资源的浪费
如果采用的是三次握手，就算是那一次失效的报文传送过来了，服务端接受到了那条失效报文并回复了确认报文，但是客户端不会再次发出确认。由于服务器收不到确认，就知道客户端并没有请求接。

防止服务端的确认报文丢失的情况下，服务端认为已连接，客服端认为没有连接，从而照成死锁情况。
如果使用的是两次握手建立连接，假设有这样一种场景，客户端发送了第一个请求连接并且被服务端收到，服务端向客户端发送一个应答的报文，如果这个报文丢失了。那么有两次握手协议，服务端会为连接已经建立好，而客户端会认为连接还没有建立成功。这样客户端就会忽略服务端发来的任何数，只等待应答报文。而服务端在发送报文超时后，就会重新发送，这样服务端和客户端就产生了一个环。

<h4 id="四次挥手">四次挥手</h4>

简单来说就是：

客户端-发送一个 FIN，用来关闭客户端到服务器的数据传送
服务器-收到这个 FIN，它发回一 个 ACK，确认序号为收到的序号加 1 。和 SYN 一样，一个 FIN 将占用一个序号
服务器-关闭与客户端的连接，发送一个 FIN 给客户端
客户端-发回 ACK 报文确认，并将确认序号设置为收到序号加 1

<h4 id="为什么需要四次挥手">为什么需要四次挥手</h4>
第一次挥手：客户端发送一个 FIN=M，用来关闭客户端到服务器端的数据传送，客户端进入 FIN WAIT_1 状态。意思是说"我客户端没有数据要发给你了"，但是如果你服务器端还有数据没有发送完，则不必急着关闭连接，可以继续发送数据。
第二次挥手：服务器端收到 FIN 后，先发送 ack=M+1，告诉客户端，你的请求我收到了，但是还没准备好，请继续你等我的消息。这个时候客户端就进入 FIN_WAIT_2 状态，继续等待服务器端的IN 报文。
第三次挥手：当服务器端确定数据已发送完成，则向客户端发送 FIN=N 报文，告诉客户端，好，我这边数据发完了，准备好关闭连接了。服务器端进入 LAST_ACK 状态。
第四次挥手：客户端收到 FIN=N 报文后，就知道可以关闭连接了，但是他还是不相信网络，怕务器端不知道要关闭，所以发送 ack=N+1 后进入 TIME_WAIT 状态，如果 Server 端没有收到 ACK 可以重传。服务器端收到 ACK 后，就知道可以断开连接了。客户端等待了 2MSL 后依然没有收到回，则证明服务器端已正常关闭，那好，我客户端也可以关闭连接了。最终完成了四次握手。
<h5 id="为什么客户端最后还要等待2MSL-">为什么客户端最后还要等待 2MSL？</h5>
<blockquote>
MSL（Maximum Segment Lifetime），TCP 允许不同的实现可以设置不同的 MSL 值。
第一，保证客户端发送的最后一个 ACK 报文能够到达服务器，因为这个 ACK 报文可能丢失，站在服器的角度看来，我已经发送了 FIN+ACK 报文请求断开了，客户端还没有给我回应，应该是我发送的求断开报文它没有收到，于是服务器又会重新发送一次，而客户端就能在这个 2MSL 时间段内收到这重传的报文，接着给出回应报文，并且会重启 2MSL 计时器。
第二，防止类似与“三次握手”中提到了的“已经失效的连接请求报文段”出现在本连接中。客户端送完最后一个确认报文后，在这个 2MSL 时间中，就可以使本连接持续的时间内所产生的所有报文段从网络中消失。这样新的连接中不会出现旧连接的请求报文。
</blockquote>
<h4 id="为什么建立连接是三次握手-关闭连接确是四次挥手呢-">为什么建立连接是三次握手，关闭

接确是四次挥手呢？</h4>
<blockquote>
建立连接的时候， 服务器在 LISTEN 状态下，收到建立连接请求的 SYN 报文后，把 ACK 和 SYN 放在一个报文里发送给客户端。
而关闭连接时，服务器收到对方的 FIN 报文时，仅仅表示对方不再发送数据了但是还能接收数据，而己也未必全部数据都发送给对方了，所以己方可以立即关闭，也可以发送一些数据给对方后，再发送 FN 报文给对方来表示同意现在关闭连接，因此，己方 ACK 和 FIN 一般都会分开发送，从而导致多了次。
</blockquote>
<h3 id="HTTP协议与TCP-IP协议的关系">HTTP 协议与 TCP/IP 协议的关系</h3>
HTTP 的长连接和短连接本质上是 TCP 长连接和短连接。HTTP 属于应用层协议，在传输层使用 CP 协议，在网络层使用 IP 协议。 IP 协议主要解决网络路由和寻址问题，TCP 协议主要解决如何在 IP 层之上可靠地传递数据包，使得网络上接收端收到发送端所发出的所有包，并且顺序与发送顺序一致 TCP 协议是可靠的、面向连接的。
<h3 id="URL到页面的整个过程">URL 到页面的整个过程</h3>

DNS 解析通常会经过以下这几个过程：

浏览器缓存 - 浏览器缓存 DNS 记录一段时间
系统缓存 - 从 Hosts 文件查找是否有该域名和对应 IP
路由器缓存 - 一般路由器也会缓存域名信息
ISP DNS 缓存 - 到电信的 DNS 查找缓存
都没有找到，则向根域名服务器查找域名对应 IP，根域名服务器把请求转发到下一级查找 IP

建立连接

TCP 协议：与服务器建立 TCP 连接（运输层）
IP 协议：建立 TCP 协议时，需要发送数据，发送数据需要在网络层使用 IP 协议
OPSF 协议：IP 数据包在路由器之间，路由器选择 OPSF 协议
ARP 协议：路由器在与服务器通信时，将 IP 地址转换成 MAC 地址，需要使用 ARP 协议
HTTP 协议：在 TCP 建立完成后，使用 HTTP 协议访问网页

服务器处理请求

浏览器根据 URL 内容生成 HTTP 请求，请求中包含请求文件的位置、请求文件的方式等等
服务器接到请求后，会根据 HTTP 请求中的内容来决定如何获取相应的 HTML 文件
服务器将得到的 HTML 文件发送给浏览器

浏览器解析渲染页面

在执行 HTML 中代码时，根据需要，浏览器会继续请求图片、CSS、JavsScript 等文件，过程同求 HTML 。
