1：简述 OSI 七层协议。

2：什么是C/S和B/S架构？

3：简述 三次握手、四次挥手的流程。

4：什么是arp协议？

5：TCP和UDP的区别？

6：什么是局域网和广域网？

7：为何基于tcp协议的通信比基于udp协议的通信更可靠？

8：什么是socket？简述基于tcp协议的套接字通信流程。

9：什么是粘包？ socket 中造成粘包的原因是什么？ 哪些情况会发生粘包现象？

11：什么是防火墙以及作用？

13：简述 进程、线程、协程的区别 以及应用场景？

15：Python中如何使用线程池和进程池？

17：进程之间如何进行通信？

18：什么是并发和并行？

19：进程锁和线程锁的作用？

21：路由器和交换机的区别？

22：什么是域名解析？

23：如何修改本地hosts文件？

25：什么是cdn？

33：简述 gevent模块的作用和应用场景。

1.答：

应用层

与其它计算机进行通讯的一个应用，它是对应应用程序的通信服务的。例如，一个没有通信功能的字处理程序就不能执行通信的代码，从事字处理工作的程序员也不关心OSI的第7层。但是，如果添加了一个传输文件的选项，那么字处理器的程序员就需要实现OSI的第7层。示例：TELNET，HTTP，FTP，NFS，SMTP等。

表示层

这一层的主要功能是定义数据格式及加密。例如，FTP允许你选择以二进制或ASCII格式传输。如果选择二进制，那么发送方和接收方不改变文件的内容。如果选择ASCII格式，发送方将把文本从发送方的字符集转换成标准的ASCII后发送数据。在接收方将标准的ASCII转换成接收方计算机的字符集。示例：加密，ASCII等。

会话层

它定义了如何开始、控制和结束一个会话，包括对多个双向消息的控制和管理，以便在只完成连续消息的一部分时可以通知应用，从而使表示层看到的数据是连续的，在某些情况下，如果表示层收到了所有的数据，则用数据代表表示层。示例：RPC，SQL等。

传输层

这层的功能包括是否选择差错恢复协议还是无差错恢复协议，及在同一主机上对不同应用的数据流的输入进行复用，还包括对收到的顺序不对的数据包的重新排序功能。示例：TCP，UDP，SPX。

网络层

这层对端到端的包传输进行定义，它定义了能够标识所有结点的逻辑地址，还定义了路由实现的方式和学习的方式。为了适应最大传输单元长度小于包长度的传输介质，网络层还定义了如何将一个包分解成更小的包的分段方法。示例：IP，IPX等。

数据链路层

它定义了在单个链路上如何传输数据。这些协议与被讨论的各种介质有关。示例：ATM，FDDI等。

物理层

OSI的物理层规范是有关传输介质的特这些规范通常也参考了其他组织制定的标准。连接头、帧、帧的使用、电流、编码及光调制等都属于各种物理层规范中的内容。物理层常用多个规范完成对所有细节的定义。示例：Rj45，802.3等。

2.答：

C/S架构软件（即客户机/服务器模式）分为客户机和服务器两层：第一层是在客户机系统上结合了表示与业务逻辑，第二层是通过网络结合了数据库服务器。简单的说就是第一层是用户表示层，第二层是数据库层。客户端和服务器直接相连，这两个组成部分都承担着重要的角色。

B/S架构第一层是浏览器（即客户端）只有简单的输入输出功能，处理极少部分的事务逻辑。由于客户不需要安装客户端，只要有浏览器就能上网浏览，所以它面向的是大范围的用户，所以界面设计得比较简单，通用。

第二层是WEB服务器，扮演着信息传送的角色。当用户想要访问数据库时，就会首先向WEB服务器发送请求，WEB服务器统一请求后会向数据库服务器发送访问数据库的请求，这个请求是以SQL语句实现的。

第三层是数据库服务器，它存放着大量的数据。当数据库服务器收到了WEB服务器的请求后，会对SQL语句进行处理，并将返回的结果发送给WEB服务器，接下来，WEB服务器将收到的数据结果转换为HTML文本形式发送给浏览器。

3.答：

所谓三次握手（Three-Way Handshake）即建立TCP连接，就是指建立一个TCP连接时，需要客户端和服务端总共发送3个包以确认连接的建立。在socket编程中，这一过程由客户端执行connect来触发，整个流程如下图所示：

图2 TCP三次握手

（1）第一次握手：Client将标志位SYN置为1，随机产生一个值seq=J，并将该数据包发送给Server，Client进入SYN\_SENT状态，等待Server确认。

（2）第二次握手：Server收到数据包后由标志位SYN=1知道Client请求建立连接，Server将标志位SYN和ACK都置为1，ack=J+1，随机产生一个值seq=K，并将该数据包发送给Client以确认连接请求，Server进入SYN\_RCVD状态。

（3）第三次握手：Client收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=K+1，并将该数据包发送给Server，Server检查ack是否为K+1，ACK是否为1，如果正确则连接建立成功，Client和Server进入ESTABLISHED状态，完成三次握手，随后Client与Server之间可以开始传输数据了。

所谓四次挥手（Four-Way Wavehand）即终止TCP连接，就是指断开一个TCP连接时，需要客户端和服务端总共发送4个包以确认连接的断开。在socket编程中，这一过程由客户端或服务端任一方执行close来触发，整个流程如下图所示：

图3 TCP四次挥手

由于TCP连接时全双工的，因此，每个方向都必须要单独进行关闭，这一原则是当一方完成数据发送任务后，发送一个FIN来终止这一方向的连接，收到一个FIN只是意味着这一方向上没有数据流动了，即不会再收到数据了，但是在这个TCP连接上仍然能够发送数据，直到这一方向也发送了FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭，上图描述的即是如此。

（1）第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN\_WAIT\_1状态。

（2）第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），Server进入CLOSE\_WAIT状态。

（3）第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST\_ACK状态。

（4）第四次挥手：Client收到FIN后，Client进入TIME\_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。

4.答：ARP (Address Resolution Protocol) 是个地址解析协议。最直白的说法是：在IP以太网中，当一个上层协议要发包时，有了该节点的IP地址，ARP就能提供该节点的MAC地址。

5.答：

1、TCP面向连接（如打电话要先拨号建立连接）;UDP是无连接的，即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

Tcp通过校验和，重传控制，序号标识，滑动窗口、确认应答实现可靠传输。如丢包时的重发控制，还可以对次序乱掉的分包进行顺序控制。

3、UDP具有较好的实时性，工作效率比TCP高，适用于对高速传输和实时性有较高的通信或广播通信。

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对系统资源要求较多，UDP对系统资源要求较少。

6.答：什么是广域网（WAN、公网、外网），什么是局域网（LAN、私网、内网）？

广域网（WAN），就是我们通常所说的Internet，它是一个遍及全世界的网络。 局域网（LAN），相对于广域网（WAN）而言，主要是指在小范围内的计算机互联网络。这个“小范围”可以是一个家庭，一所学校，一家公司，或者是一个政府部门。 BT中常常提到的公网、外网，即广域网（WAN）；BT中常常提到私网、内网，即局域网（LAN）。

广域网上的每一台电脑（或其他网络设备）都有一个或多个广域网IP地址（或者说公网、外网IP地址），广域网IP地址一般要到ISP处交费之后才能申请到，广域网IP地址不能重复；局域网（LAN）上的每一台电脑（或其他网络设备）都有一个或多个局域网IP地址（或者说私网、内网IP地址），局域网IP地址是局域网内部分配的，不同局域网的IP地址可以重复，不会相互影响。

7.答：

TCP的可靠保证，是它的三次握手双向机制，这一机制保证校验了数据，保证了他的可靠性。而UDP就没有了，udp信息发出后,不验证是否到达对方,所以不可靠。

8.答：

Socket 又称”套接字”,是系统提供的用于网络通信的方法.它的是指并不是一种协议,没有规定计算机应当怎么传递信息,只是给程序员提供了一个发送消息的接口,程序员能使用这个接口同的方法,发送与接收消息.Socket 描述了一个 IP,端口对.它简化了程序员的操作,知道对方的 IP以及 port 就可以给对方发送消息,再由服务器处理发送的这些消息,Socket 包含了通信的双发,即客户端与服务端.

TCP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();

3、绑定IP地址、端口等信息到socket上，用函数bind();

4、开启监听，用函数listen()；

　 5、接收客户端上来的连接，用函数accept()；

　　6、收发数据，用函数send()和recv()，或者read()和write();

　　7、关闭网络连接；

8、关闭监听；

SOCK\_STREAM这种的特点是面向连接的，即每次收发数据之前必须通过connect建立连接，而SOCK\_DGRAM这种是User Datagram Protocol协议的网络通讯，它是无连接的，不可靠的。

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；

9.答：

粘包：多个数据包被连续存储于连续的缓存中，在对数据包进行读取时由于无法确定发生方的发送边界，而采用某一估测值大小来进行数据读出，若双方的size不一致时就会使指发送方发送的若干包数据到接收方接收时粘成一包，从接收缓冲区看，后一包数据的头紧接着前一包数据的尾。

出现粘包现象的原因是多方面的，它既可能由发送方造成，也可能由接收方造成。发送方引起的粘包是由TCP协议本身造成的，TCP为提高传输效率，发送方往往要收集到足够多的数据后才发送一包数据。若连续几次发送的数据都很少，通常TCP会根据优化算法把这些数据合成一包后一次发送出去，这样接收方就收到了粘包数据。接收方引起的粘包是由于接收方用户进程不及时接收数据，从而导致粘包现象。这是因为接收方先把收到的数据放在系统接收缓冲区，用户进程从该缓冲区取数据，若下一包数据到达时前一包数据尚未被用户进程取走，则下一包数据放到系统接收缓冲区时就接到前一包数据之后，而用户进程根据预先设定的缓冲区大小从系统接收缓冲区取数据，这样就一次取到了多包数据。分包是指在出现粘包的时候我们的接收方要进行分包处理。（在长连接中都会出现） 数据包的边界发生错位，导致读出错误的数据分包，进而曲解原始数据含义。

当时短连接的情况下，不用考虑粘包的情况

2.如果发送数据无结构，如文件传输，这样发送方只管发送，接收方只管接收存储就ok，也不用考虑粘包

3.如果双方建立连接，需要在连接后一段时间内发送不同结构数据

处理方式：

接收方创建一预处理线程，对接收到的数据包进行预处理，将粘连的包分开

注：粘包情况有两种，一种是粘在一起的包都是完整的数据包，另一种情况是粘在一起的包有不完整的包

11.答：

什么是防火墙

防火墙是指设置在不同网络（如可信任的企业内部网和不可信的公共网）或网络安全域之间的一系列部件的组合。 它可通过监测、限制、更改跨越防火墙的数据流，尽可能地对外部屏蔽网络内部的信息、结构和运行状况， 以此来实现网络的安全保护。

在逻辑上，防火墙是一个分离器，一个限制器，也是一个分析器，有效地监控了内部网和Internet之间的任何活动， 保证了内部网络的安全。

2.使用Firewall的益处

保护脆弱的服务

通过过滤不安全的服务，Firewall可以极大地提高网络安全和减少子网中主机的风险。例如， Firewall可以禁止NIS、NFS服务通过，Firewall同时可以拒绝源路由和ICMP重定向封包。

控制对系统的访问

Firewall可以提供对系统的访问控制。如允许从外部访问某些主机，同时禁止访问另外的主机。例如， Firewall允许外部访问特定的Mail Server和Web Server。

集中的安全管理

Firewall对企业内部网实现集中的安全管理，在Firewall定义的安全规则可以运行于整个内部网络系统， 而无须在内部网每台机器上分别设立安全策略。Firewall可以定义不同的认证方法， 而不需要在每台机器上分别安装特定的认证软件。外部用户也只需要经过一次认证即可访问内部网。

增强的保密性

使用Firewall可以阻止攻击者获取攻击网络系统的有用信息，如Figer和DNS。

记录和统计网络利用数据以及非法使用数据

Firewall可以记录和统计通过Firewall的网络通讯，提供关于网络使用的统计数据，并且，Firewall可以提供统计数据， 来判断可能的攻击和探测。

策略执行

Firewall提供了制定和执行网络安全策略的手段。未设置Firewall时，网络安全取决于每台主机的用户。

13．答：

1.线程和进程：

线程是属于进程的，线程运行在进程空间内，同一进程所产生的线程共享同一内存空间，当进程退出时该进程所产生的线程都会被强制退出并清除。线程可与属于同一进程的其它线程共享进程所拥有的全部资源，但是其本身基本上不拥有系统资源，只拥有一点在运行中必不可少的信息(如程序计数器、一组寄存器和栈)。

2.线程、进程与协程：

线程和进程的操作是由程序触发系统接口，最后的执行者是系统；协程的操作则是程序员

协程存在的意义：对于多线程应用，CPU通过切片的方式来切换线程间的执行，线程切换时需要耗时（保持状态，下次继续）。协程，则只使用一个线程，在一个线程中规定某个代码块执行顺序。

协程的适用场景： 当程序中存在大量不需要CPU的操作时（IO），适用于协程；

17.答：进程间通信（IPC，Interprocess communication），方法包括管道（PIPE）、消息排队、旗语、共用内存以及套接字（Socket）。

18.答：并发是指一个处理器同时处理多个任务。

并行是指多个处理器或者是多核的处理器同时处理多个不同的任务。

并发是逻辑上的同时发生（simultaneous），而并行是物理上的同时发生。

19. 答：　线程锁:大家都不陌生，主要用来给方法、代码块加锁。当某个方法或者代码块使用锁时，那么在同一时刻至多仅有有一个线程在执行该段代码。当有多个线程访问同一对象的加锁方法/代码块时，同一时间只有一个线程在执行，其余线程必须要等待当前线程执行完之后才能执行该代码段。但是，其余线程是可以访问该对象中的非加锁代码块的。

　　进程锁:也是为了控制同一操作系统中多个进程访问一个共享资源，只是因为程序的独立性，各个进程是无法控制其他进程对资源的访问的，但是可以使用本地系统的信号量控制（操作系统基本知识）。

21. 答：

（1）工作层次不同

最初的的交换机是工作在OSI/RM开放体系结构的数据链路层，也就是第二层，而路由器一开始就设计工作在OSI模型的网络层。由于交换机工作在OSI的第二层（数据链路层），所以它的工作原理比较简单，而路由器工作在OSI的第三层（网络层），可以得到更多的协议信息，路由器可以做出更加智能的转发决策。

（2）数据转发所依据的对象不同

交换机是利用物理地址或者说MAC地址来确定转发数据的目的地址。而路由器则是利用不同网络的ID号（即IP地址）来确定数据转发的地址。IP地址是在软件中实现的，描述的是设备所在的网络，有时这些第三层的地址也称为协议地址或者网络地址。MAC地址通常是硬件自带的，由网卡生产商来分配的，而且已经固化到了网卡中去，一般来说是不可更改的。而IP地址则通常由网络管理员或系统自动分配。

（3）传统的交换机只能分割冲突域，不能分割广播域；而路由器可以分割广播域由交换机连接的网段仍属于同一个广播域，广播数据包会在交换机连接的所有网段上传播，在某些情况下会导致通信拥挤和安全漏洞。连接到路由器上的网段会被分配成不同的广播域，广播数据不会穿过路由器。虽然第三层以上交换机具有VLAN功能，也可以分割广播域，但是各子广播域之间是不能通信交流的，它们之间的交流仍然需要路由器。

（4）路由器提供了防火墙的服务

路由器仅仅转发特定地址的数据包，不传送不支持路由协议的数据包传送和未知目标网络数据包的传送，从而可以防止广播风暴。

22.答：

域名解析是把域名指向网站空间IP，让人们通过注册的域名可以方便地访问到网站的一种服务。IP地址是网络上标识站点的数字地址，为了方便记忆，采用域名来代替IP地址标识站点地址。域名解析就是域名到IP地址的转换过程。域名的解析工作由DNS服务器完成。

23. 答：

hosts 文件所在位置

c:/windows/system32/drivers/etc/hosts

左下角 搜索框 搜索 cmd

弹出命令框

输入

ipconfig /displaydns

显示所有 dns内容

ipconfig /flushdns

刷新所有 dns内容

25. 答：

CDN的全称是Content Delivery Network，即内容分发网络。其基本思路是尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输的更快、更稳定。通过在网络各处放置节点服务器所构成的在现有的互联网基础之上的一层智能虚拟网络，CDN系统能够实时地根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务节点上。其目的是使用户可就近取得所需内容，解决 Internet网络拥挤的状况，提高用户访问网站的响应速度。

33. 答：

gevent是第三方库，通过greenlet实现协程，其基本思想是：当一个greenlet遇到IO操作时，比如访问网络，就自动切换到其他的greenlet，等到IO操作完成，再在适当的时候切换回来继续执行。由于IO操作非常耗时，经常使程序处于等待状态，有了gevent为我们自动切换协程，就保证总有greenlet在运行，而不是等待IO。

由于切换是在IO操作时自动完成，所以gevent需要修改Python自带的一些标准库，这一过程在启动时通过monkey patch完成

使用gevent，可以获得极高的并发性能，但gevent只能在Unix/Linux下运行，在Windows下不保证正常安装和运行。

由于gevent是基于IO切换的协程，所以最神奇的是，我们编写的Web App代码，不需要引入gevent的包，也不需要改任何代码，仅仅在部署的时候，用一个支持gevent的WSGI服务器，立刻就获得了数倍的性能提升。