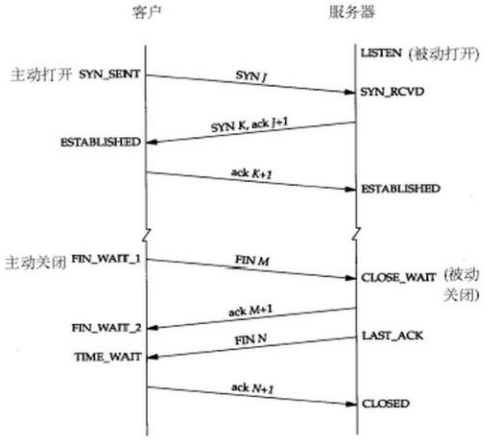
来源：牛客面经/Java 开发

四次挥手（完美）：
//网络重定向，流程（招银科技）：
TCP（美团）：
TCP 四次挥手的原因？TIME_WAIT 等待超时会怎样?）（京东）：
Raft 协议的 leader 选举（京东）：
输入 www.baidu.com 会发生什么？（网易）CDN、负载均衡：
TCP3 次握手四次挥手（百度）：
HTTP 报文格式（华为）
HTTP 与 HTTPS 的区别（华为）：
服务器与 app 之间怎么加密传输（华为）：
HTTP 状态码（网易）
TCP/UDP 区别，SYN 攻击（TX）
网络攻击手段，如何预防（TX）
DNS 均衡（乐视）
阻塞解决（京东）
TCP/IP 几层，每层含义（七牛云）
DNS 解析流程（七牛云）
客户端出故障，服务器会如何处理？
应用层如何在保证不在运输层多取数据？
IP 地址的合法性
HTTP 请求流程
HTTP 长连接设置
HTTP 怎么记住状态
怎么防止网络被拦截
HTTPS 工作原理
SSL 原理
TCP 洪水攻击
Session 原理
端点续传原理
Socket 通信原理



TCP 三次握手

第一次握手:

客户端发送一个 TCP 的 SYN 标志位置 1 的包指明客户打算连接的服务器的端口，以及初始序号 X,保存在包头的序列号(Sequence Number)字段里。

第二次握手:

服务器发回确认包(ACK)应答。即 SYN 标志位和 ACK 标志位均为 1 同时，将确认序号(Acknowledgement Number)设置为客户的 I S N 加 1 以.即 X+1。

第三次握手.

客户端再次发送确认包(ACK) SYN 标志位为 0,ACK 标志位为 1.并且把服务器发来 ACK 的序号字段+1,放在确定字段中发送给对方.并且在数据段放写 ISN 的+1

SYN 攻击/TCP 洪水攻击

Syn 攻击就是 攻击客户端 在短时间内伪造大量不存在的 IP 地址，向服务器不断地发送 syn 包，服务器回复确认包，并等待客户的确认，由于源地址是不存在的，服务器需要不断的重发直 至超时，这些伪造的 SYN 包将长时间占用未连接队列，正常的 SYN 请求被丢弃，目标系统运行缓慢，严重者引起网络堵塞甚至系统瘫痪。一般较新的 TCP/IP 协议栈都对这一过程进行修正来防范 Syn 攻击，修改 tcp 协议实现。主要方法有 SynAttackProtect 保护机制、SYN cookies 技术、增加最大半连接和缩短超时时间等. 但是不能完全防范 syn 攻击

网络攻击手段及预防

（1） DOS 攻击/洪水攻击

（2） ARP 攻击 ARP 欺骗是通过 MAC 翻译错误造成计算机内的身份识别冲突，表现为网络提示连接出现故障，IP 冲突，无法打开网页，频繁弹出错误对话框

（3） 脚本攻击（SQL 注入）就是利用现有应用程序，将(恶意)的 SQL 命令注入到后台数据库引擎执行的能力，这种攻击脚本最直接，也最简单，当然，脚本攻击更多的是建立在对方漏洞的基础上，它比 DOS 和 ARP 攻击的门槛更高。SQL 注入利用的是正常的 HTTP 服务端口，表面上看来和正常的 web 访问没有区别，隐蔽性极强，不易被发现。

（4） 嗅探扫描

TCP 四次挥手

TCP 的连接的拆除需要发送四个包，因此称为四次挥手(four-way handshake)。客户端或服务器均可主动发起挥手动作，在 socket 编程中，任何一方执行 close()操作即可产生挥手操作。

（1）客户端 A 发送一个 FIN，用来关闭客户 A 到服务器 B 的数据传送

（2）服务器 B 收到这个 FIN，它发回一个 ACK，确认序号为收到的序号加 1。和 SYN 一样，一个 FIN 将占用一个序号。

（3）服务器 B 关闭与客户端 A 的连接，发送一个 FIN 给客户端 A。

（4）客户端 A 发回 ACK 报文确认，并将确认序号设置为收到序号加 1。

为什么建立连接协议是三次握手，而关闭连接却是四次握手呢？

这是因为服务端的 LISTEN 状态下的 SOCKET 当收到 SYN 报文的连接请求后，它可以把 ACK 和 SYN(ACK 起应答作用，而 SYN 起同步作用)放在一个报文里来发送。但关闭连接时，当收到对方的 FIN 报文通知时，它仅仅表示对方没有数据发送给你了；但未必你所有的数据都全部发送给对方了，所以你可能未必会马上会关闭 SOCKET,也即你可能还需要发送一些数据给对方之后，再发送 FIN 报文给对方来表示你同意现在可以关闭连接了，所以它这里的 ACK 报文和 FIN 报文多数情况下都是分开发送的。

TIME_WAIT 问题

TIME_WAIT 的状态就是主动断开的一方，发送完最后一次 ACK 之后进入的状态。并且持续

时间还挺长的。能不能发送完 ACK 之后不进入 TIME_WAIT 就直接进入 CLOSE 状态呢？不行的，这个是为了 TCP 协议的可靠性，由于网络原因，ACK 可能会发送失败，那么这个时候，被动一方会主动重新发送一次 FIN，这个时候如果主动方在 TIME_WAIT 状态，则还会再发送一次 ACK，从而保证可靠性。

TCP 协议和 UDP 协议的区别是什么

TCP 协议是有连接的，有连接的意思是开始传输实际数据之前 TCP 的客户端和服务器端必须通过三次握手建立连接，会话结束之后也要结束连接。而 UDP 是无连接的

TCP 协议保证数据按序发送，按序到达，提供超时重传来保证可靠性，但是 UDP 不保证按序到达，甚至不保证到达，只是努力交付，即便是按序发送的序列，也不保证按序送到。

TCP 协议所需资源多，TCP 首部需 20 个字节（不算可选项），UDP 首部字段只需 8 个字节。

TCP 有流量控制和拥塞控制，UDP 没有，网络拥堵不会影响发送端的发送速率

TCP 是一对一的连接，而 UDP 则可以支持一对一，多对多，一对多的通信。

TCP 面向的是字节流的服务，UDP 面向的是报文的服务。

TCP 报文格式

在浏览器中输入 www.baidu.com 后执行的过程

1、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 220.161.27.48，然后通过 TCP 进行封装数据包，输入到网络层。

2、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。

3、客户端的网络层不用关系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，就是通过查找路由表决定通过那个路径到达服务器。

4、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。

DNS 解析: 浏览器首先搜索浏览器自身缓存的 DNS 记录, 如果浏览器缓存中没有找到需要的记录或记录已经过期，则搜索 hosts 文件和操作系统缓存, 如果在 hosts 文件和操作系统缓存中没有找到需要的记录或记录已经过期，则向域名解析服务器发送解析请求,如果域名解析服务器也没有该域名的记录，则开始递归+迭代解析。

负载均衡：一台服务器无法支持大量的用户访问时，将用户分摊到两个或多个服务器上的方法叫负载均衡。

Nginx 负载均衡

CDN 叫内容分发网络，是依靠部署在各地的边缘服务器，使用户就近获取所需内容，降低网络拥塞，提高用户访问响应速度。

TCP/IP 五层模型的协议

应用层：应用层是我们经常接触使用的部分，比如常用的 http 协议、ftp 协议（文件传输协议）、snmp（网络管理协议）、telnet （远程登录协议 ）、smtp（简单邮件传输协议）、dns（域名解析），这次主要是面向用户的交互的。这里的应用层集成了 osi 分层模型中 的应用、会话、表示层三层的功能。

传输层：传输层的作用就是将应用层的数据进行传输转运。比如我们常说的 tcp（可靠的传输控制协议）、udp（用户数据报协议）。传输单位为报文段。

网络层：网络层用来处理网络中流动的数据包，数据包为最小的传递单位，比如我们常用的

ip 协议、icmp 协议、arp 协议（通过分析 ip 地址得出物理 mac 地址）
数据链路层：数据链路层一般用来处理连接硬件的部分，包括控制网卡、硬件相关的设备驱动等。传输单位数据帧。
物理层：物理层一般为负责数据传输的硬件，比如我们了解的双绞线电缆、无线、光纤等。比特流光电等信号发送接收数据。
数据传递：HTTP 报文->TCP 报文->IP 数据报->数据帧->比特流

TCP 滑动窗口流量控制

所谓流量控制，主要是接收方传递信息给发送方，使其不要发送数据太快，是一种端到端的控制。主要的方式就是返回的 ACK 中会包含自己的接收窗口的大小，并且利用大小来控制发送方的数据发送.
持续计时器：处理死锁。
传递效率问题：尽可能一次多发送几个字节。Nagle 算法，先将第一个字节发送出去，后面的字节缓存起来，到适合大小的时候发送（发送窗口一般/报文段最大长度）
如何使用固定窗口的话。窗口过小，那么传输数据量大的时候需要不停的对数据进行确认，会造成很大的延时，窗口过大，会造成不必要的数据带来的阻塞。所以引入滑动窗口机制，窗口的大小并不是固定的，而是根据链接的带宽大小，看链接是否拥塞，接收方能否处理这么多数据。

拥塞控制

网络中的链路容量和交换结点中的缓存和处理机都有着工作的极限，当网络的需求超过它们的工作极限时，就出现了拥塞。拥塞控制就是防止过多的数据注入到网络中，这样可以使网络中的路由器或链路不致过载。
常用的方法就是：
1. 慢开始、拥塞控制
2. 快重传、快恢复

慢开始：为避免一下将大量数据注入网络，造成或增加拥塞，选择发送一个 1 字节试探报文，当收到第一个字节的确认后，发送 2 个字节的报文，若再次收到 2 个字节的确认，则发送 4 个字节，一次递增 2 的指数级，最后会到达一个提前预设的慢开始门限，小于继续使用满开始，大于使用拥塞避免算法，所谓拥塞避免算法就是：每经过一个往返时间 RTT 就把发送方的拥塞窗口+1，即让拥塞窗口缓慢地增大，按照线性规律增长；当出现网络拥塞，比如丢包时，将慢开始门限设为原先的一半，然后将 cwnd 设为 1，执行慢开始算法（较低的起点，指数级增长）；

TCPUDPIP

头

Spring IOC

Spring 是一个轻量级的 **J2EE** 开发框架

理解　**IOC**：**Java** 程序的业务逻辑至少需要 **2** 个或 **2** 个以上的对象来协作完成，每个对象在使用它的合作对象时均要 **New object**（），这样对象间的耦合度就很高，**IOC** 的思想就是通过容器来实现这些依赖对象的创建协调工作，控制这些对象的生命周期和对象间的关系，对象只需要关系业务逻辑本身，对象如何得到协作对象的责任被反转了。

反射允许程序在运行的时候动态生成对象、执行对象方法，改变对象属性，**Spring** 就是通过反射来实现注入的。

对象和对象关系怎么表示？

可以用 **xml** ， **properties** 文件等语义化配置文件表示。

描述对象关系的文件存放在哪里？

可能是 **classpath** ， **filesystem** ，或者是 **URL** 网络资源， **servletContext** 等。

有了配置文件，还需要对配置文件解析。

不同的配置文件对对象的描述不一样，如标准的，自定义声明式的，如何统一？ 在内部需要有一个统一的关于对象的定义，所有外部的描述都必须转化成统一的描述定义。

如何对不同的配置文件进行解析？需要对不同的配置文件语法，采用不同的解析器

其中 **BeanFactory** 作为最顶层的一个接口类，它定义了 **IOC** 容器的基本功能规范，**BeanFactory** 有三个子类：**ListableBeanFactory**、**HierarchicalBeanFactory** 和 **AutowireCapableBeanFactory**。但是从上图中我们可以发现最终的默认实现类是 **DefaultListableBeanFactory**，他实现了所有的接口。那为何要定义这么多层次的接口呢？查阅这些接口的源码和说明发现，每个接口都有他使用的场合，它主要是为了区分在 **Spring** 内部在操作过程中对象的传递和转化过程中，对对象的数据访问所做的限制。例如 **ListableBeanFactory** 接口表示这些 **Bean** 是可列表的，而 **HierarchicalBeanFactory** 表示的是这些 **Bean** 是有继承关系的，也就是每个 **Bean**

有可能有父 Bean。AutowireCapableBeanFactory 接口定义 Bean 的自动装配规则。这四个接口共同定义了 Bean 的集合、Bean 之间的关系、以及 Bean 行为.
最基本的 IOC 容器接口 BeanFactory, 在 BeanFactory 里只对 IOC 容器的基本行为作了定义，根本不关心你的 bean 是如何定义怎样加载的。正如我们只关心工厂里得到什么的产品对象，至于工厂是怎么生产这些对象的，这个基本的接口不关心。
SpringIOC 容器管理了我们定义的各种 Bean 对象及其相互的关系，Bean 对象在 Spring 实现中是以 BeanDefinition 来描述的.
IoC 容器的初始化包括 BeanDefinition 的 Resource 定位、载入和注册这三个基本的过程。
当 Spring IoC 容器完成了 Bean 定义资源的定位、载入和解析注册以后，IoC 容器中已经管理类 Bean 定义的相关数据，但是此时 IoC 容器还没有对所管理的 Bean 进行依赖注入，依赖注入在以下两种情况发生：
(1)用户第一次通过 getBean 方法向 IoC 容索要 Bean 时，IoC 容器触发依赖注入。
(2)当用户在 Bean 定义资源中为<Bean>元素配置了 lazy-init 属性，即让容器在解析注册 Bean 定义时进行预实例化，触发依赖注入。
/*从注入方法上看，主要可以划分为三种类型：构造函数注入、属性注入和接口注入。
Spring 容器在初始化时先读取配置文件，根据配置文件或元数据创建与组织对象存入容器中，程序使用时再从 Ioc 容器中取出需要的对象。*/

使用 XML 配置的方式实现 IOC
使用 Spring 注解配置 IOC
自动装配

AOP 面向切面编程
核心概念：1、横切关注点
对哪些方法进行拦截，拦截后怎么处理，这些关注点称之为横切关注点
2、切面（aspect）
类是对物体特征的抽象，切面就是对横切关注点的抽象
3、连接点（joinpoint）
被拦截到的点，因为 Spring 只支持方法类型的连接点，所以在 Spring 中连接点指的就是被拦截到的方法，实际上连接点还可以是字段或者构造器

4、切入点（pointcut）
对连接点进行拦截的定义
Spring 中 AOP 代理由 Spring 的 IOC 容器负责生成、管理，其依赖关系也由 IOC 容器负责管理。因此，AOP 代理可以直接使用容器中的其它 bean 实例作为目标，这种关系可由 IOC 容器的依赖注入提供。Spring 创建代理的规则为：
1、默认使用 Java 动态代理来创建 AOP 代理，这样就可以为任何接口实例创建代理了
2、当需要代理的类不是代理接口的时候，Spring 会切换为使用 CGLIB 代理，也可强制使用 CGLIB

Spring MVC
处理请求流程：
1、 首先用户发送请求————>前端控制器，前端控制器根据请求信息（如 URL）来决定选择哪一个页面控制器进行处理并把请求委托给它，即以前的控制器的控制逻辑部分
2、 页面控制器接收到请求后，进行功能处理，首先需要收集和绑定请求参数到一个对象，

这个对象在 Spring Web MVC 中叫命令对象，并进行验证，然后将命令对象委托给业务对象进行处理；处理完毕后返回一个 ModelAndView（模型数据和逻辑视图名）3、 前端控制器收回控制权，然后根据返回的逻辑视图名，选择相应的视图进行渲染，并把模型数据传入以便视图渲染；图 2-1 中的步骤 6、7；

4、 前端控制器再次收回控制权，将响应返回给用户

核心架构的具体流程步骤如下：

1、 首先用户发送请求——>DispatcherServlet，前端控制器收到请求后自己不进行处理，而是委托给其他的解析器进行处理，作为统一访问点，进行全局的流程控制；

2、 DispatcherServlet——>HandlerMapping， HandlerMapping 将会把请求映射为 HandlerExecutionChain 对象（包含一个 Handler 处理器（页面控制器）对象、多个 HandlerInterceptor 拦截器）对象，通过这种策略模式，很容易添加新的映射策略；

3、 DispatcherServlet——>HandlerAdapter，HandlerAdapter 将会把处理器包装为适配器，从而支持多种类型的处理器，即适配器设计模式的应用，从而很容易支持很多类型的处理器；

4、 HandlerAdapter——>处理器功能处理方法的调用，HandlerAdapter 将会根据适配的结果调用真正的处理器的功能处理方法，完成功能处理；并返回一个 ModelAndView 对象（包含模型数据、逻辑视图名）；

5、 ModelAndView 的逻辑视图名——> ViewResolver， ViewResolver 将把逻辑视图名解析为具体的 View，通过这种策略模式，很容易更换其他视图技术；

6、 View——>渲染，View 会根据传进来的 Model 模型数据进行渲染，此处的 Model 实际是一个 Map 数据结构，因此很容易支持其他视图技术；

7、返回控制权给 DispatcherServlet，由 DispatcherServlet 返回响应给用户，到此一个流程结束。