转载请注明链接：http://blog.csdn.net/feather_wch/article/details/79643743

> 本文的题目涉及到Android所需要的网络基础、TCP、Http的所有内容。
> 全部背下来，就能说自己对Android 网络有深刻理解

Android网络基础(134题)

版本：2018/9/10-1(16:36)

基础(11)

1、网络分层是什么?

> 1. 将 数据的发送、转发、打包、拆包 ， 以及 控制信息的加载和拆出 等工作，分别交由不同的层级去完成。
> 2. 最终将 通信和网络互联 的复杂问题变得简单。

2、网络分层(五层从上至下)

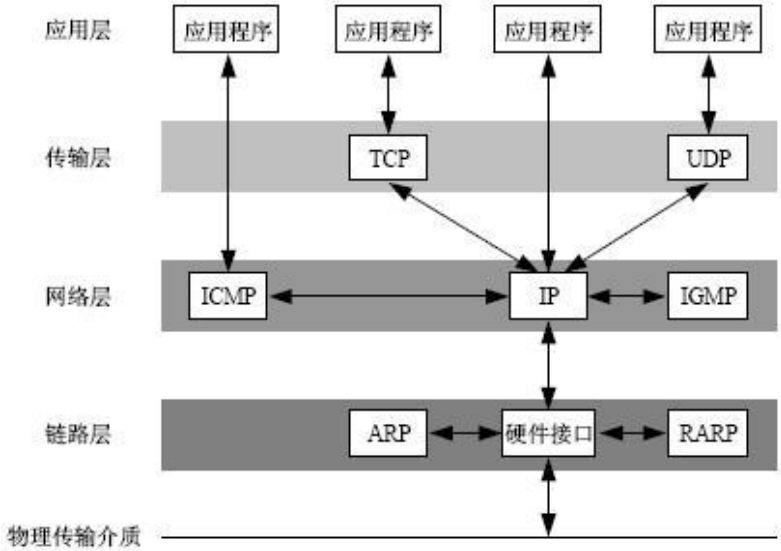
层级	作用	协议
1. 应用层	定义了如何包装和解析数据	http、ftp、dns、telnet、smtp、pop3
2. 传输层	为两台主机上应用程序提供 端到端 的通信。	TCP(传输控制协议)和UDP(用户数据报协议)

层级	作用	协议
3. 网络层	该层决定 数据 如何从发送方 路由 到接收方， 会根据网络等各种因素决定传输的 最佳路径 。	IP协议
4. 数据链路层	控制 网络层 和 物理层 之间的通信。作用： 确保在不可靠物理线路上进行数据的可靠传输。 网络层数据 会分割为可供 物理层 传输的 帧 ， 该层会额外附加上 接收方物理地址、纠错和控制信息 ， 接收方如果检测到差错，发送方会重新发送该帧数据。	ARP协议
5. 物理层	该层负责 比特流 在节点间的传输。该层的协议与链路有关， 也与传输介质有关(光纤、双绞线等)	

3、什么叫做协议?

> 1. 为实现一定功能需要双方共同遵守的规则

4、ARP和RARP协议的作用。

> 1. ARP协议是将IP地址解析为Mac地址，即硬件地址，这样就找到了唯一设备。
> 2. RARP协议相反，将物理地址映射为IP地址。
>
> 
>

5、什么是PDU

> 1. 协议数据单元PDU（ `Protocol Data Unit` ）是指对等层次之间传递的数据单位。
> 2. 应用层PDU：报文-Mesage
> 3. 传输层PDU: 数据段-Segment
> 4. 网络层PDU：数据包-Datagram，分组-Packet
> 5. 数据链路层PDU: 帧-frame
> 6. 物理层PDU: 数据位-bit

层级	协议数据单元(PDU)	备注
应用层	报文(Message)	
传输层	数据段/报文段(Segment)	TCP所使用
网络层	分组(Packet)、数据报(Datagram)	前者是IP协议使用、后者是UDP协议使用
数据链路层	帧(frame)	
物理层	bit	

6、报文(Message)是什么?

> 1. Message
> 2. 网络中交换与传输的数据单元
> 3. 包含了需要发送的完整的数据信息。
> 4. 传输过程中自上而下的封装成数据报(Datagram)和数据段/报文段(Segment)、分组(package)、帧(Frame)来传输。封装的方法就是添加一些控制信息。
> 5. 起始和目的地都是 应用层

7、数据段/报文段(Segment)是什么?

> 1. Segment
> 2. TCP使用数据段
> 3. 起始和目的地都是 传输层

8、数据报(Datagram)是什么?

> 1. Datagram:
> 2. UDP使用数据报
> 3. 起始和目的地都是 网络层

9、分组(Packet)是什么?

> 1. Packet: 用于 `IP` 协议
> 2. 在网络中传输的二进制格式的单元,
> 3. 会将报文分成多个更小的部分(Packet),并加上必要的控制信息和尾部，组成Packet
> 4. 起始和目的地都是 网络层

10、帧(frame)是什么?

> 1. frame
> 2. 数据链路层的协议数据单元
> 3. 起始点和目的点都是 数据链路层 。

11、物理层的PDU是bit

TCP(32)

1、什么是TCP/IP?

> 1. 是一类协议的集合，协议族。
> 2. 包括了tcp、ip、udp、ftp、arp、icmp、telnet等众多协议、
> 3. TCI/IP通常分为4层，
> 4. 应用层、传输层、网络层、网络接口层(数据链路层、物理层)

2、TCP是什么?

> 1. 传输控制协议
> 2. 面向连接(三次握手，四次挥手)
> 3. 数据大小无限制
> 4. 数据传输稍慢，可靠。

3、UDP是什么?

> 1. 用户数据报协议
> 2. 不建立连接
> 3. 数据大小有限制(64K)
> 4. 速度快，不可靠

4、运行在TCP协议上的协议

依赖TCP的协议	
HTTP/HTTPS	
FTP-File Transfer Protocol	文件传输协议：用于文件传输。
POP3-Post Office Protocol, version 3	邮局协议：接收电子邮件。
SMTP-Simple Mail Transfer Protocol	简单邮件传输协议：发送电子邮件。
SSH-Secure Shell	用于替代安全性差的TELNET，用于加密安全登陆用。

5、运行在UDP协议上的协议：DHCP

> 1. Dynamic Host Configuration Protocol, 动态主机配置协议
> 2. 用于动态配置IP地址。

6、TCP的传输过程

> 打开连接->写请求数据->读相应数据->关闭连接

7、Socket和TCP/UDP的关系?

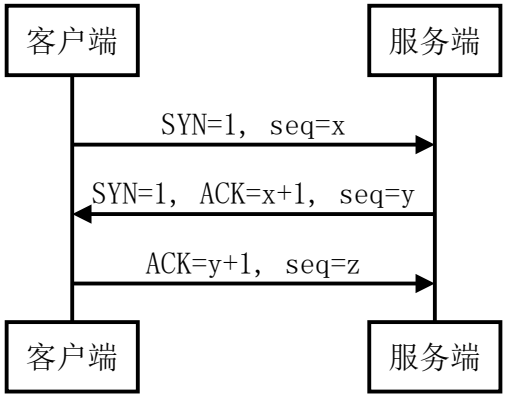
> 1. Socket是一组封装的编程调用接口。

2. 通过Socket能使用TCP、UDP进行网络通信。

三次握手

8、TCP三次握手

1. 第一次握手：建立连接后客户端发送 连接请求报文段(SYN =1, Sequence Number(seq) = x)，并等待服务端确认(客户端进入SYN_SENT状态)。
2. 第二次握手：服务端接收信息并确认后，将 SYN, ACK=(seq)x+1, seq=y 返回给 客户端 ，自身进入 SYN_RCVD状态
3. 第三次握手：客户端接收到服务端的 SYN+ACK 报文段, 将 ACK=y+1 报文段发送给服务器。发送后 服务端和客户端 进入 ESTABLISHED(连接成功) 状态。



x是生成的随机数

9、SYN是什么?

synchronize，请求同步

10、ACK是什么?

acknowledgement，确认同步

11、建立连接为什么需要三次握手？(或者为什么需要滴三次握手？)

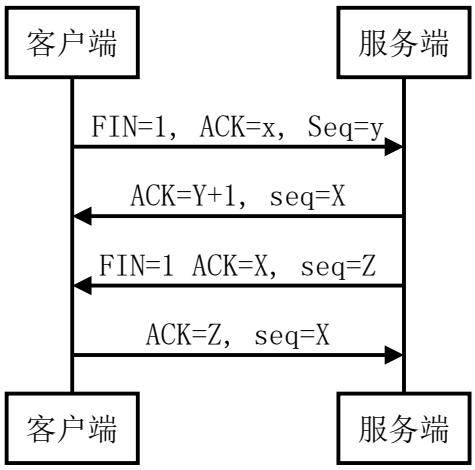
1. 前两次握手是必要的：客户端请求连接，服务端响应连接
2. 第三次握手的意义在于：客户端在发送第一次握手的连接请求后(第一次握手)，因为某些原因断开了连接。服务端之后接收到该请求，然后发送响应小心(第二次握手)，并且开始等待客户端发送数据。但其实连接已经断开了，通过第三次握手，确保连接的可靠性，减少资源浪费。

四次挥手

12、TCP四次挥手:

1. 第一次挥手：客户端向服务端发送一个 FIN报文段 并进入 FIN_WAIT_1状态 -表示客户端没有数据要发送给服务端了。
2. 第二次挥手：服务端收到后，向客户端发送 ACK报文段 。

3. 第三次挥手：服务端向客户端发送 FIN报文段 ， 请求关闭连接，自身进入 LAST_ACK状态
4. 第四次挥手: 客户端收到服务端发送的 FIN报文段 ， 并向服务端发送 ACK报文段 ， 客户端进入 TIME_WAIT状态，服务端收到客户端的 ACK报文段 后就关闭连接。客户端等待 2MSL(最大报文段生存时间) 后没收到回复就关闭连接。



13、挥手为什么比握手要多一次?

1. 挥手中的三步和握手大体是一致的。
2. 区别在于：挥手在第二步只是发送ACK，然后再发送FIN，主要是服务端可能有没有发送完的数据。在发送好之后，再发送FIN。

滑动窗口协议

14、滑动窗口协议是什么?

1. 是保证TCP的可靠传输的根本。
2. 当 发送窗口 只有接收到确认帧的时候，才会向后移动窗口，继续发送后续的帧。

协议分类

15、滑动窗口协议根据发送窗口和接收窗口大小的设定分为三个协议

协议	发送窗口	接收窗口
停止-等待协议	=1	=1

协议	发送窗口	接收窗口
后退N帧协议	>1	=1
选择重传协议	>1	>1

16、停止-等待协议

> 1. 每发一帧都要等待确认信息，才发送下一帧。
> 2. 效率很差。

17、后退N帧协议

> 1. 接收方累计接收到N帧数据后，统一发送一份确认帧给发送方窗口。用于表示N帧数据已经接收到。
> 2. 发送方规定时间内没有接收到确认信息，会发送已经收到的确认帧以后的帧。

18、选择重传协议

> 1. 出现差错，只重传出现差错的帧。
> 2. 传输效率高，减少了不必要的重传

流量控制

19、TCP流量控制的作用?

> 1. 就是为了让发送速率不要过快，也就是 发送方窗口大小 <= 接收方窗口大小
> 2. 在TCP报文段中有 窗口大小字段 来实现动态调整。

20、如果接收方没有缓存可以使用，会怎么样?

> 1. 接收方会发送 零窗口大小 的报文，让发送方将发送窗口大小设置为0，也就是 发送方停止发送数据 。
> 2. 接收方有足够缓存后，会发送 非零窗口大小 的报文。
> 3. 但是如果 非零窗口大小 报文在中途丢失，会导致发送方窗口一直为 `0` ，导致 死锁 。

21、如何解决 零窗口大小 的死锁问题?

> 1. TCP会在接收到 零窗口大小 报文后，设置一个持续计时器。
> 2. 会周期性发送一个 零窗口探测报文 给接收方，接收方会返回此时的窗口大小。
> 3. 注意：TCP规定零窗口时，也必须接收 零窗口探测报文段 、 确认报文段 和 携带紧急数据的报文段

拥塞控制

22、什么是拥塞?

> 1. 拥塞现象是指通信子网中的分组数量过多，使得该部分网络来不及处理，以致引起这部分乃至整个网络性能下降的现象，严重时甚至会出现死锁现象。

> 2. 会导致整个网络的吞吐量下降。

23、什么是吞吐量?

> 1. 分组数(package) / 秒数
> 2. 通信子网的负荷较少时，吞吐量会随着网络负荷的增加而增加。
> 3. 当网络负荷达到一定阀值后，吞吐量反而会下降。此时网络中出现了拥塞现象。

24、网络负荷是什么?

> 每个节点中分组(Packet)数目的平均值

25、拥塞控制

> 1. 防止过多的数据注入到网络中。
> 2. 保证网络的吞吐量。
> 3. 拥塞控制是对整个网络的控制，涉及到所有主机、路由器。

26、拥塞控制的代价

> 1. 需要提前知道网络内部流量分布的信息
> 2. 试试拥塞控制前，需要在各个结点之间交换信息和命令。以便选择最好的策略。
> 3. 这些操作会导致额外的开销。

慢开始+拥塞避免

27、拥塞窗口(cwnd)

> 1. 发送方 维护着一个 拥塞窗口 的变量。
> 2. 发送方的 拥塞窗口 决定了发送窗口的大小(一样大)。

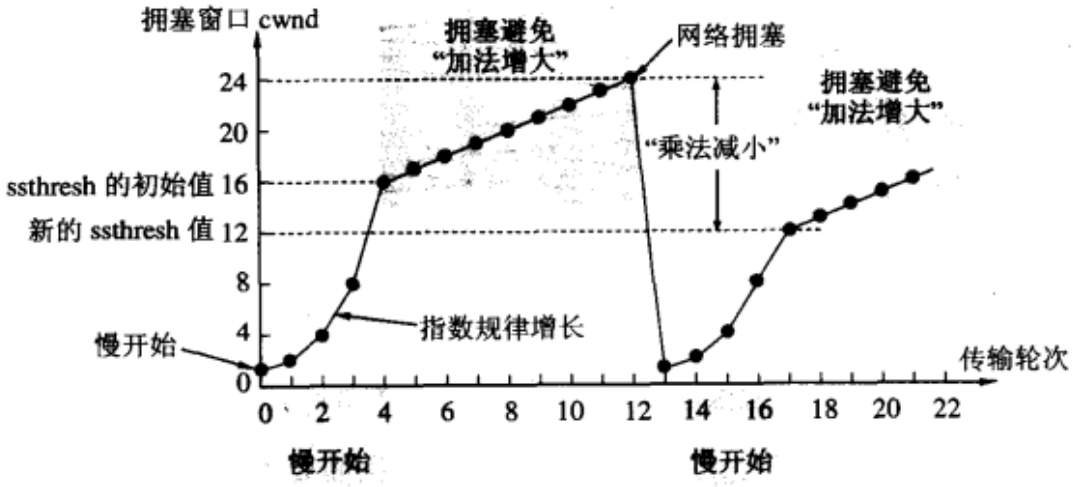


图 5-25　慢开始和拥塞避免算法的实现举例

28、慢开始

> 1. 开始传输时，从 拥塞窗口=1 开始。

2. 拥塞窗口指数增找到 慢启动的阀值
3. 然后拥塞窗口按照 拥塞避免 策略，线性增长(加法增长)
4. 达到一个极限，此时遭遇 网络拥塞
5. 此时拥塞窗口回到1，重新开始慢启动。且 慢启动新阀值 = 旧阀值 / 2 。

快重传+快恢复

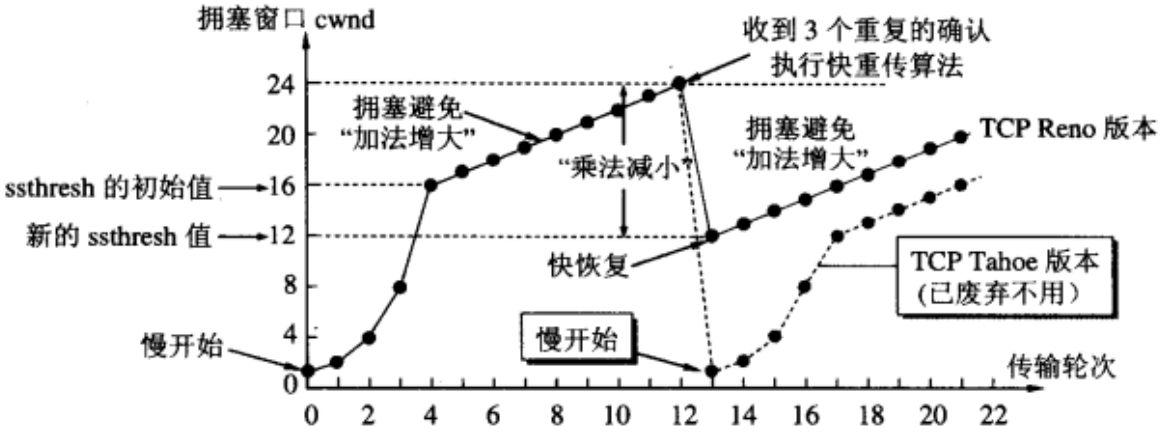


图 5-27　从连续收到三个重复的确认转入拥塞避免

29、快重传

1. 原来的重传需要等到 超时时间 后，才会重传。
2. 快重传的机制是 发送方接收到3个重复的接收方ACK ， 就会立即重传丢失的 报文段/数据段-Segment 。
3. 提高重传效率。

30、什么是发送方接收到3个重复的接收方ACK?

1. 如发送方发送了序号为 1、2、3、4、5、6 的帧。
2. 接收方无序的接收到了其中的 1、3、5、6
3. 接收方返回的ACK(确认信息)就是 2 。
4. 这种情况可能是：1-网络延迟。2-数据丢失。
5. 如果是网络延迟，过会儿就能接受到。所以接收方一般接收不到 多次的2的ACK
6. 如果是数据丢失，在接收方接收到3次重复的ACK 2 ， 会认为数据已经丢失，开始重传。

31、快恢复

1. 在出现 拥塞 时，都按照 慢启动 机制，会导致恢复过慢，且效率不高。
2. 快恢复 是指，遭遇 拥塞 后，慢启动 新阀值 = 旧阀值 / 2 ， 直接从阀值开始进入 拥塞避免 ， 而不是从1开始慢启动。

真题

32、数据链路层采用后退N帧（GBN）协议，发送方已经发送了编号为0~7的帧。当计时器超时时，若发送方只收到0、2、3号帧的确认，则发送方需要重发的帧数是（ ）。

A. 4
B. 2
C. 3
D. 5

> 答案：A
>
> 1. 接收到了0、2、3号帧的确认信息，表明0、1、2、3号帧都已经接受到(包括1号)。
> 2. 因此从3号帧后开始重发，也就是4、5、6、7 = 4个帧

Http(38)

1、 Http是什么？特点有哪些?

> 1. 超文本传输协议, 一种文本型协议。
> 2. Http是一种 应用层 的 面向对象的协议
> 3. 适用于客户端/服务器 结构
> 4. `Http` 便捷高效，适用于 分布式超媒体信息系统 。

2、 *Http的历史

> 1. Http1.1： 1997年版本，进一步完善了Http协议，当前最流行版本。
> 2. SPDY: 2009年谷歌为解决 `Http1.1`效率不高 的问题而自行研发的协议(目的在于通过 压缩、多路复用、优先级 来压缩网页的 加载时间 和 提高安全性)
> 3. Http2: 2015年发布，继承自 `SPDY` 并最终吸收了 `SPDY`协议的特性 (谷歌v587)

3、 Http协议的主要特点

> 1. 支持C/S模式
> 2. 简单快捷：请求数据只需通过 `GET`、`HEAD`、`POST` 方法，通信速度极快。
> 3. 灵活：Http允许传输 任何类型的数据对象 ，用 `Content-Type` 标记即可。
> 4. 短连接：每次连接只传输一个数据。
> 5. 无状态: `Http` 是无状态协议，
>   指 协议对事务处理没有记忆能力 。 优点：服务端应答速度块 ； 缺点：需要先前信息就需要重传之前的数据。

4、Http为什么要无状态?

> 1. 能提高服务端的响应速度
> 2. 缺点是两次请求间没有联系。

5、Http Url的格式

> `http://host[:port][abs_path]``
>
> 1. `http` : 表明是http协议
> 2. `host` : 合法的Internet主机域名或者IP地址。
> 3. `port` : 端口号

> 4. `abs_path`：请求资源的 `URL` (Web上任意可用的资源)

6、Http的请求方法有哪些？((请求行的Method字段))

> 1. GET: 请求获取 `URI` 标识的资源
> 2. POST: 在 `URI` 标识的资源后附加新的数据
> 3. PUT: 请求服务器存储一个资源(并用 `URI` 作为其标识)
> 4. HEAD: 请求由URI所标识资源的响应消息的报头
> 5. DELETE: 请求服务器删除 `URI` 所标识的资源。
> 6. TRACE: 让服务器回送收到的请求信息。多用于测试。
> 7. OPTIONS: 查询服务器的性能，或者查询资源相关的选项和需求。

7、通过GET/POST发送请求的数据的区别?

> 1. `GET` 会将请求的数据放置在 `URL` 上，会导致用户数据泄露。
> 2. `POST` 将数据放置在 `正文` 中，安全性更好。
> 3. 最好的发送数据的办法是，将信息加密后通过 `POST` 方法发送

请求报文和响应报文

8、Http的报文什么特点?

> 1. Http报文是面向文本的
> 2. 报文中每个字段都是一些ASCII码串，长度不确定

9、HTTP请求报文的组成(4个)

名称	组成
请求行	请求方法(Get/Post)、请求路径Url、协议版本等
请求报头(Headers)	header, 包括各种字段
空行	
请求体	发送的数据

10、HTTP相应报文的组成(4个)

名称	组成
状态行	状态码(200等)、协议版本等
响应报头(Headers)	header, 包括各种字段
空行	
响应正文	响应的正文数据

通用报头

11、Http的通用报头的主要组成

> 请求报文头和响应报文头都会使用

Header	含义	备注
Date	消息产生的事件	
Cache-Control	用于指定缓存指令	
Connection	连接的管理。	可以指定连接是连续的，或者指定 `close` 。 close会在响应完成后，关闭连接。
Transfer-Encoding	报文主体的传输编码方式	

请求报头

12、Http的请求报头的主要组成

Header	含义
Host	请求资源所在服务器
User-agent	请求方的浏览器类型、操作系统等信息。 如 `Oppo R9s Build`
Accept	用于指定客户端接收哪些类型的信息, 如 `text/css`
Accept-Encoding	客户端可以识别的数据编码, 如 `gzip, deflate`
Accept-Language	浏览器所支持的语言类型，如 `zh-CN,en-US`
Connection	如果是 `keep-alive` 表示客户端和服务端保持连接

13、如果本次请求希望服务端返回的数据使用gzip压缩该怎么办?

> 请求报头：Header的 `Accept-Encoding` 字段指明使用 `gzip`

响应报头

14、响应报头的主要部分

Header	含义
Location	令客户端重新定向到的URI
Server	服务器用来处理请求的软件信息

实体报头

15、实体报头

Header	含义	备注
Content-type	实体正文的媒体类型	如 `text/css`
Content-Length	正文长度	如 `1534`
Content-Encoding	编码类型	如 `gzip` ，表示: 想要获取到 `Content-type：text/css` 类型的实体内容， 需要通过 `gzip解码`
Last-Modified	资源最后的修改日期	如 `Tue,13 Mar 2018 02:59:28 GMT`
Expires	响应过期的日期和时间	如 `Wed,13 Mar 2019 03:02:26 GMT`

16、如果本次请求的内容想使用gzip压缩该怎么办?

> 实体报头：Header的 `Content-Encoding` 字段指明使用 `gzip`

17、Accept-Encoding和Content-Encoding使用gzip的区别?

> 1. Content-Encoding: 是本次请求或者响应，内容的文本采用了 `gzip` 压缩。
> 2. Accept-Encoding: 表示这次请求希望服务端返回的数据采用 `gzip` 压缩。

keepalive

18、Http的keepalive connections机制的作用

> 1. 如果有大量连接，tcp的三次握手和四次挥手会导致性能低下。
> 2. Http会在传输完成后仍然保持连接，再次请求数据时，能复用之前空闲下来的连接，提高性能。

常见返回码

19、状态码分类(5类)

> 1. 1xx属于通知类
> 2. 2xx属于成功类
> 3. 3xx属于重定向类
> 4. 4xx属于客户端错误类
> 5. 5xx属于服务端错误类

20、HTTP返回的常见错误信息

> 1. `400~499`：客户端错误，或者请求无法实现。

2. `500~599`：服务器错误，服务器不能实现合法的请求。
3. 200：请求被正常处理
4. 301：永久性重定向
5. 302：临时重定向
6. 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
7. 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
8. 307：临时重定向，与302类似，只是强制要求使用POST方法
9. 400：请求报文语法有误，服务器无法识别
10. 401：请求需要认证
11. 403：请求的对应资源禁止被访问
12. 404：服务器无法找到对应资源
13. 500：服务器内部错误
14. 503：服务器正忙

21、204的作用

1. 表响应报文中包含若干首部和一个状态行，但是没有实体的主体内容。
2. 使用场景：对于一些提交到服务器处理的数据，只需要返回是否成功的情况下，可以考虑用状态码204来作为返回信息，从而省略多余的数据传输。

22、205的作用

1. 205则是告知浏览器清除当前页面中的所有html表单元素，也就是表单重置。

Http的缓存机制

23、Http的缓存机制是如何实现的?

主要是通过Header中的两个字段来实现：

1. Cache-control
2. ETag

Cache-control

24、Cache-control的作用?

用于设置缓存策略

25、Cache-control的主要字段

字段	功能
private	只有客户端可以缓存
public	客户端和代理服务器都可以缓存

字段	功能
max-age	缓存过期的时间。如果缓存没有过期，则直接使用缓存。
no-cache	需要让服务端对比数据是否更新过来验证缓存是否有效。就算max-age没有过期，依旧需要发送一次请求向服务端确认数据是否更新。如果数据更新过，返回304；否则返回新数据，与ETag相配合
no-store	所有内存都不进行缓存

ETag

26、ETag是什么?

> 1. 用来对比缓存，是服务端资源的标识码。
> 2. 第一次请求资源时，会额外返回一个Header ETag(哈希值，通过资源计算得到)。
> 3. 第二次请求资源时，会通过header `if-None-Match` 将该ETag发给服务端。
> 4. 服务端比对 `if-None-Match` 和资源的Hash值一致，表明没有更新过数据，返回304。
> 5. 服务端比对结果不一致，表明数据需要更新，返回新资源和返回码200.

27、在第二次请求资源时将ETag的数值通过ETag header就可以提交给服务端?

> 错误!
> 需要通过 `if-None-match` 提交给服务端。

Android中的使用

28、HttpClient和HttpURLConnection的特点和区别

> 1. `HttpClient` 已经被废弃和移除(Android 6.0)
> 2. `Android 2.3`以前 适合使用 `HttpClient`，包括`2.3`及其以后的版本 适合使用 `HttpURLConnection`
> 3. `HttpURLConnection` 具有 压缩和缓存机制 ， 可以有效减少网络访问的流量。

29、HttpClient的特点

> 1. 相比于HttpUrlConnection更加高效简洁
> 2. 结构过于复杂，维护成本高。
> 3. Android 5.0后被废弃

30、HttpURLConnection的特点

> 1. Android 2.2前：存在调用close()函数会影响连接池，导致连接复用失效的重大BUG
> 2. Android 2.2后：默认开启了gzip压缩，并且提高了Https的性能
> 3. Andorid 4.4后：底层实现已经被OkHttp替换。

Cookie

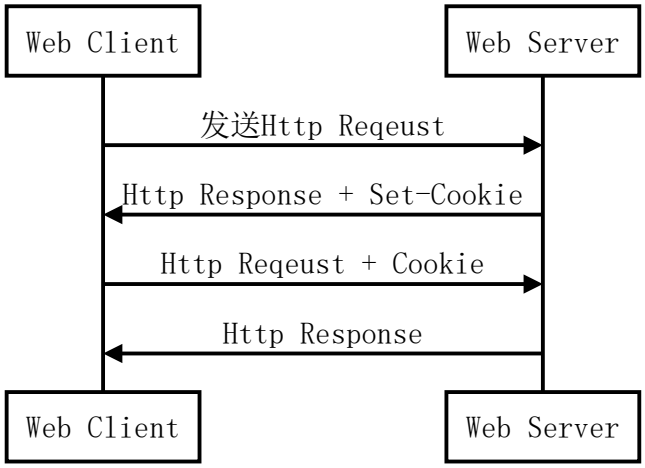
31、Cookie是什么?

> 1. Cookie 用于浏览器使用 Http协议 时与服务器保持活动状态(保活)
> 2. Cookie 文件保存在本地(具有不安全性)
> 3. Cookie 能保存例如 用户账号/密码 等信息
> 4. Cookie 具有时效性：有 临时性Cookie(超时清除，存储在app内存中) 和 持续性Cookie

32、Cookie机制定义了两种报头

> 1. Set-Cookie报头：包含于服务端的响应头中(ResponseHeader)
> 2. Cookie报头：包含于客户端的请求报文的请求头中(RequestHead)

33、Cookie在Http中的使用流程



> 1. 前两步：是第一次请求数据，并且获取cookie(主要是登陆等功能)
> 2. 后两步：是用该cookie去保持状态，请求其他数据。

Android开发中问题场景

34、Anndroid开发中的实际问题

> 1. 场景：加载WebView中的H5页面需要是已经登录状态。需要将Android原生页面登陆后得到的jsessionid写入到WebView的Cookie中。
> 2. 问题描述：Cookie始终没有被带上，导致请求验证失败。
> 3. 解决办法：

```
CookieManager cookieManager = CookieManager.getInstance();
if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP){
    // >= API21 需要手动开启
    cookieManager.setAcceptThirdPartyCookies(mWebView, true);
}else{
    cookieManager.setAcceptCookie(true);
}
```

Session

35、Session是什么？特点?

1. Session 是用于在 服务端 多个页面中具有 关联性 -例如：保持登录状态等(Http 是无状态的)
2. Session 轻量级，能避免 服务器 为了具有 关联性 ， 而去查找 数据库 等操作。
3. Session 就像是 服务端的一个全局信息
4. Session 保存在 服务器 ， 具有 安全性
5. Session 依赖于 Cookie - 本地客户端 需要用Cookie保存 Session ID
6. Session 具有时效性，例如 浏览器 关闭会导致 Session 失效(本地存储SessionID的Cookie丢失导致的)，或者服务端设置了失效时间。

36、Cookie与Session区别

1. Cookie保存在客户端浏览器，Session保存在服务端(客户端会通过Cookie来保存 SessionID)
2. Session适合保存登录信息等安全性较高的信息
3. Cookie保存其他不重要的信息

Token

37、Token是什么？特点?

1. 令牌-用户身份的验证方式
2. 用户注册或登陆后，服务器会生成Token并存在服务端数据库中，并会把 Token 返回给 客户端
3. 客户端请求时都会附带上 Toekn
4. 服务端接收到 Token 会与服务端 Toeken 对比，1-值相同，则表示用户处于登录状态(或者验证是否是合法用户) 2-值不相同，则表示登陆过期需要重登。

38、Tooken与Session的区别

1. Session 主要是解决 Http无状态 的问题
2. Tooken 主要用于 身份认证 ， 也可以包含一些参数来用保持状态(登录状态等)

Https(18)

1、Https是什么?

1. Https = Http + SSL
2. HTTPS是使用TLS/SSL加密的HTTP协议

2、Http和Https有什么区别?

1. HTTP（Hypertext Transfer Protocol）超文本传输协议, 采用明文传输信息，具有安全隐患。
2. HTTPS(Secure Hypertext Transfer Protocol) 安全超文本传输协议: 使用安全套接层协议(SSL)进行信息交换，更安全。
3. HTTP基于 TCP

4. Https基于 `SSL/TLS`，SSL/TLS基于 `TCP`
5. Http的端口是 `80`； Https的端口是 `443`

SSL

3、SSL是什么?

1. Secure Socket Layer，是安全套接字层协议(+传输层协议 `TLS`）
2. 层级：位于 `TCP/IP` 协议和应用层协议之间。
3. 为数据通讯提供安全支持，是目前使用最广泛的协议。

4、TLS是什么

1. Transport Layer Security，传输层安全协议
2. 在两个通信应用程序之间提供保密性和数据完整性
3. 该协议由两层组成： TLS 记录协议（TLS Record）和 TLS 握手协议（TLS Handshake）。
4. 底层的是TLS 记录协议，位于某个可靠的传输协议（例如 TCP）上面。

SSL证书

5、SSL数字证书(SSL证书)是什么?

1. 一种遵守SSL协议的数字证书。
2. 因为配置在服务器上，也称为SSL服务器证书。
3. 由资深的数字证书颁发机构CA(例如：GlobalSign等机构)，在验证服务器身份后颁发。

Android开发中问题场景

6、Google Play上架问题

1. 提示：SSL Error Handler
2. 原因：SSL过期，导致报错，为了避免安全问题，需要进行有效处理
3. 解决办法：找到WebViewClient按照规范处理onReceivedSslError(让用户选择)。

```
public void onReceivedSslError(WebView view, SslErrorHandler handler, SslError error) {
                final SslErrorHandler mHandler ;
                mHandler= handler;
                AlertDialog.Builder builder = new AlertDialog.Builder(activity);
                builder.setMessage("ssl证书验证失败");
                builder.setPositiveButton("继续", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        mHandler.proceed();
                    }
                });
                builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        mHandler.cancel();
                    }
                });
                builder.setOnKeyListener(new DialogInterface.OnKeyListener() {
                    @Override
                    public boolean onKey(DialogInterface dialog, int keyCode, KeyEvent event) {
                        if (event.getAction() == KeyEvent.ACTION_UP && keyCode == KeyEvent.KEYCODE
                            mHandler.cancel();
                            dialog.dismiss();
                            return true;
                        }
                        return false;
                    }
                });
                AlertDialog dialog = builder.create();
                dialog.show();
            }
```

CA

7、CA是什么?

> 1. 证书授权中心(CA, Ceritificate Authority)
> 2. CA负责签发证书、认证证书、管理证书
> 3. CA会公开对应的公钥，私钥放在服务器上。

8、CSR是什么?

> 1. 证书请求文件（Certificate Signing Request）
> 2. 申请数字证书时，CSP（加密服务提供者）会生成 私钥 和 证书请求文件 .
> 3. 申请者将CSR交给CA后，CA使用CA机构的私钥进行签名，就生成了 证书公钥文件 。该公钥就是交给用户的公钥。

对称加密算法

9、Https的数据传输是如何保障安全的？对称算法的是什么?

1. Https中数据的传输使用 对称加密算法
2. 对称加密算法 是指双方都是使用同一个 密钥s 进行加密解密
3. 服务器端 和 多个客户端 之间都采用的不同 对称加密算法
4. 对于约定使用 哪个对称加密算法 对于约定的过程就需要使用 非对称加密算法 进行加密。

非对称加密算法

10、非对称加密算法是什么？有哪些特点?

1. 服务端具有唯一私钥 ， 多个客户端 具有 公钥
2. 私钥 加密的内容，所有 客户端 都可以用 公钥 进行解密
3. 公钥 加密的内容，只有 服务端唯一的私钥 解密
4. 特点是单向安全 --- 客户端->服务端
5. 公钥 需要服务器发送给 客户端 ， 这个过程可能会被调包。

调包

11、公钥被掉包问题

1. 中间人 可以和 服务端 之间有合法的 公钥和私钥
2. 中间人 将 假公钥 发给 客户端 ， 这样 客户端 发送的内容， 中间人 可以用 假私钥 进行解密并篡改，然后使用 真公钥 发送给 服务端
3. 服务端 发送给 中间人 的信息也被会解密，篡改后用 假私钥 加密发送给 客户端

12、非对称加密算法中公钥如何防止被掉包?

1. 使用 第三方机构 的 私钥 对 服务端公钥 加密后，再发送给 客户端 (并将 数字证书(第三方机构的公钥) 一并发送给客户端)。
2. 客户端 用 第三方机构提供的公钥 进行解密
3. 中间人 用 假私钥 加密的内容，是无法用 第三方机构的公钥解密的 ， 因此保证了 公钥获取 的安全性。
4. 安全隐患 在于 中间人 也可以去申请 合法的数字证书 ， 对 数字证书 进行掉包。

数字证书

13、数字证书的作用

1. 保护 非对称加密算法 的 公钥 的安全问题
2. 数字证书由第三方机构CA颁发
3. 数字证书的内容就是明文的 服务端公钥 + 数字签名
4. 服务端和客户端通信，不再是 内容 + 服务器公钥 ， 而是 内容 + 数字证书

14、没有数字签名的数字证书的安全隐患?

1. 数字证书可以被调包。
2. 本质是假服务器将假数据(假私钥进行加密)发送给客户，并且给他一个假的数字证书(假公钥)。
3. 最终客户端拿的还是假的非对称加密算法的公钥。

数字签名

15、数字签名的作用

> 1. 保证 数字证书 内容的唯一性，不可篡改性。
> 2. 数字签名 制作过程：对 数字证书 上的明文内容(服务器公钥 +其他内容)进行 Hash 加密，生成不可逆的hash值H，再对 H 使用 第三方机构私钥 进行加密，最终生成 数字签名
> 3. 保证了 数字签名 无法篡改， 数字证书的明文内容 无法篡改(客户端用同样的方法去生成 数字签名 ，与证书上签名进行比对，只要不一样就表示 数据被篡改)

16、数字签名的制作过程

17、数字证书为什么不会被篡改?

> 1. 如果篡改了数字证书上的 服务器公钥 ，会导致用 第三方机构公钥 解密 数字签名 后得到的Hash值，和假公钥产生的Hash值不一致。 被篡改!
> 2. 如果篡改了数字证书上的 数字签名 ，假数字签名是使用 假第三方机构私钥 (拿不到CA的私钥)来生成的，因此用 第三方机构公钥 是无法解密的。 被篡改!
> 3. 如果两者都篡改，因为 篡改数字签名 就会导致失败，所以也能判断出 被篡改!

18、第三方机构的公钥为什么不会被篡改?

> 1. 系统和浏览器 维护着一个 第三方机构列表-包括它们的公钥
> 2. 第三方机构 直接从 本机上 获取的，因此 中间人 没有机会去 调包第三方机构 ， 除非系统层面被攻破
> 3. Fiddler 抓包的原理是 在系统中安装合法的fiddler证书(具有假的第三方机构公钥) ，对 我们通讯所用的数字证书掉包 后，我们会使用 Fiddler的第三方机构公钥 进行解密。发现不了 被篡改 ！

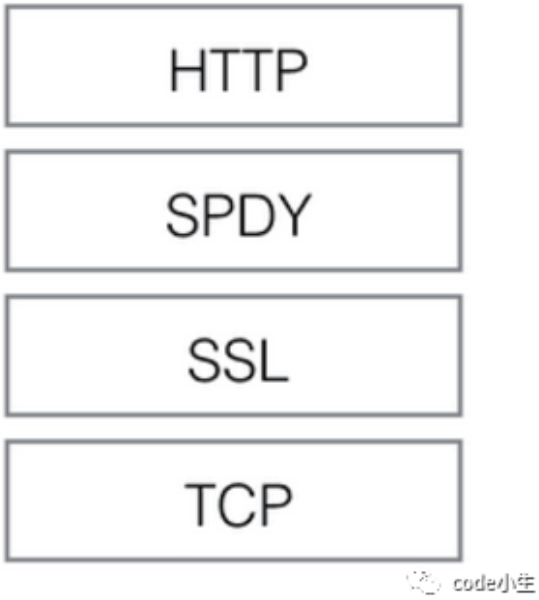
Http 2.0(11)

SPDY

1、SPDY是什么?

> 1. Google在Http1.1上进行了优化产生了SPDY协议
> 2. 降低了延迟： 多路复用通过共享一个tcp连接的方式，降低了延迟，提高了利用率。
> 3. 请求优先级：因为多路复用，会同时有多个请求。给重要请求设置优先级，保证优先响应。例如首页的html内容应该优先显示，之后才是去加载各种静态资源。
> 4. header压缩： 有效减少包的大小和数量。
> 5. 基于https的加密协议传输
> 6. 服务端推送：如访问一个SPDY页面，客户端收到css数据的同时，服务端还会将其js文件推送给服务器。这样后续去获取js的时候，额可以直接从缓存读取。

2、SPDY构成图



1. SPDY 在 SSL和Http之间。能够青松建荣Http1.1等老版本。

Http 2.0

3、Http 2.0协议是什么?

1. SPDY的升级版，但也有不同之处
2. 相对于Http 1.x具有巨大的提升
3. OkHttp支持配置使用Http 2.0协议

4、Http2.0和SPDY的区别

1. Http2.0支持明文Http传输
2. SPDY强制使用HTTPS
3. Http2.0消息头的压缩算法采用 `HPACK`
4. SPDY消息头的压缩算法采用 `DEFLATE`

5、Http2.0相对于Http1.x的惊人性能提升

1. 有公司专门对Http2.0的性能进行专业测试。Http2.0的性能比Http1.0提升8倍之多。
2. 场景：同时请求400张图片。

	Http1.1	**Http2.0**
Latency(延迟)	3ms	6ms
Load time(加载时间)	14.70s	1.61s

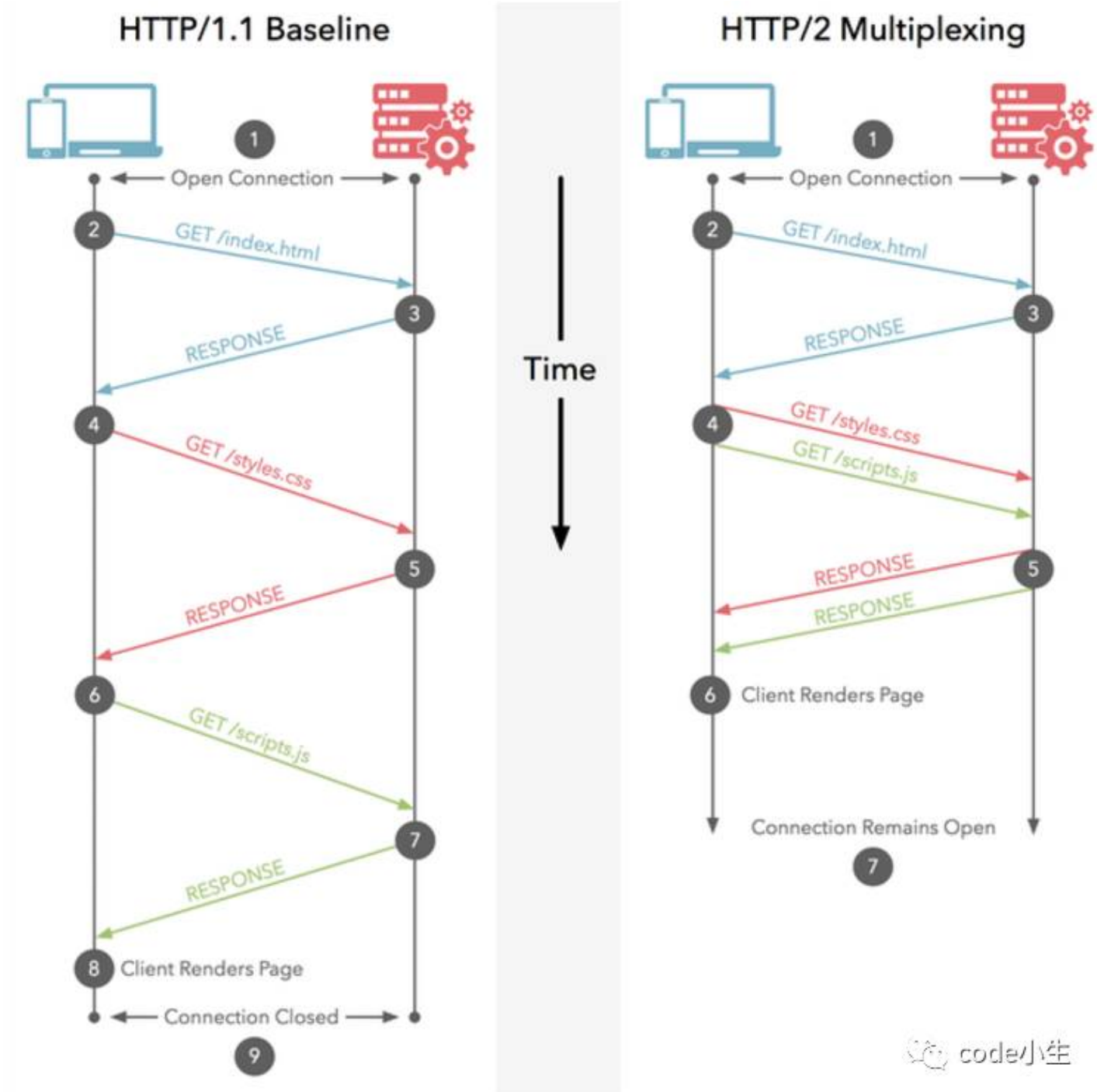
Http2.0和Http1.x

6、Http 2.0和Http 1.x的区别

1. http2.0是二进制格式；http1.x是文本协议。
2. http2.0支持多路复用；http1.x是长连接复用，http1.0是一个请求一个连接。
3. http2.0支持header头部压缩。
4. http2.0支持服务端推送。

7、Http2.0多路复用和Http1.x长连接服用的区别?

1. HTTP1.0: 一个请求一个连接，用完就关闭。
2. HTTP1.X长连接复用：和服务器具有长连接，在第一个请求并且受到响应后，发送第二个请求，并接受响应。然后发送第三个请求...如果前面的请求出现了阻塞，会阻塞后续请求。
3. HTTP2.0多路复用: 并发的在一个TCP上发送多个请求，不是HTTP1.X那种单线的操作。能提高响应速度，也不会出现阻塞。



二进制协议

8、Http 2.0的二进制格式

> 1. http2.0是二进制协议
> 2. 以 帧 为基本单位
> 3. 一帧中包含了 数据 和 `Stream Identifier` (该帧的标志，标识了该帧属于哪个request)

多路复用

9、Http2.0的多路复用

> 1. 多个请求共用一个TCP连接，可以在该TCP上进行并发请求。
> 2. 性能高，消耗少。

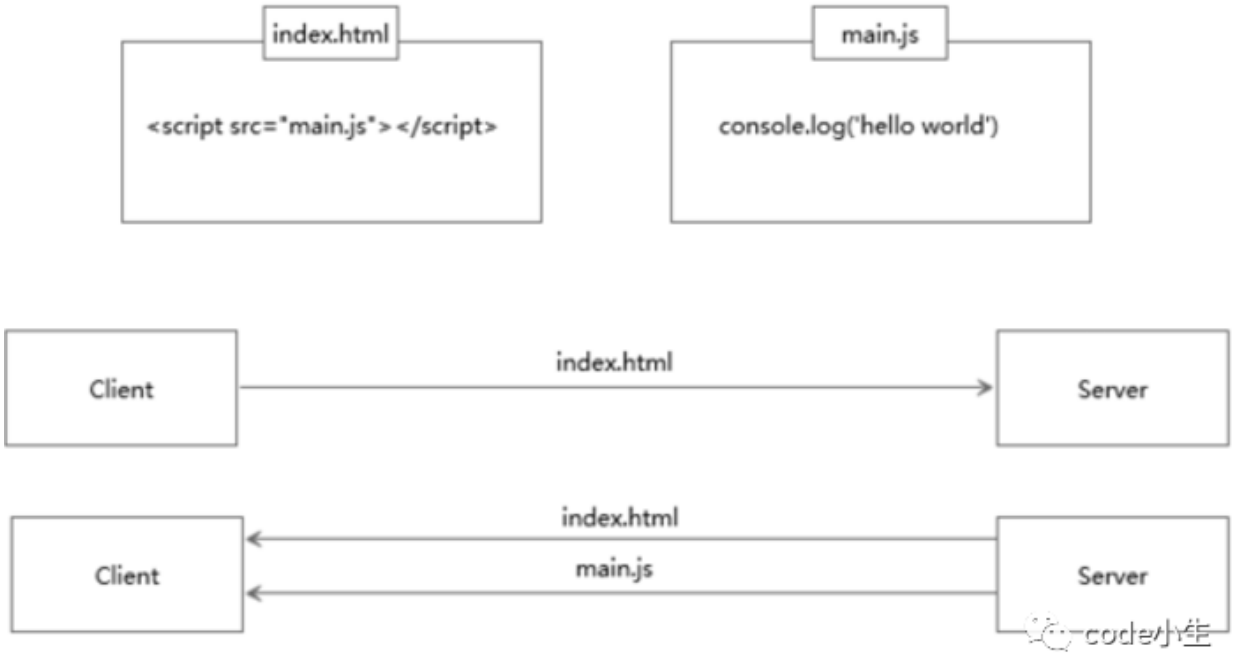
10、如何实现多个请求可以在一个TCP连接上并发?

> 1. Http2.0的二进制分帧实现的。
> 2. 每一帧数据都有一个身份标识。
> 3. 多个请求的不同帧可以无序、并发的发送出去。
> 4. 服务端会根据每一帧的身份标识，将其整理到对应的request中。

服务端推送

11、服务端推送是什么?

> 1. 服务端推送就是服务端把客户端所需要的资源跟着请求(如html)一起发送到客户端，省去了客户端重复的请求步骤。静态资源通过服务器推送，能极大提升速度。
> 2. 旧版操作流程:
>   1. 客户端请求html，服务端返回html
>   2. 客户端请求js，服务端返回js
> 3. 新版服务端推送:
>   1. 客户端请求html, 服务端返回html和相关js



补充题(2)

1、Http为什么是无状态的？为什么设计成一种无状态协议？如何解决无状态的问题？

> 1. Http对于事物处理，没有记忆能力。因此是无状态的。
> 2. 无状态意味着响应速度快，节省服务器资源。
> 3. 通过Cookie和Session能解决。

2、Http的请求流程

> 1. 域名解析
> 2. 三次握手建立TCP连接
> 3. 发送HTTP请求
> 4. 获取到响应信息
> 5. 四次挥手切断TCP连接

知识储备环节(22)

1、为什么要用使用网络请求开源库?

> 1. 不使用网络请求框架时，网络请求交互很复杂：需要考虑到线程池、缓存等一系列问题。
> 2. 异步请求
> 3. 线程池
> 4. 缓存
> 5. 降低开发难度
> 6. 缩短开发周期
> 7. 使用方便

2、ResetFul Api

> 1. 主要是用于多种前端(`Android\ios\web`)请求服务器时(同一种功能)，能使用同一个 `api接口` 而不是使用很多接口。

3、Chrome inspect工具可以调试Anfroid的WebView

4、域名解析是什么?

> 1. DNS(Domain Name Service, 域名服务)，用于域名解析查找IP地址。
> 2. 运行在TCP和UDP协议上。

5、从浏览器访问页面的域名解析的流程?

> 1. 搜索浏览器自带的DNS缓存
> 2. 搜索操作系统自带的DNS缓存
> 3. 读取hosts文件，去获取。
> 4. 向本地配置的首选DNS解析服务器发起域名解析请求。

IP

6、IP协议

> 1. IP协议是网络层协议。
> 2. TCP/IP协议中每个网络适配器都有一个唯一的IP地址。
> 3. IP地址是一个32位地址，每8个二进制位一段，一般每段用十进制显示： `192.168.0.1`

7、IP地址的两部分

> 1. 网络ID
> 2. 主机ID
> 3. 但是没有具体规定哪部分属于网络ID，哪部分属于主机ID

8、IP地址的主要分类

> - A 类地址：1.0.0.1 - 126.255.255.254
> - B 类地址：128.1.0.1 - 191.255.255.254
> - C 类地址：192.0.1.1 - 223.255.255.254

9、如何区分一个IP地址是哪类地址?

> 1. 如果 32 位的 IP 地址以 0 开头，或者十进制第一段为0~127，那么它就是一个 A 类地址。
> 2. 如果 32 位的 IP 地址以 10 开头，或者十进制第一段为128~191，那么它就是一个 B 类地址。
> 3. 如果 32 位的 IP 地址以 110 开头，或者十进制第一段为192~223，那么它就是一个 C 类地址。

IP 地址分类规则				
类型	二进制前几位	十进制第一段	举例	排除地址
A类	0	0-127	126.10.10.10	10.0.0.0-10.255.255.255 127.0.0.0-127.255.255.255
B类	10	128-191	130.100.0.1	172.16.0.0-172.31.255.255
C类	110	192-223	195.10.10.10	192.168.0.0-192.169.255.255

10、主机ID全为0

> 1. 主机ID全为0代表网络本身
> 2. 例如: 130.100.0.0，指的是ID为130.100的B类地址
> 3. 例如: 126.0.0.0，指的是ID为126的A类地址

11、主机ID全为1

> 1. 知己ID全为1代表广播，用于向网络中所有主机发送广播的。
> 2. 例如: 130.100.255.255，指的是ID为130.100的B类地址
> 3. 例如: 126.0.0.0，指的是ID为126的A类地址

12、环回地址

> 1. 以十进制 127 开头的地址都是环回地址。目的地址是环回地址的消息，其实是由本地发送和接收的。
> 2. 主要是用于测试 TCP/IP 软件是否正常工作。
> 3. 一般用的环回地址是 127.0.0.1

子网掩码

13、网络ID和主机ID在A、B、C类地址中的划分

> - A 类地址：IP 地址的前 8 位代表网络 ID，后 24 位代表主机 ＩＤ。
> - B 类地址：IP 地址的前 16 位代表网络 ID，后 16 位代表主机 ＩＤ。
> - C 类地址：IP 地址的前 24 位代表网络 ID，后 8 位代表主机 ＩＤ。

14、A、B、C类地址默认的子网掩码

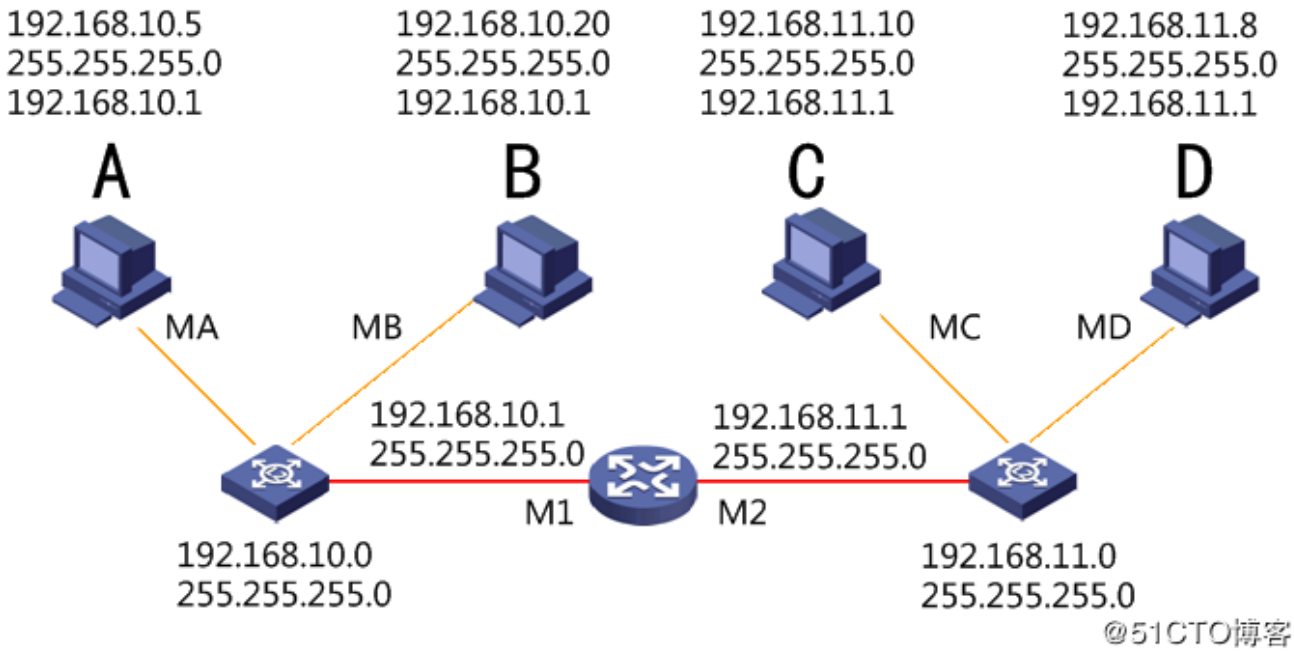
> A: 255.0.0.0
> B: 255.255.0.0
> C: 255.255.255.0

15、子网掩码有什么用?

> 1. 子网掩码必须和IP一起协同工作。
> 2. 只有通过子网掩码，才能表明一台主机所在的子网与其他子网的关系，使网络正常工作。
> 3. 多个家庭网络组成了小区级别的子网，这些子网再组成更大的子网。
> 4. 子网掩码可以将 网络ID 和 主机ID 分离，先判断是否处于同一个网段，如果相同，那么可以把数据包直接发送到目标主机，否则就需要路由网关将数据包转发送到目的地。

16、实例一: 子网间的通信

> 1. A和B能直接通信，C和D能直接通信。
> 2. A和C，B和D不能直接通信。
> 3. A主机要与B主机通信，A和B各自的IP地址与A主机的子网掩码进行And与运算，再比较 网络ID(网络地址) 是否相同：
>   1. 结果如果相同，则说明这两台主机是处于同一个网段，这样A可以通过ARP广播发现B的MAC地址，B也可以发现A的MAC地址来实现正常通信。
>   2. 如果结果不同，ARP广播会在本地网关终结，这时候A会把发给B的数据包先发给本地网关，网关再根据B主机的IP地址来查询路由表，再将数据包继续传递转发，最终送达到目的地B。



17、实例二：跨子网通信

A：10.1.1.2/24
网关：10.1.1.1/24
D：8.8.8.8

> 1. 当A试图访问D时，用24位掩码来按位& 8.8.8.8
> 2. 得到网段8.8.8，和自己的网段10.1.1不相同，表明需要网关(代理介入)
> 3. 会把发给8.8.8.8的包发送给网关(代理)
> 4. 网关会依据8.8.8.8来查询路由表，将包发到上游的Internet路由器上，最终到达目的地。

18、计算机的网关是什么?

> 1. 计算机的网关(GateWay)就是到其他网段的出口，也就是路由器接口IP地址。
> 2. 路由器接口使用的IP地址可以是本网段中任何一个地址，不过通常使用该网段的第一个可用的地址或最后一个可用的地址，这是为了尽可能避免和本网段中的主机地址冲突。

19、公网IP需要填写子网掩码吗？是否可以随便填写?

> 1. 必须要填写，不可以随便填写。
> 2. 公网IP是 `ISP` 提供的用于接入他们网络的地址, 再通过他们的网络将你的数据包转发出去。
> 3. 本质是本身先和 `ISP` 接成了一个网络，显然在这个网络中必须要有子网掩码。
> 4. 全球的网络就是大大小小的子网拼出来的，是网络就有子网掩码。

私有地址

20、私有地址

> 1. REC 1918留出了3块IP地址空间（1个A类地址段，16个B类地址段，256个C类地址段）作为私有内部地址。
> 2. 这些范围内的IP地址不能被路由到Internet骨干网上。
> 3. Internet路由器将丢弃该私有地址。

类型	IP范围	子网数量
A类	10.0.0.0到10.255.255.255	1
B类	172.16.0.0到172.31.255.255	16
C类	192.168.0.0到192.168.255.255	256

21、如何让私用地址能连接到Internet?

> 1. 使用私有地址将网络连至Internet，需要将私有地址转换为公有地址。
> 2. 这个转换过程称为网络地址转换(Network Address Translation，NAT)
> 3. 通常使用路由器来执行NAT转换。

22、除了3种私有地址，其他的地址也可以用作内网吗?

> 1. 都可以, 任何A、B、C类地址都可以作为私网使用。
> 2. 就算是合法的公网ip也可以拿来用，这种效果是通过NAT技术实现的。
> 3. 缺点：在需要访问该公网ip时，会直接跳转到内网。

参考资料

1. Android Https相关完全解析 当OkHttp遇到Https
2. TCP流量控制
3. TCP窗口滑动以及拥塞控制

4. SPDY协议是什么?
5. HTTP1.0、HTTP1.1 和 HTTP2.0 的区别
6. 也许，这样理解HTTPS更容易
7. 面试时，你被问到过 TCP/IP 协议吗?
8. HTTP面试题都在这里
9. IP地址和子网划分学习笔记之《子网掩码详解》

临时

1. 深入浅出Retrofit
2. 主流网络请求开源库的对比
3. Retrofit的使用、优势与源码分析