# OSI七层协议

文本, 表格

中度可信度描述已自动生成

# TCP/IP五层协议

TCP/IP五层协议包括：物理层，数据链路层，网络层，运输层，应用层

# TCP/UDP对应的协议

**1、TCP传输控制协议对应的协议：**

（1） FTP：定义了文件传输协议，使用21端口。

（2） 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。

（3） SMTP：邮件传送协议，用于发送邮件。服务器开放的是25号端口。

（4） POP3：它是和SMTP对应，POP3用于接收邮件。POP3协议所用的是110端口。

（5）HTTP：是从Web服务器传输超文本到本地浏览器的传送协议。

**2、UDP用户数据报协议 对应的协议：**

（1） 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。

（2） 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。

（3） TFTP(Trival File Tran敏感词er Protocal)，简单文件传输协议，该协议在熟知端口69上使用UDP服务。

# UDP与TCP区别

1、TCP面向连接（三次握手四次挥手）;UDP是无连接的，发送数据之前不需要建立连接

2、TCP提供可靠的服务。也就是说，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尽最大努力交付，即不保证可靠交付

3、TCP面向字节流，实际上是TCP把数据看成一连串无结构的字节流;UDP是面向报文的，UDP没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如IP电话，实时视频会议等）

4、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

5、TCP首部开销20字节;UDP的首部开销小，只有8个字节

6、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

# 三次握手

文本

描述已自动生成

为什么二次握手不行？

图片包含 图示

描述已自动生成

# 四次挥手

文本

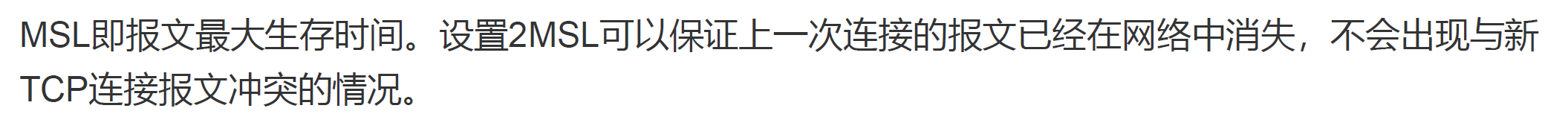
描述已自动生成

为什么三次挥手不行，要四次？

图片包含 文本

描述已自动生成、

为什么要等待2msl？



# TLS介绍

TLS协议可以分为两部分

**记录协议（Record Protocol）**

通过使用客户端和服务端协商后的秘钥进行数据加密传输。

**握手协议（Handshake Protocol）**

客户端和服务端进行协商，确定一组用于数据传输加密的秘钥串。

# TLS记录协议

图示

描述已自动生成

# TLS握手协议

**步骤 1.** ClientHello – 客户端发送所支持的 SSL/TLS 最高协议版本号和所支持的加密算法集合及压缩方法集合等信息给服务器端。

**步骤 2.** ServerHello – 服务器端收到客户端信息后，选定双方都能够支持的 SSL/TLS 协议版本和加密方法及压缩方法，返回给客户端。

**（可选）步骤 3.** SendCertificate – 服务器端发送服务端证书给客户端。

**（可选）步骤 4.** RequestCertificate – 如果选择双向验证，服务器端向客户端请求客户端证书。

**步骤 5.** ServerHelloDone – 服务器端通知客户端初始协商结束。

**（可选）步骤 6.** ResponseCertificate – 如果选择双向验证，客户端向服务器端发送客户端证书。

**步骤 7.** ClientKeyExchange – 客户端使用服务器端的公钥，对客户端公钥和密钥种子进行加密，再发送给服务器端。

**（可选）步骤 8.** CertificateVerify – 如果选择双向验证，客户端用本地私钥生成数字签名，并发送给服务器端，让其通过收到的客户端公钥进行身份验证。

**步骤 9.** CreateSecretKey – 通讯双方基于密钥种子等信息生成通讯密钥。

**步骤 10.** ChangeCipherSpec – 客户端通知服务器端已将通讯方式切换到加密模式。

**步骤 11.** Finished – 客户端做好加密通讯的准备。

**步骤 12.** ChangeCipherSpec – 服务器端通知客户端已将通讯方式切换到加密模式。

**步骤 13.** Finished – 服务器做好加密通讯的准备。

**步骤 14.** Encrypted/DecryptedData – 双方使用客户端密钥，通过对称加密算法对通讯内容进行加密。

**步骤 15.** ClosedConnection – 通讯结束后，任何一方发出断开 SSL 连接的消息。

# cookie/session/token

**工作原理**

**cookie**本质上是一个文本消息，保存了一些用户身份验证相关的信息。

（1）浏览器端第一次发送请求到服务器端

（2）服务器端创建Cookie，该Cookie中包含用户的信息，然后将该Cookie发送到浏览器端

（3）浏览器端再次访问服务器端时会携带服务器端创建的Cookie

（4）服务器端通过Cookie中携带的数据区分不同的用户

图形用户界面, 文本, 应用程序

描述已自动生成

**session**也是用户保存用户身份验证相关的信息

（1）浏览器端第一次发送请求到服务器端，服务器端创建一个Session，同时会创建一个特殊的Cookie（name为JSESSIONID的固定值，value为session对象的ID），然后将该Cookie发送至浏览器端

（2）浏览器端发送第N（N>1）次请求到服务器端,浏览器端访问服务器端时就会携带该name为JSESSIONID的Cookie对象

（3）服务器端根据name为JSESSIONID的Cookie的value(sessionId),去查询Session对象，从而区分不同用户。

图示

低可信度描述已自动生成

token是一种令牌，用于服务端验证用户的身份

1 服务端向用户发送一个token，token是一个经过了签名的hash值，明文是用户的身份验证信息（用户id），该信息经过hash算法处理后用密钥签名成为一个token，服务端不保存token

2 用户再次访问服务端时，在请求头中添加token字段，字段包含用户id以及token，服务端使用密钥和hash算法同样的处理用户id，与用户发来的token对比，相同则验证通过。

**区别**

1 session保存在服务端，cookie保存在客户端浏览器中；

2 cookie不安全，容易被伪造和窃取，session保存在服务端不容易被获取；

3 session会被多次使用，容易消耗服务器性能，而cookie不会；

4 单个cookie大小有限制（4k），而session没有大小限制

一般来说，cookie中保存不敏感的信息，session中保存用户身份验证相关的信息

# HTTP协议结构

**http位于应用层，是建立在tcp/ip之上的，HTTP 协议是一个请求-响应式协议，一般由客户端（比如浏览器）向服务端发起，服务端处理后，再向客户端返回信息。**

**客户端发出的消息称为HTTP 请求（Request），服务端返回的消息称为HTTP 响应（Response）。**

**请求消息体结构**

1 请求行：分别是请求方法+空格+URL+空格+协议版本+\r\n

2 请求头部：由多个请求头部键值对组成，中间以冒号:隔开，每个键值对最后是\r\n

3 空行：即\r\n

4 请求包体：包体部分

**响应消息体机构**

1 状态行：分别是协议版本+空格+状态码+空格+状态码描述+\r\n

2 响应头部：由多个响应头部键值对组成，中间以冒号:隔开，每个键值对最后是\r\n

3 空行：即\r\n

4 响应包体：包体部分

# HTTP请求方法有哪些？

图形用户界面, 文本, 应用程序

描述已自动生成

# GET和POST区别

文本

中度可信度描述已自动生成

# HTTP请求头/响应头属性

**请求头属性**

文本

描述已自动生成

**响应头属性**

文本

描述已自动生成

# HTTP响应码

1xx：信息类 100 101

2xx：成功类 200成功

3xx：重定向类 302重定向

4xx：客户端错误类 400请求地址/参数不存在 401未授权/认证失败 404未找到资源 403禁止访问

5xx：服务端错误类 500内部服务器错误 503服务器访问过载

# 摘要、签名与数字证书

摘要：对一份数据，进行一个单向的 Hash 函数，生成一个固定长度的 Hash 值，这个值就是这份数据的摘要，也称为指纹。常见摘要算法：md5 hash 单项，用于验证完整性

数字签名：确保数据的准确性和不可否认性。

数字证书：确保数据接收者的公钥是没有被篡改过的。

证书生成过程：

图示

描述已自动生成

证书验证过程：

图示

描述已自动生成

# 证书验证过程

1 检验基本信息：首先浏览器读取证书中的证书所有者、有效期等信息进行一一校验

2 校验 CA 机构：浏览器开始查找操作系统中已内置的受信任的证书发布机构 CA，与服务器发来的证书中的颁发者 CA 比对，用于校验证书是否为合法机构颁发；如果找不到，浏览器就会报错，说明服务器发来的证书是不可信任的。

3 解密证书：如果找到，那么浏览器就会从操作系统中取出 颁发者 CA 的公钥，然后对服务器发来的证书里面的签名进行解密

4 比对 hash 值：浏览器使用相同的 hash 算法计算出服务器发来的证书的 hash 值，将这个计算的 hash 值与证书中签名做对比

5 对比结果一致，则证明服务器发来的证书合法，没有被冒充

6 此时浏览器就可以读取证书中的公钥，用于后续加密了

# https建立过程

1 客户端在浏览器中输入url，浏览器会在浏览器的dns缓存、本地dns缓存、host中查找对应的记录，如果没有，则会请求dns服务器来获取对应的ip。

2 客户端向服务端发送支持的加密算法、通信协议、版本

3 服务端筛选出合适的加密协议，返回用自己私钥加密过的证书

4 客户端解密，并使用根证书验证证书合法性

5 客户端生成对称加密密钥，然后使用服务端的公钥进行加密

6 服务端使用私钥解密得到对称加密密钥，然后使用该密钥加密数据

7 客户端解密数据，双方开始通信。

# DNS解析过程

1 浏览器在浏览器的dns缓存、本地dns缓存、host中查找对应的记录，如果没有命中，则进入下一步

2 请求本地域名服务器解析域名，如果没有命中，本地dns服务器会返回一个根域名服务器

3 请求根域名服务器，根域名服务器会返回一个主域名服务器

4 请求主域名服务器，主域名服务器会返回一个域名服务器地址，该域名服务器就是需要解析的域名注册的域名服务器

5 该域名服务器返回域名的ip，本地域名服务器缓存这个记录，并返回给用户，用户将该记录缓存在本地的系统缓存中，域名解析结束。

# 访问百度的过程

1.主要先要解析出百度域名www.baidu.com所对应的ip地址（一个域名只对应一个IP地址，一个ip地址可对应多个域名）

2.得到了baidu.com对应的ip地址后,会发送tcp的3次握手,进行连接

3.使用http协议发送请求数据给web服务器

4.web服务器收到数据请求之后,通过查询自已的服务器得到相应的结果,原路返回给浏览器

5.浏览器接收到数据后通过浏览器自已的渲染功能来显示这个网页

6.浏览器关闭tcp连接,即4次挥手