1 三次握手

（1）第一次握手：主机 A 发送位码为 SYN=1，随机产生 sequence number=x 的数据包到服务器，主机 B 由 SYN=1 知道，A 要求建立连接

（2）第二次握手：主机 B 收到请求后要确认联机信息，向 A 发送 acknowledge number=x+1，ACK=1，随机产生 sequence number=y 的包

（3）第三次握手：主机 A 收到后检查 acknowledge number 是否为 x+1，以及位码 ACK 是否为 1，若正确，主机 A 会再发送 acknowledge number=y+1，ACK=1，主机 B 收到后确认 acknowledge number 的值和 ACK 是否为 1，是则连接建立成功。

完成三次握手，主机 A 和主机 B 开始传送数据。

调用什么 api 会触发 3 次握手

为什么要三次握手而不是两次 为什么不是四次

防止失效的连接被服务器接收，若只是两次握手的话，服务器收到请求返回一个确认就认为已经建立连接，然后就开始等待客户端的数据。而客户端收到确认并不会回复，这会持续消耗服务器的资源，故需要三次握手

2 四次挥手

（1）第一次挥手：主机 A 向主机 B 发送 FIN 报文段，表示关闭数据传送，主机 A 进入 FIN_WAIT_1 状态，表示没有数据要传输了

（2）第二次挥手：主机 B 收到 FIN 报文段后进入 CLOSE_WAIT 状态（被动关闭），然后发送 ACK 确认，表示同意关闭请求了，主机间的数据链路关闭，主机 A 进入 FIN_WAIT_2 状态

（3）第三次挥手：主机 B 等待主机 A 发送完数据，发送 FIN 到主机 A 请求关闭，主机 B 进入 LAST_ACK 状态

（4）第四次挥手：主机 A 收到主机 B 发送的 FIN 后，回复 ACK 确认到主机 B，主机 A 进入 TIME_WAIT 状态。主机 B 收到主机 A 的 ACK 后就关闭连接了，状态为 CLOSED。主机 A 等待 2MSL，仍然没有收到主机 B 的回复，说明主机 B 已经正常关闭了，主机 A 关闭连接

3 超时重传

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。这里比较重要的是超时重传的时间，怎样设置这个定时器的时间（RTO），能保证对网络资源最小的浪费。因为 RTO 太小，可能有些报文只是遇到拥堵或网络不好延迟较大而已，这样就会造成不必要的重传。太大的话，使发送端需要等待过长的时间才能发现数据丢失，影响网络传输效率。由于不同的网络情况不一样，不可能设置一样的 RTO，实际中 RTO 是根据网络中的 RTT（传输往返时间）来自适应调整的

4 奇偶校验

在每个字节中加上一个奇偶校验位，并被传输，即每个字节发送九位数据。数据传输前会确定是奇校验还是偶校验，以保证发送端和接收端用相同的校验方法进行数据校验。假如校验位不符，则认为传输出错。奇校验是在每个字节后增加一个附加位使得 1 的总数为奇数…奇校验用于同步传输，偶校验用于异步传输或低速传输。奇偶校验简单但并不是一种安全的检错方法，其识别错误的能力较低（发生错误的位数为偶数时无法识别）

5 拥塞控制

慢启动 拥塞避免 拥塞发生时，快速重传 快速恢复

6 流量控制（滑动窗口模型）

7 如何应对针对 3 次握手的恶意攻击

DDoS（分布式拒绝服务）攻击：A 向 B 发送一个 SYN 包，B 返回 ACK+SYN 包，但 A 不再发送 ACK 确认包，那么 B 会这样一直等待 A 返回的确认包，占用了资源，这样的状况称为半开连接，直到连接超时（30s 到 2min）才会关闭连接。如果 A 向 B 发送大量的 SYN 包，B 的网络连接资源被耗尽，就构成了攻击。还有更坏的一种方式是在发送的 SYN 包中把源地址设为一个不存在的地址，服务器向一个不存在的地址发送数据包自然得不到回应

DDoS 预防（没有根治的办法，除非不用 TCP/IP 连接）：

（1）设置防火墙（启用防 DDoS 的属性）

（2）限制同时打开 SYN 的半连接数目

（3）缩短 SYN 半连接的 time out 时间

（4）关闭不必要的服务

8 TIME_WAITING 状态

9 解决 TCP 粘包问题

原因：

发生 TCP 粘包或拆包有很多原因，现列出常见的几点，可能不全面，欢迎补充，

1、要发送的数据大于 TCP 发送缓冲区剩余空间大小，将会发生拆包。

2、待发送数据大于 MSS（最大报文长度），TCP 在传输前将进行拆包。

3、要发送的数据小于 TCP 发送缓冲区的大小，TCP 将多次写入缓冲区的数据一次发送出去，将会发生粘包。

4、接收数据端的应用层没有及时读取接收缓冲区中的数据，将发生粘包。

等等。

解决办法：

1、发送端给每个数据包添加包首部，首部中应该至少包含数据包的长度，这样接收端在接收到数据后，通过读取包首部的长度字段，便知道每一个数据包的实际长度了。

2、发送端将每个数据包封装为固定长度（不够的可以通过补 0 填充），这样接收端每次从接收缓冲区中读取固定长度的数据就自然而然的把每个数据包拆分开来。

3、可以在数据包之间设置边界，如添加特殊符号，这样，接收端通过这个边界就可以将不同的数据包拆分开。

等等。

10 用 UDP 实现 TCP

11 TCP 和 UDP 优缺点及应用场景

TCP 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，而且在数据传输时，有确认、窗口、重传及拥塞控制机制，在数据传完后，还会断开连接用来节约系统资源

TCP 缺点：慢，效率低，占用系统资源高，易被攻击

TCP 在传输数据前建立连接会消耗时间，而且在数据传输时，确认机制、重传机制及拥塞控制机制等都会消耗大量的时间

TCP 维持的连接会占用系统的 CPU、内存等硬件资源

TCP 的确认机制、三次握手机制易被攻击，DOS、DDOS 等……

UDP 优点：快，比 TCP 稍安全

UDP 没有 TCP 的握手、确认、窗口、重传及拥塞控制等机制，UDP 是一个无状态的传输协议，所以它在传递数据时非常快　没有这些机制，被攻击者攻击的漏洞就少一点，但 UDP 也是无法避免攻击的，UDP Flood 攻击……

UDP 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

TCP 应用场景：对网络通讯质量有要求时，整个数据要准确无误的传递给对方

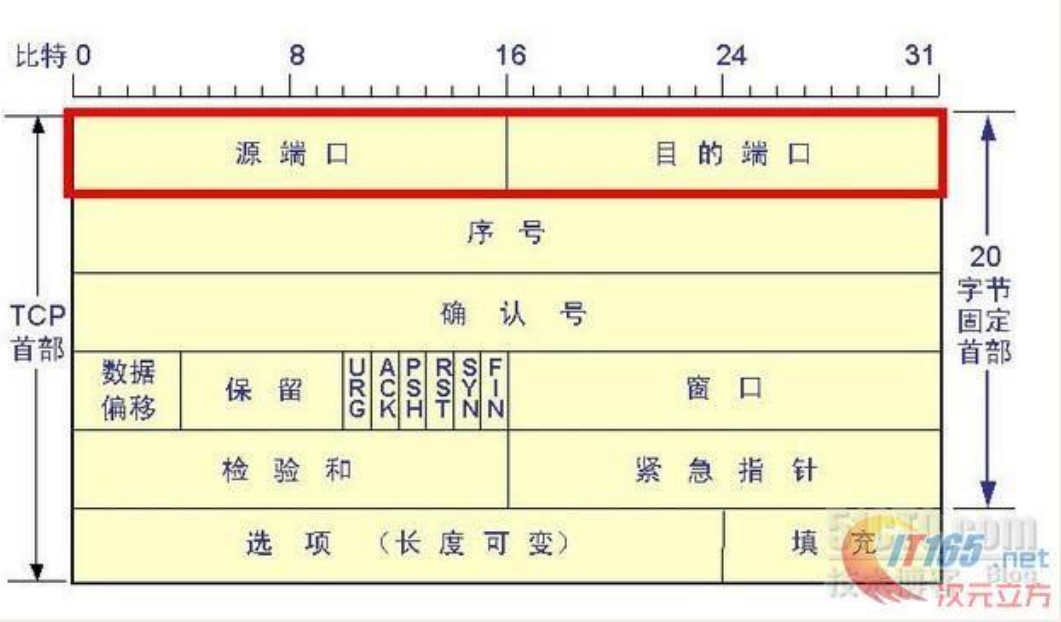
HTTP、HTTPS、FTP 等传输文件的协议，POP、SMTP 等邮件传输协议

QQ 文件传输

UDP 应用场景：对网络通讯质量要求不高且要求网络通讯速度能尽量快时

DNS、QQ 语音、QQ 视频

12 TCP 及 UDP 的报文结构



u32 位端口号:
源端口和目的端口各占 16 位，2 的 16 次方等于 65536，看端口的命令：netstat。
u32 位序号:
也称为顺序号（Sequence Number），简写为 SEQ，
u32 位确认序号:
也称为应答号（Acknowledgment Number），简写为 ACK。在握手阶段，确认序号将发送方的序号加 1 作为回答。
u4 位首部长度:
这个字段占 4 位，它的单位时 32 位（4 个字节）。本例值为 7，TCP 的头长度为 28 字节，等于正常的长度 2 0 字节加上可选项 8 个字节。，TCP 的头长度最长可为 60 字节（二进制 1111 换算为十进制为 15，15*4 字节=60 字节）。
u6 位标志字段:
ACK 置 1 时表示确认号（为合法，为 0 的时候表示数据段不包含确认信息，确认号被忽略。
RST 置 1 时重建连接。如果接收到 RST 位时候，通常发生了某些错误。
SYN 置 1 时用来发起一个连接。
FIN 置 1 时表示发端完成发送任务。用来释放连接，表明发送方已经没有数据发送了。
URG 紧急指针，告诉接收 TCP 模块紧要指针域指着紧要数据。注：一般不使用。
PSH 置 1 时请求的数据段在接收方得到后就可直接送到应用程序，而不必等到缓冲区满时才传送。注：一般不使用。
u16 位检验和:
检验和覆盖了整个的 TCP 报文段： TCP 首部和 TCP 数据。这是一个强制性的字段，一定是由发端计算和存储，并由收端进行验证。
u16 位紧急指针:
注：一般不使用。
只有当 U R G 标志置 1 时紧急指针才有效。紧急指针是一个正的偏移量，和序号字段中的值相加表示紧急数据最后一个字节的序号。
u 可选与变长选项:
通常为空，可根据首部长度推算。用于发送方与接收方协商最大报文段长度（MSS），或在高速网络环境下作窗口调节因子时使用。首部字段还定义了一个时间戳选项。

u 最常见的可选字段是最长报文大小，又称为 MSS (Maximum Segment Size)。每个连接方通常都在握手的第一步中指明这个选项。它指明本端所能接收的最大长度的报文段。1460 是以太网默认的大小。

UDP 首部：

16	31
源端口	目的端口
数据包长度	校验值
数据 DATA	

u2 字节源端口字段
源端口是一个大于 1023 的 16 位数字，由基于 UDP 应用程序的用户进程随机选择。
u2 字节节的端口字段
u2 字节长度字段
指明了包括首部在内的 UDP 报文段长度。UDP 长字段的值是 UDP 报文头的长度(8 字节)与 UDP 所携带数据长度的总和。
u2 字节校验和字段
是指整个 UDP 报文头和 UDP 所带的数据的校验和（也包括伪报文头）。伪报文头不包括在真正的 UDP 报文头中，但是它可以保证 UDP 数据被正确的主机收到了。因在校验和中加入了伪头标，故 ICMP 除能防止单纯数据差错之外，对 IP 分组也具有保护作用。

13 HTTP 和 HTTPS 区别
（1）HTTPS 协议需要到 ca 申请证书，一般免费证书较少，需要交费
（2）HTTP 和 HTTPS 连接方式完全不同，用的端口也不一样，HTTP 是 80，HTTPS 是 443
（3）HTTP 是超文本传输协议，信息是明文传输，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（Secure Sockets Layer）安全套接层

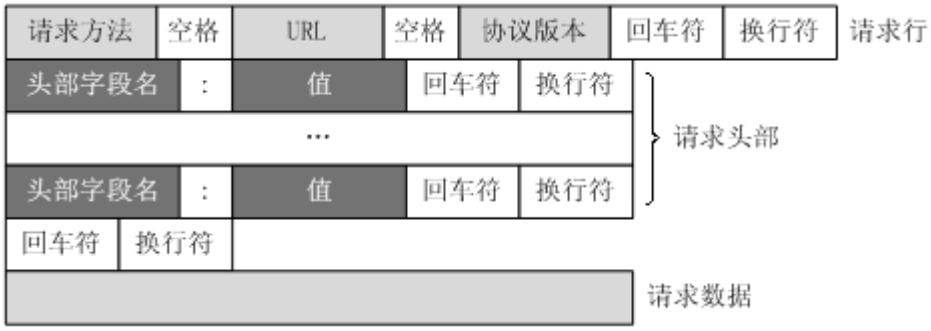
14 HTTPS 请求过程
https://www.cnblogs.com/zhangshitong/p/6478721.html
https://wetest.qq.com/lab/view/110.html
https://www.zhihu.com/question/33645891
对称加密：加密解密用的是同一个密钥，密钥是控制加密及解密过程的指令
优点：算法公开、计算量小、加密速度快、加密效率高
缺点：密钥维护麻烦 密钥协商不安全
非对称加密：公钥加密，私钥解密 RSA MD5
优点：不存在密钥分发的问题 解决了密钥管理的复杂度问题
缺点：加解密的速度没有对称加密快
混合加密：非对称加密建立安全传输信道，对称加密传输数据

15 HTTP 请求过程

一次完整的 HTTP 请求从 TCP 三次握手建立连接成功开始，客户端按照指定的格式开始向服务端发送 HTTP 请求，服务端接收请求后，解析 HTTP 请求，处理完业务逻辑，最后返回一个 HTTP 的响应给客户端，HTTP 的响应内容同样有标准的格式

16 浏览器输入 www.taobao.com，客户端到服务器端的全过程
https://www.jianshu.com/p/c1dfc6caa520

17 HTTP 的报文结构
一个 HTTP 请求报文由请求行（request line）、请求头部（header）、空行和请求数据 4 个部分组成，下图给出了请求报文的一般格式。



or
< request-line >
< headers >
< blank line >
[< request-body >

18 HTTP 请求方式
GET、POST、HEAD、OPTIONS、PUT、DELETE、TRACE、CONNECT

19 GET 和 POST 的区别
最直观的区别就是 GET 把参数包含在 URL 中，POST 通过 request body 传递参数。
GET 在浏览器回退时是无害的，而 POST 会再次提交请求。
GET 产生的 URL 地址可以被 Bookmark，而 POST 不可以。
GET 请求会被浏览器主动 cache，而 POST 不会，除非手动设置。
GET 请求只能进行 url 编码，而 POST 支持多种编码方式。
GET 请求参数会被完整保留在浏览器历史记录里，而 POST 中的参数不会被保留。
GET 请求在 URL 中传送的参数是有长度限制的，而 POST 么有。
对参数的数据类型，GET 只接受 ASCII 字符，而 POST 没有限制。
GET 比 POST 更不安全，因为参数直接暴露在 URL 上，所以不能用来传递敏感信息。
GET 参数通过 URL 传递，POST 放在 Request body 中。

20 状态码
200：成功，服务器已成功处理了请求
301：永久移动，请求的网页已永久移动到新位置
302：临时移动，服务器目前从不同位置的网页响应请求，但请求者应继续用原有位置来进行以后的请求

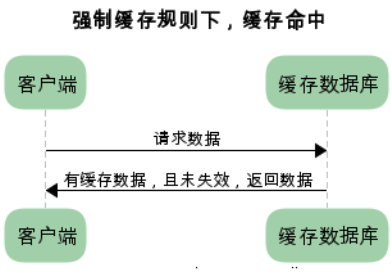
404：未找到，服务器找不到请求的网页
500：服务器内部错误，服务器遇到错误，无法完成请求

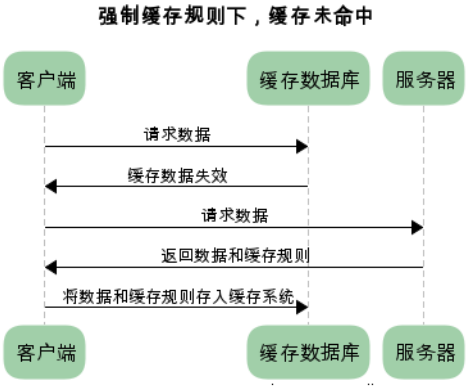
21 HTTP1.0、1.1 和 2.0 的区别
HTTP1.0 与 HTTP 1.1 的主要区别
1 长连接
长连接：客户端和服务器之间用于传输 HTTP 数据的 TCP 连接不会关闭，客户端再次访问这个服务器时，会继续使用一条已经建立的连接。
HTTP 1.0 需要使用 keep-alive 参数来告知服务器端要建立一个长连接，而 HTTP1.1 默认支持长连接。
2 节约带宽

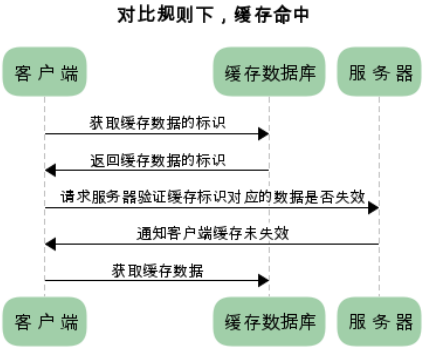
- HTTP 1.1 支持只发送 header 信息(不带任何 body 信息)，如果服务器认为客户端有权限请求服务器，则返回 100，否则返回 401。客户端如果接收到 100，才开始把请求 body 发送到服务器。这样当服务器返回 401 的时候，客户端就可以不用发送请求 body 了，节约了带宽。
- 另外 HTTP 还支持传送内容的一部分。这样当客户端已经有一部分的资源后，只需要跟服务器请求另外的部分资源即可。这是支持文件断点续传的基础。

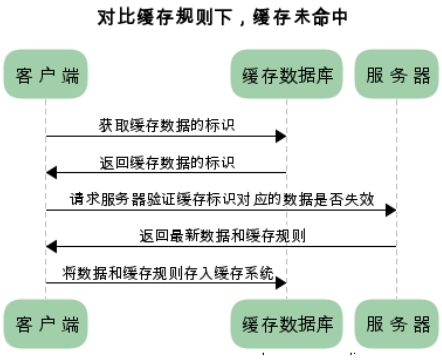
3HOST 域
在网络中，一个 IP 对应多个域名。假设 使用 Tomcat 搭建网站站点，在 Tomact 中搭建多个站点，站点使用相同的 IP 和 PORT，当域名映射成 ip，如何区分站点呢？使用 Host。
现在可以用 web server（例如 tomat），设置虚拟站点是非常常见的，也即是说，web server 上的多个虚拟站点可以共享同一个 ip 和端口。
HTTP1.0 是没有 host 域的，HTTP1.1 才支持这个参数。
HTTP1.1 与 HTTP 2.0 的主要区别
1 多路复用
HTTP2.0 使用了多路复用的技术，做到同一个连接并发处理多个请求，而且并发请求的数量比 HTTP1.1 大了好几个数量级。
当然 HTTP1.1 也可以多建立几个 TCP 连接，来支持处理更多并发的请求，但是创建 TCP 连接本身也是有开销的。
TCP 连接有一个预热和保护的过程，先检查数据是否传送成功，一旦成功过，则慢慢加大传输速度。因此对应瞬时并发的连接，服务器的响应就会变慢。所以最好能使用一个建立好的连接，并且这个连接可以支持瞬时并发的请求。
2 二进制分帧
3 首部压缩
HTTP1.1 不支持 header 数据的压缩，HTTP2.0 使用 HPACK 算法对 header 的数据进行压缩，这样数据体积小了，在网络上传输就会更快。
4 服务器推送
HTTP 2.0 新增的一个强大的新功能，就是服务器可以对一个客户端请求发送多个响应。换句话说，除了对最初请求的响应外，服务器还可以额外向客户端推送资源，而无需客户端明确地请求。
有了 HTTP2.0 的服务器推送，HTTP1.x 时代的内嵌资源的优化手段也变得没有意义了。而且使用服务器推送的资源的方式更加高效，因为客户端还可以缓存起来，甚至可以由不同的页面共享（依旧遵循同源策略）。

22 HTTP 的缓存









我们可以看到两类缓存规则的不同，强制缓存如果生效，不需要再和服务器发生交互，而对比缓存不管是否生效，都需要与服务端发生交互。
两类缓存规则可以同时存在，强制缓存优先级高于对比缓存，也就是说，当执行强制缓存的规则时，如果缓存生效，直接使用缓存，不再执行对比缓存规则。

23 **Cookie 与 Session 的原理**
Cookie 实际上是一小段的文本信息。客户端请求服务器，如果服务器需要记录该用户状态，就使用 response 向客户端浏览器颁发一个 Cookie。客户端浏览器会把 Cookie 保存起来。当浏览器再请求该网站时，浏览器把请求的网址连同该 Cookie 一同提交给服务器。服务器检查该 Cookie，以此来辨认用户状态。服务器还可以根据需要修改 Cookie 的内容。
Session 是另一种记录客户状态的机制，不同的是 Cookie 保存在客户端浏览器中，而 Session 保存在服务器上。客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上。这就是 Session。客户端浏览器再次访问时只需要从该 Session 中查找该客户的状态就可以了。
如果说 Cookie 机制是通过检查客户身上的“通行证”来确定客户身份的话，那么 Session 机制就是通过检查服务器上的“客户明细表”来确认客户身份。Session 相当于程序在服务器上建立的一份客户档案，客户来访的时候只需要查询客户档案表就可以了。

24 一次请求中会在那些地方发生编码解码
1. URL 的编解码
2. HTTP Header 的编解码
3. POST 表单的编解码

4. HTTP BODY 的编解码
5. JS 中的编解码
6. 其他需要编码的地方

除了 URL 和参数编码问题外，在服务端还有很多地方可能存在编码，如可能需要读取 XML、Velocity 模板引擎、JSP 或者从数据库读取数据等。

25 转发与重定向的区别

（1）转发是服务器行为，而重定向是客户端行为

（2）转发速度比重定向快

（3）转发是一次请求，地址栏不变化，而重定向是两次不同的请求，地址栏发生了变化

（4）可以在转发后的页面通过 request.getParameter()取值，而重定向由于没有参数传递，只能通过 session，application 等来取值

（5）转发的路径必须是同一个 web 容器下的 url，而重定向无限制

转发：request.getRequestDispatcher("new.jsp").forward(request, response);

重定向：response.sendRedirect("new.jsp");

26 微信电脑端登录，需要手机验证，中间的通信原理

微信手机客户端从网页二维码里面得到一些信息，然后发送给网页微信的服务器，网页服务器验证信息并响应

https://www.zhihu.com/question/20368066

https://yiweifen.com/html/news/WaiYu/80744.html

27 长连接

HTTP1.1 规定了默认保持长连接（HTTP persistent connection ，也有翻译为持久连接），数据传输完成了保持 TCP 连接不断开（不发 RST 包、不四次握手），等待在同域名下继续用这个通道传输数据；相反的就是短连接。

HTTP 首部的 Connection: Keep-alive 是 HTTP1.0 浏览器和服务器的实验性扩展，当前的 HTTP1.1 RFC2616 文档没有对它做说明，因为它所需要的功能已经默认开启，无须带着它，但是实践中可以发现，浏览器的报文请求都会带上它。如果 HTTP1.1 版本的 HTTP 请求报文不希望使用长连接，则要在 HTTP 请求报文首部加上 Connection: close。《HTTP 权威指南》提到，有部分古老的 HTTP1.0 代理不理解 Keep-alive，而导致长连接失效：客户端-->代理-->服务端，客户端带有 Keep-alive，而代理不认识，于是将报文原封不动转给了服务端，服务端响应了 Keep-alive，也被代理转发给了客户端，于是保持了“客户端-->代理”连接和

“代理-->服务端”连接不关闭，但是，当客户端第发送第二次请求时，代理会认为当前连接不会有请求了，于是忽略了它，长连接失效。书上也介绍了解决方案：当发现 HTTP 版本为 1.0 时，就忽略 Keep-alive，客户端就知道当前不该使用长连接。其实，在实际使用中不需要考虑这么多，很多时候代理是我们自己控制的，如 Nginx 代理，代理服务器有长连接处理逻辑，服务端无需做 patch 处理，常见的是客户端跟 Nginx 代理服务器使用 HTTP1.1 协议&长连接，而 Nginx 代理服务器跟后端服务器使用 HTTP1.0 协议&短连接。
在实际使用中，HTTP 头部有了 Keep-Alive 这个值并不代表一定会使用长连接，客户端和服务器端都可以无视这个值，也就是不按标准来，譬如我自己写的 HTTP 客户端多线程去下载文件，就可以不遵循这个标准，并发的或者连续的多次 GET 请求，都分开在多个 TCP 通道中，每一条 TCP 通道，只有一次 GET，GET 完之后，立即有 TCP 关闭的四次握手，这样写代码更简单，这时候虽然 HTTP 头有 Connection: Keep-alive，但不能说是长连接。正常情况下客户端浏览器、web 服务端都有实现这个标准，因为它们的文件又小又多，保持长连接减少重新开 TCP 连接的开销很有价值。

28 谈谈 DNS 解析，递归查询和迭代查询
（1）递归查询
递归查询是一种 DNS 服务器的查询模式，在该模式下 DNS 服务器接收到客户机请求，
必须使用一个准确的查询结果回复客户机。
如果 DNS 服务器本地没有存储查询 DNS 信息，那么该服务器会询问其他服务器，并将返回的查询结果提交给客户机。客户机和服务器之间的查询是递归查询
是递归查询告诉客户机 IP
（2）迭代查询
DNS 服务器另外一种查询方式为迭代查询，DNS 服务器会向客户机提供其他能够解析查询请求的 DNS 服务器地址，
当客户机发送查询请求时，DNS 服务器并不直接回复查询结果，而是告诉客户机另一台 DNS 服务器地址，
客户机再向这台 DNS 服务器提交请求，依次循环直到返回查询的结果为止。服务器之间的查询是迭代查询

29 ARP 协议
IP 地址-->MAC 地址 Address Resolution Protocol
（1）每个主机都会在自己的 ARP 缓冲区中建立一个 ARP 列表，以表示 IP 地址和 MAC 地址之间的对应关系
（2）当源主机要发送数据时，第一步会检查 ARP 列表中是否有对应 IP 地址的目的主机的 MAC 地址，如果有，则直接发送数据，如果没有，就向本网段的所有主机发送 ARP 数据包，包括的内容有：源主机 IP 地址，源主机 MAC 地址，目的主机的 IP 地址
（3）当本网络的所有主机收到 ARP 数据包时，检查数据包中的 IP 地址是否是自己的 IP 地址，如果不是，则忽略此数据包，如果是，则会从数据包中取出源主机的 IP 和 MAC 地址写入到 ARP 列表中，如果已经存在，则覆盖，然后把自己的 MAC 地址写入 ARP 响应包中，告诉源主机自己是它想要找的 MAC 地址
（4）源主机收到 ARP 响应包后，把目的主机的 IP 和 MAC 地址写入 ARP 列表，并用此信息发送数据。如果源主机一直没有收到 ARP 响应数据包，表示 ARP 查询失败
广播发送 ARP 请求，单播发送 ARP 响应

30 RARP 协议
MAC 地址-->IP 地址
RARP 是逆地址解析协议，作用是完成硬件地址到 IP 地址的映射，主要用于无盘工作站，因为给无盘工作站配置的 IP 地址不能保存。工作流程：在网络中配置一台 RARP 服务器，里面保存着 IP 地址和 MAC 地址的映射关系，当无盘工作站启动后，就封装一个 RARP 数据包，里面有其 MAC 地址，然后广播到网络上去，当服务器收到请求包后，就查找对应的 MAC 地址的 IP 地址装入响应报文中发回给请求者。因为需要广播请求报文，所以 RARP 只能用于具有广播能力的网络

31 OSI 及五层协议的体系结构
OSI 分层 （7 层）：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层。
五层协议 （5 层）：物理层、数据链路层、网络层、运输层、应用层。
物理层：通过媒介传输比特,确定机械及电气规范（比特 Bit） 中继器、集线器
数据链路层：把比特组装成帧和点到点的传递（帧 Frame） PPP、MAC 网桥、交换机
网络层：负责
数据包从源到宿的传递和网际互连（包 PackeT） IP ARP 路由器
传输层：提供端到端的可靠报文传递和错误恢复（段 Segment） TCP UDP
会话层：建立、管理和终止会话（会话协议数据单元 SPDU）
表示层：对数据进行翻译、加密和压缩（表示协议数据单元 PPDU）
应用层：允许访问 OSI 环境的手段（应用协议数据单元 APDU） FTP、DNS、HTTP、SMTP

IP 地址与子网掩码相与得到网络号：
ip : 192.168.2.110
&
Submask : 255.255.255.0

网络号 ： 192.168.2.0

32 怎么判断客户端的连接是不是恶意的
用户名密码验证连接
端口扫描是正常的连接
有些服务器如果在连续的端口地址收到请求就把它视为恶意扫描，会屏蔽对方的 IP，但是扫描器也可以把扫描的端口顺序打乱…
对于网站来说，只开 80 21 端口，关闭不需要的端口
对特定的端口进行监视，如果发现有相同的 IP 对这些端口发出连接请求（在短时间内），把这个 IP 视为恶意的，拒绝请求。但是扫描器可以改变伪装 IP，所以说，对于可要可不要的端口，最好是不要，非要开的端口，一定要打好补丁，限制其守护进程的进程数和内存占用量，防止 DoS，并且作好日志记录