URL访问网站时的网络传输全过程，归纳起来就是：

首先要通过域名找到IP，如果缓存里没有就要请求DNS服务器；得到IP后开始于目的主机进行三次握手来建立TCP连接；连接建立后进行HTTP访问，传输并获取网页内容；传输完后与目的主机四次挥手来断开TCP连接。

整个过程基本分做下面几个部分：

1、域名解析成IP地址；

2、与目的主机进行TCP连接（三次握手）；

3、发送与收取数据；

4、与目的主机断开TCP连接（四次挥手）；

TCP的优点： 可靠，稳定 TCP的可靠体现在TCP在传递数据之前，会有三次握手来建立连接，而且在数据传递时，有确认、窗口、重传、拥塞控制机制，在数据传完后，还会断开连接用来节约系统资源。

TCP的缺点： 慢，效率低，占用系统资源高，易被攻击。 TCP在传递数据之前，要先建连接，这会消耗时间，而且在数据传递时，确认机制、重传机制、拥塞控制机制等都会消耗大量的时间，而且要在每台设备上维护所有的传输连接，事实上，每个连接都会占用系统的CPU、内存等硬件资源。 而且，因为TCP有确认机制、三次握手机制，这些也导致TCP容易被人利用，实现DOS、DDOS、CC等攻击。

UDP的优点： 快，比TCP稍安全 UDP没有TCP的握手、确认、窗口、重传、拥塞控制等机制，UDP是一个无状态的传输协议，所以它在传递数据时非常快。没有TCP的这些机制，UDP较TCP被攻击者利用的漏洞就要少一些。但UDP也是无法避免攻击的，比如：UDP Flood攻击……

UDP的缺点： 不可靠，不稳定 因为UDP没有TCP那些可靠的机制，在数据传递时，如果网络质量不好，就会很容易丢包。

基于上面的优缺点，那么： 什么时候应该使用TCP： 当对网络通讯质量有要求的时候，比如：整个数据要准确无误的传递给对方，这往往用于一些要求可靠的应用，比如HTTP、HTTPS、FTP等传输文件的协议，POP、SMTP等邮件传输的协议。 在日常生活中，常见使用TCP协议的应用如下： 浏览器，用的HTTP FlashFXP，用的FTP Outlook，用的POP、SMTP Putty，用的Telnet、SSH QQ文件传输 ………… 什么时候应该使用UDP： 当对网络通讯质量要求不高的时候，要求网络通讯速度能尽量的快，这时就可以使用UDP。 比如，日常生活中，常见使用UDP协议的应用如下： QQ语音 QQ视频 TFTP ……

TCP报头结构

16位端口号，告知主机该报文段来自哪里（源端口号）以及传给哪个上层协议或应用程序（目的端口）的。TCP通信时，客户端通常使用系统自动选择的临时端口号，而服务器使用知名端口号。

32位序号：一次TCP通信过程中某个传输方向上的字节流的每个字节的编号。

第一个报文段中序号值被系统初始化为某个随机值ISN，那么该传输方向上后续的TCP报文段中序号值将被系统设置为ISN加上该报文段所携带的第一个字节在整个字节流的偏移。

32位确认号：用作对另一方发送的TCP报文段的响应。其值是收到的TCP报文段的序号值加1.假如主机A和主机B进行TCP通信，那么A发送的TCP报文段不仅携带自己的序号，还有对B发送来的TCP报文段的确认号。

4位头部长度：标识该TCP头部有多少个32bit字（4字节），因为4位最大表示15所以TCP头部最长是60字节。

6位标志位：选项有，ACK标志：表示确认号是否有效。将携带ACK标志的TCP报文段成为“确认报文段”。SYN标志：表示请求建立一个连接，将携带SYN标志的报文段称为“同步报文段”。 FIN标志：表示通知对方要关闭连接，将携带FIN标志的报文段称为“结束报文段”。

16位窗口大小：指的是接受通告窗口。它告诉接受端自己接受缓冲区还能容纳多少字节的数据。

16位校验和：由发送端填充，接收端对TCP报文段执行CRC算法检验报文段在传输中是否损坏，检验TCP头部和数据部分。这是TCP可靠传输的一个重要保障。 16位紧急指针。。

②头部选项

还有最后一个选项字段是可变长的可选信息，最多40字节。。。

tcp、udp的区别

1.基于面向连接与无连接。因为每个连接都会占用系统的CPU、内存等硬件资源，所以对系统资源的要求TCP较多UDP少；并且tcp是一对一的，udp是一对一，一对多，多对一，多对多通信。

2.字节流：应用程序对数据的发送和接受没有边界限制，即发送端执行的写操作次数和接受端的读操作次数没有任何数量关系，并且send只是将数据写进发送缓冲区，recv只是从输出缓冲区中获取数据。【因为如果发送端应用程序连续执行写操作，tcp先将数据放入TCP发送缓冲区中，等到缓冲区真正发送数据时，数据可能被封装了一个或多个TCP报文段发出，即TCP发出的tcp报文段个数和执行的写操作次数没有关系。 而接受端收到一个或多个TCP报文段，先将他们携带的数据按序号依次放到接受缓冲区中，通知应用进程接受。应用程序可以一次全部读出，也可以分几次读出，这取决于应用程序的缓冲区大小，即TCP模块接受的报文段个数和读操作次数没关系。】

数据报：sendto发送数据和对端recvfrom接受数次数相等，因此UDP保留了应用层数据的边界。【 发送端应用程序每执行写操作一次，UDP模块就将它封装成一个数据报发送了。接收端必须及时对每个udp数据报执行读操作并且一次接受完，否则不读数据会丢包，或者不读完整数据会被截断发生数据丢失。】

3.tcp是可靠传输，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达;udp是不保证可靠交付。

可靠性体现在：应用程序将数据发送给TCP，然后TCP把数据流分区成适当长度的报文段（通常受该计算机连接的网络的数据链路层的最大传输单元（ MTU）的限制）之后TCP把结果包传给IP层，由它来通过网络将包传送给接收端实体的TCP层。TCP为了保证不发生丢包，就给每个包一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体对已成功收到的包发回一个相应的确认（ACK）；如果发送端实体在合理的往返时延（RTT）内未收到确认，那么对应的数据包就被假设为已丢失将会被进行重传。TCP用一个校验和函数来检验数据是否有错误；在发送和接收时都要计算校验和。

具体体现为：

1．应用数据被分割成TCP认为最适合发送的数据块。这和UDP完全不同，应用程序产生的数据长度将保持不变。由TCP传递给IP的信息单位称为报文段或段。【在TCP三次我握手的前两次握手中（也就是两个SYN报文段中），通过一个“协商”的方式来告知对方自己期待收到的最大报文段长度（MSS），结果使用通信双发较小的MSS最为最终的MSS。在SYN=1的报文段中，会在报文段的选项部分来指定MSS大小（相当于告知对方自己所能接收的最大报文段长度）。在后续通信双发发送应用层数据之前，如果发送数据超过MSS，会对数据进行分段】。

2.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。当TCP收到发自TCP连接另一端的数据，它将发送一个确认。TCP有延迟确认的功能，在此功能没有打开，则是立即确认。功能打开，则由定时器触发确认时间点。

3.TCP有超时重传机制，如果没有收到对端对于发送报文的确认，则会重传一份。如果发送端在合理的往返时延（RTT）内未收到确认，那么对应的数据包就被假设为已丢失将会被进行重传。

4．TCP将保持它首部和数据的检验和。CRC对TCP整个数据报进行校验（头部和数据部分都会校验）用一个校验和函数来检验数据是否有错误；在发送和接收时都要计算校验和。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报文段（希望发端超时并重发）。

5．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失序、重复，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、整理，将收到的数据以正确的顺序交给应用层。【在同一次传输的同一个方向上每个TCP报文都有唯一的标识(序号），也保证了传送到接收端实体的包的按序接收。然后接收端对已成确认功收到的包发回一个相应的ACK；若重复TCP的接收端必须丢弃重复的数据。】

udp和ip协议都是提供不可靠服务，它们需要上层协议来处理数据确认和超时重传。

三次握手与四次挥手

为什么三次握手？1.如果两次握手，因为存在一种情况。client发送的第一个连接请求报文段没有丢失，而是在某个网络节点长时间滞留了，以致延误到连接断开后的某个时间才到服务器。

这个失效报文段到了sever，sever以为client想建立连接，就发送确认报文段并同意建立连接。此时若只有两次握手，那新的连接就建立了。但由于client其实并没建立连接，所以对确认报文段无反应，更发不了数据。可是sever以为新的连接已建立，就傻傻地一直等着发数据，导致sever很多资源被浪费掉。

所以三次握手可以防止已失效的连接请求的报文段突然到sever，使服务器一直等待 浪费了资源。但是三次握手也有缺陷，会导致syn攻击

Timewait是主动关闭的一端在本端已经关闭的前期下，收到对端的关闭请求（FIN报文段）并且将ACK发送出去后所处的状态，这种状态表示：双方都已经完成工作，只是为了确保迟来的数据报能被识别并丢弃；可靠的终止TCP连接。这种状态下客户端连接要等一段2MSL（报文段最大生存时间【2min】）的时间，才能完全关闭。

具体存在原因：①可靠的终止tcp连接。假如用于确认服务器结束报文段的tcp报文段7丢失了，那服务器将重发结束报文段。所以客户端需要停留在一个状态处理多次收到的服务器结束报文段。（也就是向服务器发送确认报文段）。否则，客户端将发送复位报文段给服务器，服务器认为这是一个错误。

②保证让迟来的tcp报文段有足够时间被识别和丢弃。因为如果没有timewait状态，应用程序可以立即建立一个和刚关闭的相似的链接。这个连接可能接受原来连接的携带应用数据的tcp报文段（迟来的报文段）。因此我们此时不应该用此链接的端口号建立新连接，设为timewait状态。然后等待2MSL（保证网络的两个传输方向上迟到的报文段都消失了【被中转路由器丢弃】时间去使迟来的报文段被识别丢弃！

CLOSE\_WAIT是被动关闭的一端在接收到对端关闭请求（FIN报文段）并且将ACK发送出去后所处的状态，这种状态表示：收到了对端关闭的请求，但是本端还没有完成工作，还未关闭。

TCP的流量控制与拥塞控制

超时重传：TCP服务必须能够重传超时时间内未收到确认报文段的TCP报文段。TCP为每个TCP模块都维护了一个重传定时器，定时器在第一次发送TCP报文段时启动，如果超时时间内未收到接收方的确认号报文段，TCP模块将重传TCP报文段并重置定时器。 至于下次重传的超时时间如何选择，以及最多重传几次，这是TCP的重传策略。 与TCP超时重连的策略相似。

滑动窗口：TCP中采用滑动窗口来进行流量控制，滑动窗口的大小意味着接收方还有多大的缓冲区可以用于接收数据。发送方可以通过滑动窗口的大小来确定应该发送多少字节的数据。当滑动窗口为0时，发送方一般不能再发送数据报。

滑动窗口协议的基本原理就是在任意时刻，发送方都维持了一个连续的允许发送的帧的序号，称为发送窗口；同时，接收方也维持了一个连续的允许接收的帧的序号，称为接收窗口。发送窗口和接收窗口的序号的上下界不一定要一样，甚至大小也可以不同。不同的滑动窗口协议窗口大小一般不同。发送方窗口内的序列号代表了那些已经被发送，但是还没有被确认的帧，或者是那些可以被发送的帧。

流量控制的处理过程：

TCP连接阶段，双方协商窗口尺寸，同时接收方预留数据缓存区；

发送方根据协商的结果，发送符合窗口尺寸的数据字节流，并等待对方的确认；

发送方根据确认信息，改变窗口的尺寸，增加或者减少发送未得到确认的字节流中的字节数。调整过程包括：如果出现发送拥塞，发送窗口缩小为原来的一半，同时将超时重传的时间间隔扩大一倍。

TCP的拥塞控制，流量控制

拥塞控制：TCP模块为了防止过多的数据注入网络，使网络中的路由器或链路不致于过载。以此提高网络利用率，降低丢包率，并抱证网络资源对每条数据流的公平性而采取的控制手段。拥塞控制包含四部分内容：慢启动、拥塞避免、快速重传、快速恢复。

慢启动： 网络传输刚开始时，并不知道网络的实际情况，所以采取一种试探的方式控制数据的发送速率。慢启动阶段，数据的发送速率以指数方式增长，即就是拥塞窗口的增长，每次都是收到确认报文段数量的2倍。可以看出慢启动并不慢，拥塞窗口增长的速度很快。所以，为其设置了一个慢启动门限，当达到门限时，就进入到拥塞避免阶段。

拥塞避免： 当拥塞窗口的大小采用慢启动方式增长到慢启动门限时，就进入拥塞避免阶段，拥塞避免阶段不再以指数形式增长拥塞窗口，而是每经过一个往返时间RTT就将发送方的拥塞窗口+1， 使其增长速度减缓。按照线性方式增长。如果发生网络拥塞，比如丢包时，就将慢启动门限设为原来的一半，然后将拥塞窗口设置为1，开始执行慢启动算法。（较低的起点，指数级增长）。

快速重传： 假设发送方发送的报文段分别为： M1,M2,M3,M4,M5,M6 , 当接收方收到M1和M2后，M3报文段丢失，接着收到的M4，M5，M6都不做处理，而是没接收到一个报文段就对M2重复确认，发送方接收到重复的三次确认报文段，就对其后的报文段进行重新发送，而不需要等待超时时间到达。当快速重传后，立即进入到快速恢复阶段。

快速恢复：将慢启动门限设置为原来的一半，然后将拥塞窗口设置为现在的慢启动门限值，不再执行慢启动算法，而是直接进入拥塞避免阶段。使发送窗口成线性方式增大。

流量控制（滑动窗口协议）TCP连接的每一方都有固定大小的缓冲空间。TCP的接收端只允许另一端发送接收端缓冲区所能接纳的数据。这将防止较快主机致使较慢主机的缓冲区溢出。【滑动窗口技术存在于数据链路层和传输层。两者有不同的协议，但基本原理相同。区别是一个是发送帧，一个是发送字节数据。接收方的接受窗口告诉发送方本端tcp接受缓冲区还能容纳多少字节，发送方的发送窗口就可以控制发送数据的速度。】

UDP没有流量控制和拥塞控制，所以在网络拥塞时不会是源主机发送速率降低（对实时通信很有用，比如QQ电话，视频会议等）

————————————————

版权声明：本文为CSDN博主「yuebaii」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_19525389/article/details/81502805