计网高频面试题

- 建立TCP服务器的各个系统调用
  - TCP编程的服务器端一般步骤是：
    - 1、创建一个socket，用函数socket()；　　SOCKET SocketListen =socket(AF_INET,SOCK_STREAM, IPPROTO_TCP);
    - 2、设置socket属性，用函数setsockopt(); * 可选
    - 3、绑定IP地址、端口等信息到socket上，用函数bind(); SOCKET_ERROR = bind(SocketListen,(const sockaddr*)&addr,sizeof(addr))
    - 4、开启监听，用函数listen()；　　SOCKET_ERROR == listen(SocketListen,2)
    - 5、接收客户端上来的连接，用函数accept()；　　SOCKET SocketWaiter = accept(SocketListen,
    - _Out_　struct sockaddr *addr
    - _Inout_ int *addrlen);
    - 6、收发数据，用函数send()和recv()，或者read()和write();
    - 7、关闭网络连接；　closesocket(SocketListen);closesocket(SocketWaiter);
    - 8、关闭监听；
- 继上一题，说明socket网络编程有哪些系统调用？其中close是一次就能直接关闭的吗，半关闭状态是怎么产生的?
- 对路由协议的了解与介绍。内部网关协议IGP包括RIP，OSPF，和外部网关协议EGP和BGP.
  - RIP
  - RIP“路由信息协议(Route Information Protocol)”的简写，主要传递路由信息，通过每隔30秒广播一次路由表，维护相邻路由器的位置关系，同时根据收到的路由表信息计算自己的路由表信 息。RIP是一个距离矢量路由协议,最大跳数为16跳,16跳以及超过16跳的网络则认为目标网络不可达。此协议通常用在网络架构较为简单的小型网络环境.现在分为RIPv1和RIPv2两个版本,后者支持VLSM技术以及一系列技术上的改进。RIP的收敛速度较慢。
  - OSPF
  - OSPF协议是“开放式最短路径优先(Open Shortest Path First)”的缩写，属于链路状态路由协议。OSPF提出了“区域（area）”的概念，每个区域中所有路由器维护着一个相同的链路状态数据库 （LSDB）。区域又分为骨干区域（骨干区域的编号必须为0）和非骨干区域（非0编号区域），如果一个运行OSPF的网络只存在单一区域，则该区域可以是 骨干区域或者非骨干区域。如果该网络存在多个区域，那么必须存在骨干区域，并且所有非骨干区域必须和骨干区域直接相连。OSPF利用所维护的链路状态数据 库，通过最短生成树算法（SPF算法）计算得到路由表。OSPF的收敛速度较快。由于其特有的开放性以及良好的扩展性，目前OSPF协议在各种网络中广泛部署。
  - IS-IS
  - IS-IS协议是Intermediate system to intermediate system（中间系统到中间系统）的缩写，属于链路状态路由协议。标准IS-IS协议是由国际标准化组织制定的ISO/IEC 10589:2002 所定义的，标准IS-IS不适合用于IP网络，因此IETF制 定了适用于IP网络的集成化IS-IS协议（Integrated IS-IS）。和OSPF相同，IS-IS也使用了“区域”的概

> 念，同样也维护着一份链路状态数据库，通过最短生成树算法（SPF）计算出最佳路径。 IS-IS的收敛速度较快。集成化IS-IS协议是ISP骨干网上最常用的IGP协议。

- 路由协议所使用的算法
- TCP和UDP的区别
- TCP和UDP相关的协议与端口号
- TCP（UDP，IP）等首部的认识（http请求报文构成）
- 网页解析的过程与实现方法
- 在浏览器中输入URL后执行的全部过程（如www.baidu.com）
- 网络层分片的原因与具体实现
- TCP的三次握手与四次挥手的详细介绍（TCP连接建立与断开是热门问题）
- TCP握手以及每一次握手客户端和服务器端处于哪个状态（11种状态）
- 为什么使用三次握手，两次握手可不可以?
- TIME_WAIT的意义（为什么要等于2MSL）
- 超时重传机制（不太高频）
- TCP怎么保证可靠性（面向字节流，超时重传，应答机制，滑动窗口，拥塞控制，校验等）?

> - TCP提供一种面向连接的、可靠的字节流服务。
> - 面向连接：意味着两个使用TCP的应用（通常是一个客户和一个服务器）在彼此交换数据之前必须先建立一个TCP连接。在一个TCP连接中，仅有两方进行彼此通信。广播和多播不能用于TCP。 TCP通过下列方式来提供可靠性：
> - 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报长度将保持不变。 (将数据截断为合理的长度)
> - 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。如果不能及时收到一个确认，将重发这个报文段。(超时重发)
> - 3、当TCP收到发自TCP连接另一端的数据，它将发送一个确认。这个确认不是立即发送，通常将推迟几分之一秒 。 (对于收到的请求，给出确认响应) (之所以推迟，可能是要对包做完整校验)
> - 4、 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，目的是检测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报文段。 (校验出包有错，丢弃报文段，不给出响应，TCP发送数据端，超时时会重发数据)
> - 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，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，将收到的数据以正确的顺序交给应用层。 (对失序数据进行重新排序，然后才交给应用层)
> - 6、既然IP数据报会发生重复，TCP的接收端必须丢弃重复的数据。(对于重复数据，能够丢弃重复数据)
> - 7、TCP还能提供流量控制。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。(TCP可以进行流量控制，防止较快主机致使较慢主机的缓冲区溢出)TCP使用的流量控制协议是可变大小的滑动窗口协议。
> - 字节流服务:: 两个应用程序通过TCP连接交换8bit字节构成的字节流。TCP不在字节流中插入记录标识符。我们将这称为字节流服务（bytestreamservice）。TCP对字节流的内容不作任何解释:: 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。

- 流量控制的介绍，采用滑动窗口会有什么问题（死锁可能，糊涂窗口综合征）?

    - 糊涂窗口综合症
        - 这个问题可以归结为小包的问题，就是由于发送端和接收端上的处理不一致，导致网络上产生很多的小包，之前也介绍过避免网络上产生过多小包的措施，比如Nagle算法。在滑动窗口机制下，如果发送端和接收端速率很不一致，也会产生这种比较犯傻的状态：发送方发送的数据，只要一个大大的头部，携带数据很少。
        - 对于接收端来讲，如果接收很慢，一次接收1个字节或者几个字节，这个时候接收端 缓冲区很快就会被填满，然后窗口通告为0字节，这个时候发送端停止发送，应用程序收上去1个字节后，发出窗口通告为1字节，发送方收到通告之后，发出1个字节的数据，这样周而复始，传输效率会非常低。
        - 同时如果发送端程序一次发送一个字节，虽然窗口足够大，但是发送仍是一个字节一个字节的传输，效率很低
- tcp滑动窗口协议
- 拥塞控制和流量控制的区别
- TCP拥塞控制，算法名字？（极其重要）
- http协议与TCP联系
- http/1.0和http/1.1的区别
- http的请求方法有哪些？get和post的区别。
- http的状态码
- http和https的区别，由http升级为https需要做哪些操作
    - HTTP协议传输的数据都是未加密的，也就是明文的，因此使用HTTP协议传输隐私信息非常不安全，为了保证这些隐私数据能加密传输，于是网景公司设计了SSL（Secure Sockets Layer）协议用于对HTTP协议传输的数据进行加密，从而就诞生了HTTPS。简单来说，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，要比http协议安全。
    - HTTPS和HTTP的区别主要如下：
    - 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，前者是80，后者是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。
- https的具体实现，怎么确保安全性
- http中浏览器一个URL的流程，这个过程中浏览器做了什么，URL包括哪三个部分？
    - 在浏览器地址栏输入URL
    - 浏览器查看缓存，如果请求资源在缓存中并且新鲜，跳转到转码步骤
        - 如果资源未缓存，发起新请求
        - 如果已缓存，检验是否足够新鲜，足够新鲜直接提供给客户端，否则与服务器进行验证。
        - 检验新鲜通常有两个HTTP头进行控制Expires和Cache-Control：
            - HTTP1.0提供Expires，值为一个绝对时间表示缓存新鲜日期
            - HTTP1.1增加了Cache-Control: max-age=,值为以秒为单位的最大新鲜时间
    - 浏览器解析URL获取协议，主机，端口，path
    - 浏览器组装一个HTTP（GET）请求报文

- 浏览器获取主机ip地址，过程如下：
  - 浏览器缓存
  - 本机缓存
  - hosts文件
  - 路由器缓存
  - ISP DNS缓存
  - DNS递归查询（可能存在负载均衡导致每次IP不一样）
- 打开一个socket与目标IP地址，端口建立TCP链接，三次握手如下：
  - 客户端发送一个TCP的SYN=1，Seq=X的包到服务器端口
  - 服务器发回SYN=1， ACK=X+1， Seq=Y的响应包
  - 客户端发送ACK=Y+1， Seq=Z
- TCP链接建立后发送HTTP请求
- 服务器接受请求并解析，将请求转发到服务程序，如虚拟主机使用HTTP Host头部判断请求的服务程序
- 服务器检查HTTP请求头是否包含缓存验证信息如果验证缓存新鲜，返回304等对应状态码
- 处理程序读取完整请求并准备HTTP响应，可能需要查询数据库等操作
- 服务器将响应报文通过TCP连接发送回浏览器
- 浏览器接收HTTP响应，然后根据情况选择关闭TCP连接或者保留重用，关闭TCP连接的四次握手如下：
  - 主动方发送Fin=1， Ack=Z， Seq= X报文
  - 被动方发送ACK=X+1， Seq=Z报文
  - 被动方发送Fin=1， ACK=X， Seq=Y报文
  - 主动方发送ACK=Y， Seq=X报文
- 浏览器检查响应状态吗：是否为1XX，3XX， 4XX， 5XX，这些情况处理与2XX不同
- 如果资源可缓存，进行缓存
- 对响应进行解码（例如gzip压缩）
- 根据资源类型决定如何处理（假设资源为HTML文档）
- 解析HTML文档，构件DOM树，下载资源，构造CSSOM树，执行js脚本，这些操作没有严格的先后顺序，以下分别解释
- 构建DOM树：
  - Tokenizing：根据HTML规范将字符流解析为标记
  - Lexing：词法分析将标记转换为对象并定义属性和规则
  - DOM construction：根据HTML标记关系将对象组成DOM树
- 解析过程中遇到图片、样式表、js文件，启动下载
- 构建CSSOM树：
  - Tokenizing：字符流转换为标记流
  - Node：根据标记创建节点
  - CSSOM：节点创建CSSOM树
- 根据DOM树和CSSOM树构建渲染树:
  - 从DOM树的根节点遍历所有可见节点，不可见节点包括：1）script,meta这样本身不可见的标签。2)被css隐藏的节点，如display: none
  - 对每一个可见节点，找到恰当的CSSOM规则并应用

        - 发布可视节点的内容和计算样式
    - js解析如下：
        - 浏览器创建Document对象并解析HTML，将解析到的元素和文本节点添加到文档中，此时document.readystate为loading
        - HTML解析器遇到没有async和defer的script时，将他们添加到文档中，然后执行行内或外部脚本。这些脚本会同步执行，并且在脚本下载和执行时解析器会暂停。这样就可以用document.write()把文本插入到输入流中。同步脚本经常简单定义函数和注册事件处理程序，他们可以遍历和操作script和他们之前的文档内容
        - 当解析器遇到设置了async属性的script时，开始下载脚本并继续解析文档。脚本会在它下载完成后尽快执行，但是解析器不会停下来等它下载。异步脚本禁止使用document.write()，它们可以访问自己script和之前的文档元素
        - 当文档完成解析，document.readState变成interactive
        - 所有defer脚本会按照在文档出现的顺序执行，延迟脚本能访问完整文档树，禁止使用document.write()
        - 浏览器在Document对象上触发DOMContentLoaded事件
        - 此时文档完全解析完成，浏览器可能还在等待如图片等内容加载，等这些内容完成载入并且所有异步脚本完成载入和执行，document.readState变为complete,window触发load事件
    - 显示页面（HTML解析过程中会逐步显示页面）
- 一个机器能够使用的端口号上限是多少，为什么？可以改变吗？那如果想要用的端口超过这个限制怎么办?
- 对称密码和非对称密码体系
- 数字证书的了解（高频）
- 客户端为什么信任第三方证书
- RSA加密算法，MD5原理（MD5不算加密算法）
- 单条记录高并发访问的优化
- 介绍一下ping的过程，分别用到了哪些协议
    - UDP ICMP ARP OSPF
    - ICMP报文：
        - ICMP是（Internet Control Message Protocol）Internet控制报文协议。它是TCP/IP协议族的一个子协议，用于在IP主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用
            - 1、Pc1在应用层发起个目标IP为192.168.2.2的Ping请求。
            - 2、传输层接到上层请求的数据，将数据分段并加上UDP报头。下传到Internet层。
            - 3、网际层接收来处上层的数据后，根据ICMP协议进行封装，添加PC1的IP为源IP为和PC2IP为目标IP后封装成数据包。下传到网络接口层，因Pc1ip与pc2ip不在同一网段，所以数据包将发往网关Router E0口。
            - 4、网络接口层接收数据包后，进行封装，源MAC地址为PC1的MAC地址，目标MAC地址则查询自己的ARP缓存表以获取网关MAC地址。如果PC1 arp缓存表中没有网关对应的MAC地址，则PC1发出一个ARP广播报文。ARP报文中源MAC地址为Pc1mac地址，源IP地址为PC1 IP，所要请求的是网关IP对应的MAC地址
            - 5、交换机1从F0/1接收到ARp帧后，检查自己Arp缓存表中是否有与F0/1口相对应PC1的mac地址。没有，则将PC1Mac地址与F0/1接口对应起来，存

储到交换机1的arp缓存表中。然后将该ARP请求报文进行除F0/1口以外的所有端口进行泛洪。

- 6、Router收到ARP广播后，进行解封装，发现所要请求的MAC地址是自己的。则Router将PC1的mac地址写入arp缓存表中。然后向PC1发送一个ARP应答单播。该单播消息包括目标IP为PC1ip，目标Mac为pc1mac地址，源IP为Router的E0口IP，源Mac为Router的E0的Mac。
- 7、ARP帧F0/24口传给交换机，交换机同样检查MAC表，然后将F0/24口与Router的E0的MaC地址对应起来，存入MAC缓存表中，然后转发该帧。
- 8、Pc1接收到Router的arp应答帧后，将Router的E0的MAC地址存入arp缓存中，并将Router的E0的Mac地址作为目标地址封装到数据帧中。发给下层进行网络传输。
- 9、Router的E0接收这个帧后，看目标mac地址是否指向自己。是，PC2则将帧头去掉，然后检查目标ip地址，发现这个目标ip不是自己，刚不再进行解封装。
- 10、Router在自己的route表中检查自己的是否有去往目标地址的路由，没有则丢弃该帧。有，路由器经检查发现是去往与E1口直连的网段。则路由器对数据包进行二层封装成帧，源IP为pc1的IP，源mac地址为routerE1口的Mac地址，目标IP为Pc2的ip，目标Mac地址则检查自己的arp缓存表获取。如果没有，则发送ARp请求报文。
- 11、交换机收到报文后也检查ARp缓存表，然后存储对应接口的MAC地址后进行除接收端口外的泛洪。
- 12、PC2收到ARP广播后，进行解封装，发现所请求的MAC地址是自己的。则RouterE1的mac地址写入arp缓存表中。然后向PC1发送一个ARP应答单播。该单播消息包括目标IP为RouterE1的ip，目标Mac为RouterE1的mac地址，源IP为PC2的IP，源Mac为pc2的Mac。
- 13、ARP帧经F0/24口传给交换机，交换机同样检查MAC表，然后将F0/24口与PC2的MaC地址对应起来，存入MAC缓存表中，然后转发该帧。
- 14、RouterE1口接收到PC2的arp应答帧后，将Pc2的MAC地址存入arp缓存中，并将Pc2的Mac地址作为目标地址封装到数据帧中，然后转发。
- 15、Pc2网际层接收到这个信息包，查看包头，发现目标IP和自己匹配，则解封装，将数据向上层传输。
- 16、传输层接收来自下层的Ping请求的UDP报文，则去掉UDP报头，向应用层传送。
- 17、应用层收到ping请求后，发送一个PIng回应报文给PC

- TCP/IP的分片粘包过程
- 有没有抓过TCP包，描述一下
- 一个ip配置多个域名，靠什么识别?
- 服务器攻击（DDos攻击）