0.那你说一下OSI 网络分层模型是怎样分层的?

应用层、表示层、会话层、传输层、网络层、数据链路层、物理层

```
application layer、presentation layer、session layer、transport layer、network layer、data link layer、physical layer
```

1.TCP/IP 网络分层模型是怎样分层的?

应用层、传输层、网际层、链接层

```
1  application layer、transport layer、internet layer、link layer
```

2.TCP 和 UDP 区别?

TCP 和 UDP 都是传输层的协议，但二者有着截然不同的基因。

```
1  TCP:
```

- 面向连接
- 面向字节流
- 有状态
- 保证可靠交付
- 具备拥塞控制
- 点对点传播
- 有序

```
1  UDP:
```

- 无连接
- 面向数据报
- 无状态
- 不保证可靠交付
- 不具备拥塞控制
- 广播、多播
- 无序

3.TCP 的三次握手和四次挥手简单说一下

三次握手

- 1.客户端主动发起 SYN
- 2.服务端收到并返回 SYN 以及 ACK 客户端的 SYN
- 3.客户端收到服务端的 SYN 和 ACK 后，发送 ACK 的 ACK 给服务端，服务端收到后连接建立
- `Client -> SYN -> Server`
- `Server -> SYN/ACK -> Client`
- `Client -> ACK -> Server`

四次挥手

- 1.客户端发送 FIN 给服务端
- 2.服务端收到后发送 ACK 给客户端
- 3.服务端发送 FIN 给客户端
- 4.客户端收到后，发送 ACK 的 ACK 给服务端，服务端关闭，客户端等待 2MSL 后关闭
- `Client -> FIN -> Server`
- `Server -> ACK -> Client`
- `Server -> FIN -> Client`
- `Client -> ACK -> Server -> CLOSED`
- `Client -> 2MSL 的时间 -> CLOSED`

4.什么是HTTP协议?

(小白回答版)

HTTP 就是 `超文本传输协议` 呀，它的英文是 `HyperText Transfer Protocol。`

敲黑板!

(罗剑锋老师的完美回答版)

```
1  HTTP 是一个在计算机世界里专门在两点之间传输文字、图片、音频、视频等超文本数据的约定和规范。
```

(面试官：理解的不错)

5.你知道哪些 HTTP 的请求方法?

- GET 获取资源 `（幂等）`
- POST 新增资源
- HEAD 获取HEAD元数据 `（幂等）`
- PUT 更新资源 `（带条件时幂等）`
- DELETE 删除资源 `（幂等）`
- CONNECT 建立 Tunnel 隧道
- OPTIONS 获取服务器支持访问资源的方法 `（幂等）`
- TRACE 回显服务器收到的请求，可以定位问题。`（有安全风险）`

6.说一下HTTP/0.9、HTTP/1.0、HTTP/1.1、HTTP/2、HTTP/3各版本之间的区别?

请移步我的另一篇专栏

7.说一下你对HTTPS的了解

HTTPS 就是在 HTTP 和 TCP 协议中间加入了 SSL/TLS 安全套接层。

结合非对称加密和对称加密的各自优点，配合证书。既保证了安全性，也保证了传输效率。

目前应用最广泛的是 `TLS1.2`，实现原理如下：

- 1.Client 发送 `random1+对称加密套件列表+非对称加密套件列表`
- 2.Server 收到信息， 选择 `对称加密套件+非对称加密套件` 并和 `random2+证书(公钥在证书中)` 一起返回
- 3.Client 验证证书有效性，并用 `random1+random2` 生成 `pre-master` 通过服务器公钥加密+浏览器确认 发送给 Server
- 4.Server 收到 `pre-master`，根据约定的加密算法对 `random1+random2+pre-master（解密）生成 master-secret`，然后发送服务器确认
- 5.Client 收到生成同样的 `master-secert`，对称加密秘钥传输完毕

`TLS1.3` 则简化了握手过程，完全握手只需要一个消息往返，提升了性能。不仅如此，还对部分不安全的加密算法进行了删减。

8.你所谓的约定的加密算法应该是 ECDHE 椭圆算法吧？HTTP 传输消息都是明文的，黑客完全可以作为中间人劫持消息，再利用 ECDHE 算法，这样不就将密钥*解了吗?

ECDHE 算法利用了 `椭圆曲线和离散对数` 等思想，按照当下的计算机算力，很难在短时间进行*解。且每次握手时生成的都是一对临时的公钥和私钥，这样就保证每次的密钥对也不同。

即使大费力气*解了一次的密钥，之前的历史消息也不会受到影响，保证了前向安全。

当然，TLS 协议的安全性受限于当下最快的计算机运行速度，理论上绝对安全的是 `量子通讯传递密钥`。

(面试官：小伙子有点东西)

(基操，勿6)

9.说一说你对DNS的理解?

`DNS（Domain Name System）`是互联网中的重要基础设施，负责对域名的解析工作，为了保证高可用、高并发和分布式，它设计成了树状的层次结构。

由根DNS服务器、顶级域 DNS 服务器和权威 DNS 服务器组成。

解析顺序是首先从 `浏览器缓存` 、 `操作系统缓存` 以及 `本地 DNS 缓存（/etc/hosts）` 逐级查找，然后从 `本地 DNS 服务器` 、 `根 DNS` 、 `顶级 DNS` 以及 `权威 DNS` 层层递归查询。

还可以基于域名在内网、外网进行负载均衡。

不过传统的 DNS 有很多问题(解析慢、更新不及时)， `HTTPDNS` 通过客户端 SDK 和服务端配合，直接通过 HTTP 调用解析 DNS 的方式，可以绕过传统 DNS 这些缺点，实现智能调度。

(面试官：小伙子理解的挺细啊)

10.说一说你对 CDN 的理解?

`CDN（Content Delivery Network）` 就是内容分发网络。

为了突破现实生活中的光速、传输距离等物理限制，CDN 投入了大量资金，在全球范围内各大枢纽城市建立机房，部署大量高存储高带宽的节点，构建跨运营商、跨地域的专用高速传输网络。

其中分为中心节点、区域节点、边缘节点等，在用户接入网络后，首先通过全局负载均衡 `（Global Sever Load Balance）`，简称 GSLB 算法负责调度，找到离用户最合适的节点。然后通过 HTTP 缓存代理技术进行缓存，缓存命中就返回给用户，否则就回源站去取。CDN 擅长缓存静态资源(图片、音频等)，当然也支持动态内容的缓存。

11.说一说你对 WebSocket 的理解?

```
1  WebSocket` 是一种基于 TCP 的轻量级网络通信协议。和 HTTP/2 一样，都是为了解决 HTTP 某些方面的缺陷而诞生的。不过解决方式略有不同，`HTTP/2 针对的是“队头阻塞 ”，WebSocket 针对的是“请求-应答”的通信模式。
```

我们知道“请求-应答”是半双工的通信模式，不具备服务器推送能力。这就限制了 HTTP 在实时通信领域的发展。虽然可以使用轮询来不停的向服务器发送 HTTP 请求，但是缺点也很大，反复的无效请求占用了大量的带宽和 CPU 资源。所以，WebSocket 应运而生。

WebSocket 是一个全双工通信协议，具备服务端主动推送的能力。本质上是对 TCP 做了一层包装，让它可以运行在浏览器环境里。

看过我这篇专栏的同学们一定知道，Webpack 的热更新中就利用了这种协议。当然，在即时通讯、游戏以及可视化大屏展示等领域中也都有着 WebSocket 的身影。

(关于 WebSocket 的过多细节这里不再展开，后续有机会在专栏中详细介绍，感兴趣的同学们可以先自行学习)

12.HTTP 的缓存策略知道吗?

强缓存

服务器使用 `Cache-Control` 来设置缓存策略，常用 `max-age` 来表示资源的有效期。

(这里的 max-age 的时间计算起点是响应报文的创建时刻，而不是客户端收到报文的时刻。)

(浏览器也可以发送 Cache-Control 字段，使用 max-age=0 或 no-cache 来刷新数据)

如果想更精确的控制缓存策略，还可以使用 Cache-Control 的其他属性：

- no-store：不允许缓存 (用于秒杀页面等变化频率非常高的场景)
- no-cache：可以缓存，使用前必须要去服务端验证是否过期，是否是最新版本
- must-revalidate：如果缓存不过期就可以继续使用，过期了就必须去服务端验证

协商缓存

验证资源是否失效就需要使用 `条件请求` 。常用的是 `If-Modified-Since` 和 `If-None-Match`，收到 `304` 状态码就可以复用缓存里的资源。

(If-None-Match 比 If-Modified-Since 优先级更高)

验证资源是否被修改的条件有两个 `Last-modified` 和 `ETag` (ETag 比 Last-modified 的精确度更高)，需要预先在服务端的响应报文里设置，配合条件请求使用。

13.HTTP 如何进行内容协商?

内容协商就是每个 URI 指向的资源可以是任何事物，可以有很多不同的表述。对于文档来说，可以有不同的语言、不同的媒体格式，并针对不同的浏览器提供不同的压缩编码。

- 主动式内容协商
  - 客户端在请求头部中提出需要的表述形式，服务器根据其来进行特定表述
- 响应式内容协商
  - 服务端返回 300 或者 406，由客户端选择一种表述

协商要素

- 质量因子q：内容的质量、可接受类型的优先级
- 媒体资源的 MIME 类型
- 字符编码 (UTF-8)
- 内容编码 (Accept-Encoding:gzip,deflate,br)
- 表述语言 (Accept-Language:zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7)

- 国际化与本地化 (i18n,l10n)

14.说一说 HTTP 的重定向

重定向是服务器发起的跳转，要求客户端使用新的 URI 重新发送请求。在响应头字段 `Location` 中指示了要跳转的 URI。使用 `Refresh` 字段，还可以实现延时重定向。

`301 / 302` 是常用的重定向状态码。分别代表 `永久性重定向` 和 `临时性重定向` 。

除此之外还有：

- 303：类似于 302，重定向后的请求方法改为 `GET` 方法
- 307：类似于 302，含义比 302 更明确，重定向后请求的方法和实体不允许变动
- 308：类似于 301，代表永久重定向，重定向后请求的方法和实体不允许变动
- 300：是一个特殊的重定向状态码，会返回一个有多个链接选项的页面，由用户自行选择
- 304：是一个特殊的重定向状态码，服务端验证过期缓存有效后，要求客户端使用该缓存

15.你知道哪些 HTTP 的常用的首部字段?

(上文中提到过一些，这里只列举一些常用的)

(开始报菜名)

通用首部字段

- `Cache-Control` 控制缓存
- `Connection` 连接管理
- `Transfor-Encoding` 报文主体的传输编码格式
- `Date` 创建报文的时间
- `Upgrade` 升级为其他协议

请求首部字段

- `Host` 请求资源所在的服务器 (唯一一个HTTP/1.1规范里要求必须出现的字段)
- `Accept` 客户端或者代理能够处理的媒体类型
- `If-Match` 比较实体标记 (ETag)
- `If-None-Match` 比较实体标记 (ETag)，与 If-Match 相反
- `If-Modified-Since` 比较资源更新时间 (Last-Modified)
- `If-Unmodified-Since` 比较资源更新时间 (Last-Modified)， 与 If-Modified-Since 相反
- `Range` 实体的字节范围请求
- `User-Agent` 客户端信息

响应首部字段

- `Accept-Ranges` 能接受的字节范围
- `Location` 命令客户端重定向的 URI
- `ETag` 能够表示资源唯一资源的字符串
- `Server` 服务器的信息

实体首部字段

- `Allow` 资源可支持 HTTP 请求方法
- `Last-Modified` 资源最后修改时间
- `Expires` 实体主体过期时间
- `Content-Language` 实体资源语言
- `Content-Encoding` 实体编码格式
- `Content-Length` 实体大小
- `Content-Type` 实体媒体类型

16.你知道哪些 HTTP 状态码?

(上文中提到过一些，这里只列举一些常用的)

1xx

- `1xx：请求已经接收到，需要进一步处理才能完成，HTTP/1.0 不支持`
- `100 Continue：` 上传大文件前使用
- `101 Switch Protocols：` 协议升级使用
- `102 Processing：` 服务器已经收到并正在处理请求，但无响应可用

2xx

- `2xx：成功处理请求`
- `200 OK：` 成功返回响应
- `201 Created：` 有新资源在服务器端被成功创建
- `202 Accepted：` 服务器接受并开始处理请求，但请求未处理完成
- `206 Partial Content：` 使用range协议时返回部分响应内容时的响应码

3xx

请查阅上文重定向部分，这里不再赘述。

4xx

- `4xx：客户端出现错误`
- `400 Bad Request`：服务器认为客户端出现了错误，但不明确，一般是 HTTP 请求格式错误
- `401 Unauthorized`：用户认证信息确实或者不正确
- `403 Forbidden`：服务器理解请求的含义，但没有权限执行
- `407 Proxy Authentication Required`：对需要经由代理的请求，认证信息未通过代理服务器的验证
- `404 Not Found`：服务器没有找到对应的资源
- `408 Request Timeout`：服务器接收请求超时

5xx

- `5xx：服务器端出现错误`
- `500 Internal Server Error`：服务器内部错误，且不属于以下错误类型
- `502 Bad Gateway`：代理服务器无法获取到合法响应
- `503 Service Unavailable`：服务器资源尚未准备好处理当前请求
- `505 HTTP Version Not Supported`：请求使用的 HTTP 协议版本不支持

小姐姐拿起桌旁已经凉透的芋泥波波奶茶，喝了一口。

作者：童欧巴
链接：https://juejin.im/post/6844904121510854664
来源：掘金