网络（1）

- TCP编程
  - 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；
  - TCP编程的客户端一般步骤是：
    - 1、创建一个socket，用函数socket()；
    - 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();* 可选
    - 4、设置要连接的对方的IP地址和端口等属性；
    - 5、连接服务器，用函数connect()；
    - 6、收发数据，用函数send()和recv()，或者read()和write();
    - 7、关闭网络连接；
- UDP编程
  - UDP编程的服务器端一般步骤是：
    - 1、创建一个socket，用函数socket()；
    - 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();
    - 4、循环接收数据，用函数recvfrom();
    - 5、关闭网络连接；
  - UDP编程的客户端一般步骤是：
    - 　1、创建一个socket，用函数socket()；
    - 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();* 可选
    - 4、设置对方的IP地址和端口等属性;
    - 5、发送数据，用函数sendto();
    - 6、关闭网络连接；

- TCP三次握手和四次挥手的全过程
- 在浏览器中输入www.baidu.com后执行的全部过程
  - 客户端浏览器通过DNS解析到www.baidu.com的IP地址220.181.27.48，通过这个IP地址找到客户端到服务器的路径。客户端浏览器发起一个HTTP会话到220.181.27.48，然后通过TCP进行封装数据包，输入到网络层。
  - 在客户端的传输层，把HTTP会话请求分成报文段，添加源和目的端口，如服务器使用80端口监听客户端的请求，客户端由系统随机选择一个端口如5000，与服务器进行交换，服务器把相应的请求返回给客户端的5000端口。然后使用IP层的IP地址查 找目的端。
  - 客户端的网络层不用关心应用层或者传输层的东西，主要做的是通过查找路由表确定如何到达服务器，期间可能经过多个路由器，这些都是由路由器来完成的工作，我不作过多的描述，无非就是通过查找路由表决定通过那个路径到达服务器。
  - 客户端的链路层，包通过链路层发送到路由器，通过邻居协议查找给定IP地址的MAC地址，然后发送ARP请求查找目的地址，如果得到回应后就可以使用ARP的请求应答交换的IP数据包现在就可以传输了，然后发送IP数据包到达服务器的地址。
- TCP和UDP的区别?
  - TCP提供面向连接的、可靠的数据流传输，而UDP提供的是非面向连接的、不可靠的数据流传输。
  - TCP传输单位称为TCP报文段，UDP传输单位称为用户数据报。
  - TCP注重数据安全性，UDP数据传输快，因为不需要连接等待，少了许多操作，但是其安全性却一般。
  - TCP对应的协议和UDP对应的协议
  - TCP对应的协议:
    - FTP:定义了文件传输协议，使用21端口。
    - Telnet:一种用于远程登陆的端口，使用23端口，用户可以以自己的身份远程连接到计算机上，可提供基于DOS模式下
    - 的通信服务。
    - SMTP:邮件传送协议，用于发送邮件。服务器开放的是25号端口。
    - POP3:它是和SMTP对应，POP3用于接收邮件。POP3协议所用的是110端口。
    - HTTP:是从Web服务器传输超文本到本地浏览器的传送协议。
  - UDP对应的协议:
    - DNS:用于域名解析服务，将域名地址转换为IP地址。DNS用的是53号端口。
    - SNMP:简单网络管理协议，使用161号端口，是用来管理网络设备的。由于网络设备很多，无连接的服务就体现出其优势。
    - TFTP(Trival File Transfer Protocal)，简单文件传输协议，该协议在熟知端口69上使用UDP服务。
- DNS域名系统，简单描述其工作原理
  - 当DNS客户机需要在程序中使用名称时，它会查询DNS服务器来解析该名称。客户机发送的每条查询信息包括三条信息：包括：指定的DNS域名，指定的查询类型，DNS域名的指定类别。基于UDP服务，端口53. 该应用一般不直接为用户使用，而是为其他应用服务，如HTTP，SMTP等在其中需要完成主机名到IP地址的转换。
- 各种协议的介绍
  - ICMP协议： 因特网控制报文协议。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。

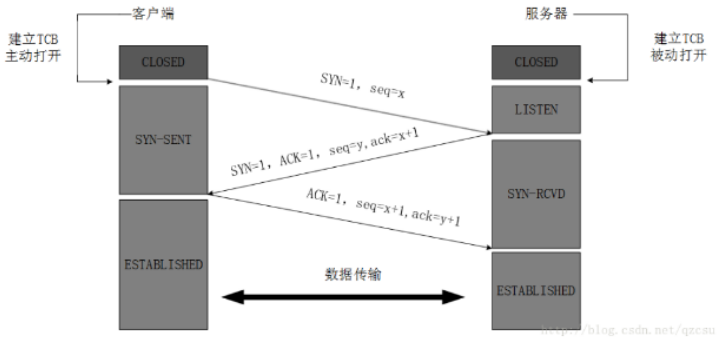
    - TFTP协议： 是TCP/IP协议族中的一个用来在客户机与服务器之间进行简单文件传输的协议，提供不复杂、开销不大的文件传输服务。
    - HTTP协议： 超文本传输协议，是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。
    - NAT协议：网络地址转换属接入广域网(WAN)技术，是一种将私有（保留）地址转化为合法IP地址的转换技术，
    - DHCP协议：动态主机配置协议，是一种让系统得以连接到网络上，并获取所需要的配置参数手段，使用UDP协议工作。具体用途：给内部网络或网络服务供应商自动分配IP地址，给用户或者内部网络管理员作为对所有计算机作中央管理的手段。
- OSI，TCP/IP，五层协议的体系结构，以及各层协议
    - 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。
    - TCP/IP分层（4层）：网络接口层、 网际层、运输层、 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。
    - 每一层的协议如下：
    - 物理层：RJ45、CLOCK、IEEE802.3 （中继器，集线器）
    - 数据链路：PPP、FR、HDLC、VLAN、MAC （网桥，交换机）
    - 网络层：IP、ICMP、ARP、RARP、OSPF、IPX、RIP、IGRP、 （路由器）
    - 传输层：TCP、UDP、SPX
    - 会话层：NFS、SQL、NETBIOS、RPC
    - 表示层：JPEG、MPEG、ASII
    - 应用层：FTP、DNS、Telnet、SMTP、HTTP、WWW、NFS
    - 每一层的作用如下：
    - 物理层：通过媒介传输比特,确定机械及电气规范（比特Bit）
    - 数据链路层：将比特组装成帧和点到点的传递（帧Frame）
    - 网络层：负责数据包从源到宿的传递和网际互连（包PackeT）
    - 传输层：提供端到端的可靠报文传递和错误恢复（段Segment）
    - 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）
- ARP是地址解析协议，简单语言解释一下工作原理。
- 首先，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。
- 当源主机要发送数据时，首先检查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，该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。
- 当本网络的所有主机收到该ARP数据包时，首先检查数据包中的IP地址是否是自己的IP地址，如果不是，则忽略该数据包，如果是，则首先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，然后将自己的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。
- 源主机收到ARP响应包后。将目的主机的IP和MAC地址写入ARP列表，并利用此信息发送数据。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。
- Ping和TraceRoute实现原理
    - Ping是通过发送ICMP报文回显请求实现。

    - TraceRoute通过发送UDP报文，设置目的端口为一个不可能的值，将IP首部中的TTL分别设置从1到N，每次逐个增加，如果收到端口不可达，说明到达目的主机，如果是因为TTL跳数超过，路由器会发送主机不可达的ICMP报文。
- HTTP
    - http的主要特点:
    - 简单快速：当客户端向服务器端发送请求时，只是简单的填写请求路径和请求方法即可，然后就可以通过浏览器或其他方式将该请求发送就行了
    - 灵活： HTTP 协议允许客户端和服务器端传输任意类型任意格式的数据对象
    - 无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接，采用这种方式可以节省传输时间。(当今多数服务器支持Keep-Alive功能，使用服务器支持长连接，解决无连接的问题)
    - 无状态：无状态是指协议对于事务处理没有记忆能力，服务器不知道客户端是什么状态。即客户端发送HTTP请求后，服务器根据请求，会给我们发送数据，发送完后，不会记录信息。(使用 cookie 机制可以保持 session，解决无状态的问题)
    - http1.1的特点
    - a、默认持久连接节省通信量，只要客户端服务端任意一端没有明确提出断开TCP连接，就一直保持连接，可以发送多次HTTP请求
    - b、管线化，客户端可以同时发出多个HTTP请求，而不用一个个等待响应
    - c、断点续传
    - http2.0的特点
    - a、HTTP/2采用二进制格式而非文本格式
    - b、HTTP/2是完全多路复用的，而非有序并阻塞的——只需一个HTTP连接就可以实现多个请求响应
    - c、使用报头压缩，HTTP/2降低了开销
    - d、HTTP/2让服务器可以将响应主动“推送”到客户端缓存中
- get/post 区别
    - 区别一： get重点在从服务器上获取资源，post重点在向服务器发送数据；
    - 区别二： get传输数据是通过URL请求，以field（字段）= value的形式，置于URL后，并用"?"连接，多个请求数据间用"&"连接，如 http://127.0.0.1/Test/login.action?name=admin&password=admin，这个过程用户是可见的； post传输数据通过Http的post机制，将字段与对应值封存在请求实体中发送给服务器，这个过程对用户是不可见的；
    - 区别三： Get传输的数据量小，因为受URL长度限制，但效率较高； Post可以传输大量数据，所以上传文件时只能用Post方式；
    - 区别四： get是不安全的，因为URL是可见的，可能会泄露私密信息，如密码等；post较get安全性较高；
- 返回状态码
    - 200：请求被正常处理
    - 204：请求被受理但没有资源可以返回
    - 206：客户端只是请求资源的一部分，服务器只对请求的部分资源执行GET方法，相应报文中通过Content-Range指定范围的资源。
    - 301：永久性重定向
    - 302：临时重定向
    - 303：与302状态码有相似功能，只是它希望客户端在请求一个URI的时候，能通过GET方法重定向到另一个URI上
    - 304：发送附带条件的请求时，条件不满足时返回，与重定向无关

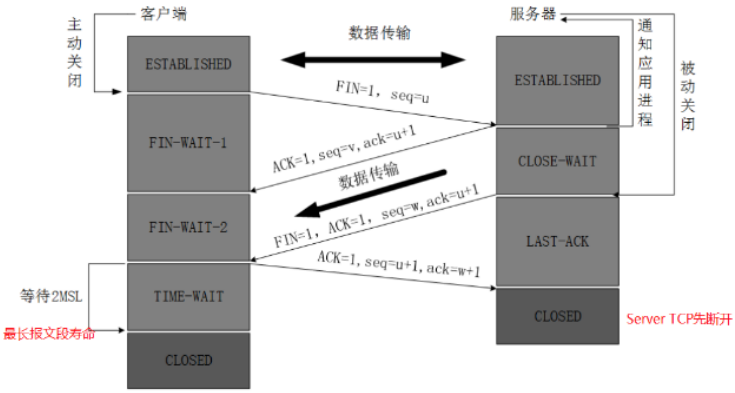
    - 307：临时重定向，与302类似，只是强制要求使用POST方法
    - 400：请求报文语法有误，服务器无法识别
    - 401：请求需要认证
    - 403：请求的对应资源禁止被访问
    - 404：服务器无法找到对应资源
    - 500：服务器内部错误
    - 503：服务器正忙
- 网络

网络

- 网络架构
  - 数据链路层，数据包，奇偶校验
  - 网络层，路由之间辗转，IP协议
  - 传输层，TCP协议（有连接），UDP协议
  - 应用层
- 网络传输不可靠
  - 丢包，重复包
  - 错误
  - 乱序
- 滑动窗口解决发包问题
- TCP解决的问题
  - 基于连接
  - 可靠传输，按序收包
  - 包不出错
  - 流量控制（滑动窗口）
  - 拥塞控制
- TCP的建立连接三次握手，断开连接的四次挥手
  - 三次握手
    -



    - 客户端发送SYN，进入SYN_SENT状态
    - 服务器收到SYN后，进入SYN_RECEIVED状态
    - 服务器发送SYN，ACK（只是标志位）
    - 客户端收到后进入ESTABLISHED状态
    - 客户端发送ACK
    - 客户端发送数据
    - 服务器收到后进入ESTABLISHED状态
  - 四次挥手
    -



    - 发起方发送FIN包，置FIN_WAIT_1
    - 接收方收到后，置CLOSE_WAIT，发送ACK包，准备好了再断开
    - 发起方收到后，置FIN_WAIT_2

    - 接收方准备好后，发送FIN包，置LAST_ACK
    - 发送包发送ACK，置TIME_WAIT
    - 接收方收到后，进入CLOSED状态
    - 发送方等待后，若没有包再发回来，则进入CLOSED状态
  - 为什么 TCP 客户端最后还要发送一次确认呢?
    - 一句话，主要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。
  - 为什么客户端最后还要等待 2MSL?
    - 第一，保证客户端发送的最后一个 ACK 报文能够到达服务器
    - 第二，等待本次连接所有包全部失效，防止其进入新连接
  -
- 从输入 URL 到页面加载发生了什么
  - DNS 解析
    - DNS 存在着多级缓存，从离浏览器的距离排序的话，有以下几种: 浏览器缓存，系统缓存，路由器缓存，IPS 服务器缓存，根域名服务器缓存，顶级域名服务器缓存，主域名服务器缓存。
  - TCP 连接
  - 发送 HTTP 请求
    - HTTP 请求报文是由三部分组成: 请求行, 请求报头和请求正文。
  - 服务器处理请求并返回 HTTP 报文
    - HTTP 响应报文也是由三部分组成: 状态码, 响应报头和响应报文。
  - 浏览器解析渲染页面
  - 连接结束
- GET和POST的区别
  -

	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。

  - 对于 GET 方式的请求，浏览器会把 http header 和 data 一并发送出去，服务器响应 200（返回数据）；
  - （这条有待争议）而对于 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）。
  - 传输数据的大小:
    - GET: 特定浏览器和服务器对 URL 长度有限制
    - POST: 由于不是通过 URL 传值，理论上数据不受限。
    - POST 的安全性要比 GET 的安全性高。
- 返回码301和302的区别
  - 301永久重定向
  - 302临时重定向
    - 比如未登陆的用户访问用户中心重定向到登陆页面。
    - 访问 404 页面会自动重定向到首页
- CDN
  - 将网站的内容通过中心平台分发到部署在各地的边缘服务器进行缓存，再通过负载均衡技术将用户的请求转发到就近的服务器上去获取所需内容，降低网络堵塞，提供访问网站的响应速度和命中率
  - 应用场景
    - 网页加速：主要用于缓存网站的静态数据，比如 JS、CSS、图片和静态页面等。
    - 流媒体服务：主要服务于视频网站，通过将流媒体内容推送到离用户最近的节点，使用户可以从网络边缘获取内容，从而缩短响应时间，提高视频传输质量，减小中心服务器的压力。
    - 文件传输加速：通过使用 CDN 节点提供下载服务，来缓解文件下载带来的性能压力和带宽压力，提供用户下载速度。
    -

        - 应用协议加速：通过对 TCP 等传输协议的优化，改善和加速和改善用户在广域网上的内容传输速度。
    - 工作原理
        - 通过接管 DNS 的方式来把请求引流到离用户最近的缓存服务器上面
- HTTP报文
    - 请求报文
        - 请求行、请求头部、空行和请求数据
        -

    - 响应报文
        - 状态行、消息报头、响应正文。
- Cookie和Session
    - cookie 机制采用的是在客户端保持状态的方案，而 session 机制采用的是在服务器端保持状态的方案。
    - Cookie
        - cookie 的内容主要包括：名字，值，过期时间，路径和域。
        - 若不设置过期时间，则表示这个 cookie 的生命期为浏览器会话期间，关闭浏览器窗口，cookie 就消失。（会话Cookie）
    - Session
        - SessionId借助Cookie发送回客户端
        - 若禁止了客户端Cookie
            - 把 session id 直接附加在 URL 路径的后面。（URL重写）
            - 表单隐藏字段,服务器会自动修改表单，添加一个隐藏字段
    - 1、cookie 数据存放在客户的浏览器上，session 数据放在服务器上。
    - 2、cookie 不是很安全，别人可以分析存放在本地的 cookie 并进行 cookie 欺骗,考虑到安全应当使用 session。
    - 3、session 会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能,考虑到减轻服务器性能方面，应当使用 cookie 。
    - 4、单个 cookie 保存的数据不能超过 4K，很多浏览器都限制一个站点最多保存 20 个 cookie。
    - 5、所以个人建议：
        - 将登陆信息等重要信息存放为 session
        - 其他信息如果需要保留，可以放在 cookie 中
- 转发与重定向的区别
    - 1. 从地址栏显示来说
        - 转发不变
        - 重定向让客户端再次发起请求，地址栏改变
    - 2. 从数据共享来说
        - 转发: 转发页面和转发到的页面可以共享 request 里面的数据.
        - 重定向: 不能共享数据.
    - 3. 从运用地方来说
        - forward: 一般用于用户登陆的时候, 根据角色转发到相应的模块.
        - redirect: 一般用于用户注销登陆时返回主页面和跳转到其它的网站等.
- 抵御DDoS的方法
    - 确保服务器的系统文件是最新的版本, 并及时更新系统补丁。
    - 关闭不必要的服务。
    - 限制同时打开的 SYN 半连接数目, 缩短 SYN 半连接的 time out 时间, 限制 SYN/ICMP 流量
    - 检查访问者的来源，封IP

- http 协议头相关
  - http数据由请求行，首部字段，空行，报文主体四个部分组成
  - 首部字段分为：通用首部字段，请求首部字段，响应首部字段，实体首部字段
- https与http的区别？如何实现加密传输？
  - https就是在http与传输层之间加上了一个SSL
  - 对称加密与非对称加密
- 浏览器中输入一个URL发生什么，用到哪些协议？
  - 浏览器中输入URL，首先浏览器要将URL解析为IP地址，解析域名就要用到DNS协议，首先主机会查询DNS的缓存，如果没有就给本地DNS发送查询请求。DNS查询分为两种方式，一种是递归查询，一种是迭代查询。如果是迭代查询，本地的DNS服务器，向根域名服务器发送查询请求，根域名服务器告知该域名的一级域名服务器，然后本地服务器给该一级域名服务器发送查询请求，然后依次类推直到查询到该域名的IP地址。DNS服务器是基于UDP的，因此会用到UDP协议。
  - 得到IP地址后，浏览器就要与服务器建立一个http连接。因此要用到http协议，http协议报文格式上面已经提到。http生成一个get请求报文，将该报文传给TCP层处理。如果采用https还会先对http数据进行加密。TCP层如果有需要先将HTTP数据包分片，分片依据路径MTU和MSS。TCP的数据包然后会发送给IP层，用到IP协议。IP层通过路由选路，一跳一跳发送到目的地址。当然在一个网段内的寻址是通过以太网协议实现(也可以是其他物理层协议，比如PPP，SLIP)，以太网协议需要直到目的IP地址的物理地址，有需要ARP协议。
-