计算机网络个人总结

1.网络的几种分层体系结构

1.OSI七层模型

1.应用层： 文件传输，电子邮件，文件服务，虚拟终端 TFTP，HTTP，SNMP，FTP，SMTP，DNS，Telnet

2.表示层： 数据格式化，代码转换，数据加密 没有协议

3.会话层： 解除或建立与别的接点的联系 没有协议

4.传输层： 提供端对端的接口 TCP，UDP

5.网络层： 为数据包选择路由 IP，ICMP，RIP，OSPF，BGP，IGMP

6.数据链路层： 传输有地址的帧以及错误检测功能 SLIP，CSLIP，PPP，ARP，RARP，MTU

7.物理层： 以二进制数据形式在物理媒体上传输数据 ISO2110，IEEE802，IEEE802.2

2.TCP/IP五层模型

1.应用层

2.传输层

3.网络层

4.数据链路层

5.物理层

3.OSI和TCP/IP的区别

1.TCP/IP协议中的应用层处理开放式系统互联模型中的第五层、第六层和第七层的功能。

2.TCP/IP协议中的传输层并不能总是保证在传输层可靠地传输数据包，因为TCP/IP协议还提供一项名为UDP（用户数据报协议）的选择，而OSI模型可以保证可靠地传输数据包。

2.建立TCP服务器的各个系统调用

1.这些系统调用包括 socket ()、bind ()、listen ()、accept ()、send () 和 receive()。

2.

3.MTU和MSS

1.MTU（Maximum Transmission Unit）最大传输单元，指的是IP数据报能经过一个物理网络的最大报文长度，其中包括了IP首部(从20个字节到60个字节不等)，一般以太网的MTU设为1500字节，加上以太帧首部的长度14字节，也就是一个以太帧不会超过1500+14 = 1514字节。

2.MSS（Maximum Segment Size，最大报文段大小，指的是TCP报文的最大数据报长度，其中不包括TCP首部长度。MSS由TCP链接的过程中由双方协商得出，其中SYN字段中的选项部分包括了这个信息。如果MSS+TCP首部+IP首部大于MTU，那么IP报文就会存在分片，如果小于，那么就可以不需要分片正常发送。

4.路由协议

5.地址解析协议**ARP**

1.基本功能为透过目标设备的IP地址，查询目标设备的MAC地址，以保证通信的顺利进行。

6.LAN、**WAN**、**WLAN**、**VLAN**和**VPN**的区别

1.局域网(Local Area Network，LAN)。

2.广域网 (Wide Area Network，WAN)。

3.无线局域网(Wireless LAN, WLAN)。

4.虚拟局域网(Virtual Local Area Network，VLAN)。指网络中的站点不拘泥于所处的物理位置，根据需要灵活划分不同的逻辑子网中的一种网络技术。

5.虚拟专用网络(Virtual Private Network，VPN)。在公用网络上建立专用网络，进行加密通讯。

7.TCP和**UDP**的区别

1.TCP是面向连接的协议；UDP是面向无连接的协议。

2.TCP保证数据顺序，UDP不保证数据顺序。

3.TCP保证可靠交付,UDP不保证可靠交付。

4.TCP是面向字节流的，把数据堪称无结构的字节流；UDP是面向报文的。

5.TCP有拥塞控制，UDP没有。

6.TCP首部开销20字节;UDP的首部开销小，只有8个字节。

7.TCP的逻辑通信信道是全双工的可靠信道，UDP则是不可靠信道。

8.TCP首部开销20字节;UDP的首部开销小，只有8个字节。

9.每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信。

8.建立**TCP**连接的三次握手

1.第一次握手(SYN=1, seq=x):客户端发送一个TCP的SYN标志位置1的包，指明客户端打算连接的服务器的端口，以及初始序号 X,保存在包头的序列号(Sequence Number)字段里。发送完毕后，客户端进入 SYN_SEND 状态。

2.第二次握手(SYN=1, ACK=1, seq=y, ACKnum=x+1):服务器发回确认包(ACK)应答。即 SYN 标志位和 ACK 标志位均为1。服务器端选择自己 ISN 序列号，放到 Seq 域里，同时将确认序号(Acknowledgement Number)设置为客户的 ISN 加1，即X+1。 发送完毕后，服务器端进入 SYN_RCVD 状态。

3.第三次握手(ACK=1，ACKnum=y+1)：客户端再次发送确认包(ACK)，SYN 标志位为0，ACK 标志位为1，并且把服务器发来 ACK 的序号字段+1，放在确定字段中发送给对方，并且在数据段放写ISN的+1。发送完毕后，客户端进入 ESTABLISHED 状态，当服务器端接收到这个包时，也进入 ESTABLISHED 状态，TCP 握手结束。

SYN攻击：攻击客户端在短时间内伪造大量不存在的IP地址，向服务器不断地发送SYN包，服务器回复确认包，并等待客户的确认。 由于源地址是不存在的，服务器需要不断的重发直至超时，这些伪造的SYN包将长时间占用未连接队列，正常的SYN请求被丢弃，导致目标系统运行缓慢，严重者会引起网络堵塞甚至系统瘫痪。

检测SYN攻击：在服务器上看到大量的半连接状态时，特别是源IP地址是随机的，基本上可以断定这是一次SYN攻击。(使用netstats命令来检测SYN攻击)。

防御SYN攻击：缩短超时（SYN Timeout）时间；增加最大半连接数；过滤网关防护；SYN cookies技术。

9.移除TCP连接的四次挥手

1.第一次挥手(FIN=1，seq=x)：客户端发送一个FIN志位置为1的包，表示自己已经没有数据可以发送了，但是仍然可以接受数据。发送完毕后，客户端进入 FIN_WAIT_1 状态。

2.第二次挥手(ACK=1，ACKnum=x+1)：服务器端确认客户端的 FIN 包，发送一个确认包，表明自己接受到了客户端关闭连接的请求，但还没有准备好关闭连接。发送完毕后，服务器端进入 CLOSE_WAIT 状态，客户端接收到这个确认包之后，进入 FIN_WAIT_2 状态，等待服务器端关闭连接。

3.第三次挥手(FIN=1，seq=y)：服务器端准备好关闭连接时，向客户端发送结束连接请求，FIN 置为1。发送完毕后，服务器端进入 LAST_ACK 状态，等待来自客户端的最后一个ACK。

4.第四次挥手(ACK=1，ACKnum=y+1)：客户端接收到来自服务器端的关闭请求，发送一个确认包，并进入TIME_WAIT状态，等待可能出现的要求重传的 ACK 包。服务器端接收到这个确认包之后，关闭连接，进入 CLOSED 状态。客户端等待了某个固定时间（两个最大段生命周期，2MSL，2 Maximum Segment Lifetime）之后，没有收到服务器端的 ACK ，认为服务器端已经正常关闭连接，于是自己也关闭连接，进入 CLOSED 状态。

TIME_WAIT的意义

1.因为在第四次挥手的时候，HOST1发送的ACK可能丢失并导致HOST2重新发送FIN消息，TIME_WAIT维护连接状态.

10.TCP怎么保证可靠性

1、确认和重传：接收方收到报文就会确认，发送方发送一段时间后没有收到确认就重传。

2、数据校验。

3、数据合理分片和排序：

4、流量控制：当接收方来不及处理发送方的数据，能提示发送方降低发送的速率，防止包丢失。

5、拥塞控制：当网络拥塞时，减少数据的发送。

11.HTTPS和HTTP的区别

1、https协议需要到ca申请证书，一般免费证书较少，因而需要一定费用。

2、http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl加密传输协议。

3、http和https使用的是完全不同的连接方式，用的端口也不一样，http是80，https是443。

4、http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。

12.HTTPS的工作原理

1.客户端发送HTTPS请求。

2.服务端完成配置并发送证书(公钥)。

3.客户端解析证书，由客户端的TLS来完成,得到随机值(私钥)，通过这个随机值来进行加密解密。

4.客户端传送加密信息，传送加密后的随机值(私钥)。

5.服务端解密信息。

6.服务端发送加密后的信息，以对称加密方式。

7.客户端解密信息。

13.对称加密vs非对称加密

1.区别

1.对称加密使用单个密钥，非对称密钥使用两个不同但是相关联的密钥。

2.在对称加密中，密钥是随机选择的，其长度通常设置为128或256位，具体长度取决于所需的安全级别；在非对称加密中，公钥和私钥之间在数学上相关联，这意味着两者之间存在算术联系。攻击者可能利用该模式破解密文，因此非对称密钥需要更长的密钥长度，才能提供相同级别的安全性。

3.对称加密算法运算速度快，并且需要较少的计算资源，但是安全级别低；非对称加密系统运行得非常缓慢，并且由于它们的密钥长度非常长，因此需要更多的计算资源。

14.数字证书

1.数字证书是由电子证书发行机构或者组织（简称CA）生成，放置于网站服务器，证明网站的合法性和加密通信用的cer文件。

2.数字证书包含：(1),网站公开信息;(2),网站公钥;(3),依赖CA的电子签名算法;(4),公开信息的信息摘要经过CA加密的密文;(5),信息摘要算法。

15.http/1.0和http/1.1的区别

1.

16.ping网站用到的协议

1.dns协议、ICMP协议、ARP协议、RARP协议。

2.首先我们需要dns协议，将网址转为IP地址；ping使用ICMP协议；到达网址的局域网中需要使用RARP查找mac地址；在发送主机不知道自己IP的时候也会用到ARP协议。

17.TCP粘包

18.浏览器输入URL后发生了什么

1.DNS域名解析；

2.建立TCP连接；

3.发送HTTP请求；

4.服务器处理请求；

5.返回响应结果；

6.关闭TCP连接；

7.浏览器解析HTML；

8.浏览器布局渲染。

19.长连接与短连接的区别以及使用场景

1.区别：短连接是连接后接收了数据就断开，长连接的连接后保持连接。

2.HTTP1.1增加了持久连接支持(长连接)。

3.使用场景：(1),长连接多用于操作频繁，点对点的通讯，而且连接数不能太多情况。(2)，短连接并发量大，节省资源。

20.单机最大tcp连接数

1.一个tcp连接需要占用一个端口号，系统用一个4四元组来唯一标识一个TCP连接：{local ip, local port,remote ip,remote port}。

2.客户端：因此本地端口个数最大只有65536，端口0有特殊含义，不能使用，所以最大连接数是65535。

3.服务端：server端单机最大tcp连接数约为2的48次方。server端tcp连接4元组中只有remote ip（也就是client ip）和remote port（客户端port）是可变的，因此最大tcp连接为客户端ip数×客户端port数，2的32次方（ip数）×2的16次方（port数）。

21.HTTP 请求方法

1.GET：请求指定的页面信息，并返回实体主体。

2.HEAD：类似于 GET 请求，只不过返回的响应中没有具体的内容，用于获取报头。

3.POST：向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。

4.PUT：从客户端向服务器传送的数据取代指定的文档的内容。

5.DELETE：请求服务器删除指定的页面。

6.CONNECT：HTTP/1.1 协议中预留给能够将连接改为管道方式的代理服务器。

7.OPTIONS：允许客户端查看服务器的性能。

8.TRACE：回显服务器收到的请求，主要用于测试或诊断。

9.PATCH：是对 PUT 方法的补充，用来对已知资源进行局部更新 。

22.HTTP GET和POST的区别

1.GET产生一个TCP数据包；POST产生两个TCP数据包。？

1.对于GET方式的请求，浏览器会把http的header和data一并发送出去，服务器响应200（返回数据）。

2.对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。

	GET	POST
后退按钮/刷新	无害	数据会被重新提交（浏览器应该告知用户数据会被重新提交）。
书签	可收藏为书签	不可收藏为书签
缓存	能被缓存	不能缓存
编码类型	application/x-www-form-urlencoded	application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。
历史	参数保留在浏览器历史中。	参数不会保存在浏览器历史中。
对数据长度的限制	是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。	无限制。
对数据类型的限制	只允许 ASCII 字符。	没有限制。也允许二进制数据。
安全性	与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。在发送密码或其他敏感信息时绝不要使用 GET ！	POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。
可见性	数据在 URL 中对所有人都是可见的。	数据不会显示在 URL 中。

23.五种IO模型介绍和对比

1.IO模型

1.阻塞式IO。使用系统调用，并一直阻塞直到内核将数据准备好，之后再由内核缓冲区复制到用户态，在等待内核准备的这段时间什么也干不了。

2.非阻塞式IO。内核在没有准备好数据的时候会返回错误码，而调用程序不会休眠，而是不断轮询询问内核数据是否准备好。

3.IO多路复用。类似与非阻塞，只不过轮询不是由用户线程去执行，而是由内核去轮询，内核监听程序监听到数据准备好后，调用内核函数复制数据到用户态。

1.多路复用包括：

1.select：线性扫描所有监听的文件描述符，不管他们是不是活跃的。有最大数量限制（32位系统1024，64位系统2048）。

2.poll：同select，不过数据结构不同，需要分配一个pollfd结构数组，维护在内核中。它没有大小限制，不过需要很多复制操作。

3.epoll：用于代替poll和select，没有大小限制。使用一个文件描述符管理多个文件描述符，使用红黑树存储。同时用事件驱动代替了轮询。epoll_ctl中注册的文件描述符在事件触发的时候会通过回调机制激活该文件描述符。epoll_wait便会收到通知。最后，epoll还采用了mmap虚拟内存映射技术减少用户态和内核态数据传输的开销

4.信号驱动式IO。使用信号，内核在数据准备就绪时通过信号来进行通知。

1.首先开启信号驱动io套接字，并使用sigaction系统调用来安装信号处理程序，内核直接返回，不会阻塞用户态。

2.数据准备好时，内核会发送SIGIO信号，收到信号后开始进行io操作。

5.异步IO。异步IO依赖信号处理程序来进行通知

1.异步IO与前面IO模型不同的是：前面的都是数据准备阶段的阻塞与非阻塞，异步IO模型通知的是IO操作已经完成，而不是数据准备完成。

2.异步IO才是真正的非阻塞，主进程只负责做自己的事情，等IO操作完成(数据成功从内核缓存区复制到应用程序缓冲区)时通过回调函数对数据进行处理。

3.unix中异步io函数以aio_或lio_打头。

2.对比

1.前面四种IO模型的主要区别在第一阶段，他们第二阶段是一样的：数据从内核缓冲区复制到调用者缓冲区期间都被阻塞住！

2.前面四种IO都是同步IO：IO操作导致请求进程阻塞，直到IO操作完成。

3.异步IO：IO操作不导致请求进程阻塞。

24.select、poll、epoll之间的区别总结

1.select系统调用的用途是：在一段指定时间内，监听用户所感兴趣的文件描述符上的可读、可写和异常事件。

1.单个进程能够监视的文件描述符的数量存在最大限制，通常是1024，当然可以更改数量，但由于select采用轮询的方式扫描文件描述符，文件描述符数量越多，性能越差；(在linux内核头文件中，有这样的定义：#define __FD_SETSIZE 1024)

2.内核 / 用户空间内存拷贝问题，select需要复制大量的句柄数据结构，产生巨大的开销。

3.select返回的是含有整个句柄的数组，应用程序需要遍历整个数组才能发现哪些句柄发生了事件。

4.select的触发方式是水平触发，应用程序如果没有完成对一个已经就绪的文件描述符进行IO操作，那么之后每次select调用还是会将这些文件描述符通知进程。

2.poll系统调用和select类似，也是在制定时间内轮询一定数量的文件描述符，以测试其中是否有就绪的文件描述符。

1.相比select来讲，它没有fd数量的限制，理论上打开fd的数目跟系统内在有关。poll使用链表保存文件描述符，因此没有了监视文件数量的限制，但其他三个缺点依然存在。

3.epoll是Linux特有的IO复用函数，被认为性能最好的一种方法。

1.调用epoll_create()建立一个epoll对象(在epoll文件系统中为这个句柄对象分配资源)。当某一进程调用epoll_create方法时，Linux内核会创建一个eventpoll结构体。

```
struct eventpoll{
    ....
    /*红黑树的根节点，这颗树中存储着所有添加到epoll中的需要监控的事件*/
    struct rb_root  rbr;
    /*双链表中则存放着将要通过epoll_wait返回给用户的满足条件的事件*/
    struct list_head rdlist;
    ....
};
```

2.调用epoll_ctl向epoll对象注册事件。

3.调用epoll_wait检测事件的发生。

4.epoll适用于连接数量多，但活动连接较少的情况。

5.工作模式：epoll对文件描述符的操作有两种模式：LT（level trigger）和ET（edge trigger）。LT模式是默认模式 1.LT模式：当epoll_wait检测到描述符事件发生并将此事件通知应用程序，应用程序可以不立即处理该事件。下次调用epoll_wait时，会再次响应应用程序并通知此事件。

2/ET模式：当epoll_wait检测到描述符事件发生并将此事件通知应用程序，应用程序必须立即处理该事件。如果不处理，下次调用epoll_wait时，不会再次响应应用程序并通知此事件。

25.一个**ip**配置多个域名

26.应用层协议常用的端口号

服务	端口号	协议

服务	端口号	协议
FTP文件传输协议	21	TCP
SSH 安全登录、文件传送(SCP)和端口重定向	22	TCP
Telnet不安全的文本传送	23	TCP
SMTP Simple Mail Transfer Protocol (E-mail)	25	TCP
DNS	53	TCP
TFTP Trivial File Transfer Protocol	69	UDP
HTTP 超文本传送协议 (WWW)	80	TCP
POP3 Post Office Protocol (E-mail) 服务端	110	TCP
SNMP（simple network management protocol）简单网络管理协客户端	161	IP
SNMP （simple network management protocol）简单网络管理协议	162	IP
HTTPS	443	TCP

27.HTTP状态码

分类	描述
1**	信息，服务器收到请求，需要请求者继续执行操作
2**	成功，操作被成功接收并处理
3**	重定向，需要进一步的操作以完成请求
4**	客户端错误，请求包含语法错误或无法完成请求
5**	服务器错误，服务器在处理请求的过程中发生了错误

URL组成部分

```
http://www.aspxfans.com:8080/news/index.asp?boardID=5&ID=24618&page=1#name
```

1.协议部分：“http:”。

2.域名部分：“www.aspxfans.com”。

3.端口部分："8080"。

4.虚拟目录部分："/news/"。从域名后的第一个“/”开始到最后一个“/”为止。

5.文件名部分:"index.asp"。从域名后的最后一个“/”开始到"?"为止。

6.参数部分："boardID=5&ID=24618&page=1"。从“？”开始到“#”为止之间的部分为参数部分。

7.锚部分："name"。从“#”开始到最后。

TCP超时重传机制

1.在发送一个数据之后，就开启一个定时器，若是在这个时间内没有收到发送数据的ACK确认报文，则对该报文进行重传，在达到一定次数还没有成功时放弃并发送一个复位信号。

2.TCP慢启动

1.慢启动算法的基本思想是当TCP开始在一个网络中传输数据或发现数据丢失并开始重发时，首先慢慢的对网路实际容量进行试探，避免由于发送了过量的数据而导致阻塞。

2.慢启动算法初始设置cwnd为1个报文段，此后每收到一个确认就翻倍。那样，这会使窗口按指数方式增长：发送 1个报文段，然后是2个，接着是4个……。

3.拥塞避免算法

1.对一个给定的连接，初始化cwnd为1个报文段，ssthresh为65535个字节。

2. TCP输出例程的输出不能超过cwnd和接收方通告窗口的大小。拥塞避免是发送方使用 的流量控制，而通告窗口则是接收方进行的流量控制。前者是发送方感受到的网络拥塞的估计，而后者则与接收方在该连接上的可用缓存大小有关。

3. 当拥塞发生时（超时或收到重复确认），ssthresh被设置为当前窗口大小的一半（cwnd和接收方通告窗口大小的最小值，但最少为2个报文段）。此外，如果是超时引起了拥塞，则cwnd被设置为1个报文段（这就是慢启动）。

4.当新的数据被对方确认时，就增加cwnd，但增加的方法依赖于是否正在进行慢启动或拥塞避免。如果cwnd小于或等于ssthresh，则正在进行慢启动，否则正在进行拥塞避免。慢启动一直持续到回到当拥塞发生时所处位置的半时候才停止（因为记录了在步骤2 中制造麻烦的窗口大小的一半），然后转为执行拥塞避免。

4.快速重传和快速恢复算法

1.当收到第3个重复的ACK时，将ssthresh设置为当前拥塞窗口cwnd的一半。重传丢失的报文段。设置cwnd为ssthresh加上3倍的报文段大小。

2. 每次收到另一个重复的ACK时，cwnd增加1个报文段大小并发送1个分组（如果新的cwnd允许发送）。

3. 当下一个确认新数据的ACK到达时，设置cwnd为ssthresh（在第1步中设置的值）。这个 ACK应该是在进行重传后的一个往返时间内对步骤1中重传的确认。另外，这个ACK也应该是对丢失的分组和收到的第1个重复的ACK之间的所有中间报文段 的确认。

DNS域名解析

1.主机向本地域名服务器的查询一般都是采用递归查询。“帮你查”

2.本地域名服务器向根域名服务器的查询的迭代查询。“自己查”

3.DNS使用UDP报文来进行传递查询信息的。考虑到效率原因，TCP连接的开销大得，故采用UDP作为DNS的运输层协议。

1.一次TCP交换则至少包含9个包：三次握手初始化TCP会话、一个查询包、一个响应包以及四次分手的包交换。

2.一次UDP名交换可以短到两个包：一个查询包、一个响应包。

为什么域名根服务器只能有**13**台呢？

1.受限于UDP报文 512字节

12 Header + 5 Question section + 31 Resource record + 15 * n (Other resource record) + 16*n (A record in additional section)

12+5+31+16n+15m（n一般等于m）=48+31n=512字节=======>n=14.968。所以我们的根服务器实际上是可以部署14甚至15台的，部署13台只是为了留点位置给后面可能会有的新服务器；

CSMA/CD

- 1.工作在据链路层。
- 2.载波监听，多点接入，碰撞检测
    - 1.发送数据前先侦听信道是否空闲,若空闲，则立即发送数据。若信道忙碌，则等待一段时间至信道中的信息传输结束后再发送数据。
    - 2.冲突：两个或两个以上的节点都提出发送请。
        - 1.监听2RTT时间。
    - 3.退避算法
        - 1.二进制退避算法：随机在0至(2^k-1)中选择一个数，再乘以RTT作为等待时间，k为碰撞次数。当k大于10时，始终保持0至2^k-1。当k大于16，则发出错误信息，丢弃该包。

为什么连接的时候是三次握手，关闭的时候却是四次握手？

- 因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉Client端，"你发的FIN报文我收到了"。只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。故需要四步握手。

为什么**TIME_WAIT**状态需要经过**2MSL(**最大报文段生存时间**)**才能返回到**CLOSE**状态？

- 可能最后一个ACK丢失。所以TIME_WAIT状态就是用来重发可能丢失的ACK报文。在Client发送出最后的ACK回复，但该ACK可能丢失。Server如果没有收到ACK，将不断重复发送FIN片段。所以Client不能立即关闭，它必须确认Server接收到了该ACK。Client会在发送出ACK之后进入到TIME_WAIT状态。Client会设置一个计时器，等待2MSL的时间。如果在该时间内再次收到FIN，那么Client会重发ACK并再次等待2MSL。所谓的2MSL是两倍的MSL(Maximum Segment Lifetime)。MSL指一个片段在网络中最大的存活时间，2MSL就是一个发送和一个回复所需的最大时间。如果直到2MSL，Client都没有再次收到FIN，那么Client推断ACK已经被成功接收，则结束TCP连接。

为什么不能用两次握手进行连接？

- 3次握手完成两个重要的功能，既要双方做好发送数据的准备工作(双方都知道彼此已准备好)，也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送和确认。

如果已经建立了连接，但是客户端突然出现故障了怎么办？

- TCP还设有一个保活计时器，显然，客户端如果出现故障，服务器不能一直等下去，白白浪费资源。服务器每收到一次客户端的请求后都会重新复位这个计时器，时间通常是设置为2小时，若两小时还没有收到客户端的任何数据，服务器就会发送一个探测报文段，以后每隔75秒钟发送一次。若一连发送10个探测报文仍然没反应，服务器就认为客户端出了故障，接着就关闭连接。

为什么要传回 **SYN**

- 接收端传回发送端所发送的 SYN 是为了告诉发送端，我接收到的信息确实就是你所发送的信号了。

传了 **SYN,**为啥还要传 **ACK**

- 传了 SYN，证明发送方到接收方的通道没有问题；接收方到发送方的通道还需要 ACK 信号来进行验证。

停止等待协议

- 停止等待协议是为了实现可靠传输的。
- 每发完一个分组就停止发送，等待对方确认。在收到确认后再发下一个分组。
- 在停止等待协议中，若接收方收到重复分组，就丢弃该分组，但同时还要发送确认。
- 确认丢失：确认消息在传输过程丢失。
    - 客户端重新发送消息。
    - 服务端接收到重复消息，直接丢弃该信息，并向客服端发送确认信息。
- 确认迟到 ：确认消息在传输过程中迟到。
    - 客服端接收到重复的迟到的确认的消息，直接丢弃。
    - 服务端收到重复的消息，直接丢弃。

自动重传请求 **ARQ** 协议

- 每发送完一个分组需要设置一个超时计时器，只要超过一段时间仍然没有收到确认，就重传前面发送过的分组。
- 简单但是信道利用率低。

连续**ARQ**协议

- 发送方维持一个发送窗口，凡位于发送窗口内的分组可以连续发送出去，而不需要等待对方确认。
- 接收方采用累计确认，对按序到达的最后一个分组发送确认，表明到这个分组为止的所有分组都已经正确收到了。
- 信道利用率高，容易实现，即使确认丢失，也不必重传，但是不能向发送方反映出接收方已经正确收到的所有分组的信息。

滑动窗口与流量控制

- TCP 利用滑动窗口实现流量控制的机制。

    - 流量控制是为了控制发送方发送速率，保证接收方来得及接收。
- TCP 中采用滑动窗口来进行传输控制，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。

DNS为什么在进行区域传输时使用TCP

- 因为DNS在需要跨越广域网或互联网，分组丢失率和往返时间的不确定性要更大些，这对于DNS客户端来说是个考验，好的重传和超时检测就显得更重要了。
- 因为数据同步传送的数据量比一个请求和应答的数据量要多得多。