网络

一、 物理层

二、 数据链路层

1. 路由器和交换器的区别

交换机工作于数据链路层，能识别 MAC 地址，根据 MAC 地址转发链路层数据帧。具有自学机制来维护 IP 地址与 MAC 地址的映射。

路由器位于网络层，能识别 IP 地址并根据 IP 地址转发分组。维护着路由表，根据路由表选择最佳路线。

三、 网络层

1. 路由器的功能?

路由选择与分组转发

2. ip 报文如何从下向上交付

3. ip 地址有什么用，ip 地址和 mac 地址，为什么需要 IP 地址

1) IP 地址是在网络上分配给每台计算机或网络设备的 32 位数字标识。在 Internet 上，每台计算机或网络设备的 IP 地址是全世界唯一的。IP 地址的格式是 xxx.xxx.xxx.xxx，其中 xxx 是 0 到 255 之间的任意整数。例如，每步站主机的 IP 地址是 219.134.132.131。
2) MAC 地址是数据链路层的地址，如果 mac 地址不可直达 ,直接丢弃，在 LAN 里面，一个网卡的 MAC 地址是唯一的。MAC 地址在 arp 协议里常常用到，mac 地址到 ip 地址的相互转化。Mac 地址是 48 位的地址。
3) IP 地址是网络层的地址，如果 ip 地址不可达，接着转发，在 WAN 里面，ip 地址不唯一，计算机的 ip 地址可以变动

4. ARP 协议的作用

ARP（地址解析）协议是一种解析协议，本来主机是完全不知道这个 IP 对应的是哪个主机的哪个接口，当主机要发送一个 IP 包的时候，会首先查一下自己的 ARP

高速缓存表（最近数据传递更新的 IP-MAC 地址对应表），如果查询的 IP - MAC 值对不存在，那么主机就向网络广播一个 ARP 请求包，这个包里面就有待查询的 IP 地址，而直接收到这份广播的包的所有主机都会查询自己的 IP 地址，如果收到广播包的某一个主机发现自己符合条件，那么就回应一个 ARP 应答包（将自己对应的 IP-MAC 对应地址发回主机），源主机拿到 ARP 应答包后会更新自己的 ARP 缓存表。源主机根据新的 ARP 缓存表准备好数据链路层的的数据包发送工作。

5. NAT 的原理，外网与内网或内网之间的通信中如何区分不同 IP 的数组包

1. **公有 IP 地址**：也叫全局地址，是指合法的 IP 地址，它是由 NIC（网络信息中心）或者 ISP(网络服务提供商)分配的地址，对外代表一个或多个内部局部地址，是全球统一的可寻 址的地址。
2. **私有 IP 地址**：也叫内部地址，属于非注册地址，专门为组织机构内部使用。因特网分配编号委员会（IANA）保留了 3 块 IP 地址做为私有 IP 地址；
3. NAT 英文全称是“Network Address Translation”，中文意思是“网络地址转换”，它是一个 IETF(Internet Engineering Task Force, Internet 工程任务组)标准，允许一个整体机构以一个公用 IP（Internet Protocol）地址出现在 Internet 上。顾名思义，它是一种把内部私有网络地址（IP 地址）翻译成合法网络 IP 地址的技术，如下图所示。因此我们可以认为，NAT 在一定程度上，能够有效的解决公网地址不足的问题。
4. NAT 就是在局域网内部网络中使用内部地址，而当内部节点要与外部网络进行通讯时，就在网关（可以理解为出口，打个比方就像院子的门一样）处，将内部地址替换成公用地址，从而在外部公网（internet）上正常使用，NAT 可以使多台计算机共享 Internet 连接，这一功能很好地解决了公共 IP 地址紧缺的问题。通过这种方法，可以只申请一个合法 IP 地址，就把整个局域网中的计算机接入 Internet 中。这时，NAT 屏蔽了内部网络，所有内部网计算机对于公共网络来说是不可见的，而内部网计算机用户通常不会意识到 NAT 的存在。

6. RIP 路由协议

1. 网络中的每一个路由器都要维护从它自己到其他每一个目标网络的距离记录；
2. 距离也称为跳数，规定从一路由器到直接连接的网络跳数为 1，而每经过一个路由器，则距离加 1；
3. RIP 认为好的路由就是它通过的路由器数量最少；
4. RIP 允许一条路径上最多有 15 个路由器，因为规定最大跳数为 16；

5. RIP 默认每 30 秒广播一次 RIP 路由更新信息。

每一个路由表项目包括三个内容：目的网络、距离、下一跳路由器

1、对地址为 X 的路由器发过来的路由表，先修改此路由表中的所有项目：把"下一跳"字段中的地址改为 X，并把所有"距离"字段都加 1。

2、对修改后的路由表中的每一个项目，进行以下步骤：

2.1、将 X 的路由表(修改过的)，与 S 的路由表的目的网络进行对比。

若在 X 中出现，在 S 中没出现，则将 X 路由表中的这一条项目添加到 S 的路由表中。

2.2、对于目的网络在 S 和 X 路由表中都有的项目进行下面步骤

2.2.1、在 S 的路由表中，若下一跳地址是 x

则直接用 X 路由表中这条项目替换 S 路由表中的项目。

2.2.2、在 S 的路由表中，若下一跳地址不是 x

若 X 路由表项目中的距离 d 小于 S 路由表中的距离，则进行更新。

3、若 3 分钟还没有收到相邻路由器的更新表，则把此相邻路由器记为不可到达路由器，即把距离设置为 16。

7. 为什么使用 IP 地址通信及端口号的作用

1) 由于全世界存在着各式各样的网络，它们使用不同的硬件地址。要使这些异构网络能够互相通信就必须进行非常复杂的硬件地址转换工作，因此几乎是不可能的事。

2) 连接到因特网的主机都拥有统一的 IP 地址，它们之间的通信就像连接在同一个网络上那样简单方便，因为调用 ARP 来寻找某个路由器或主机的硬件地址都是由计算机软件自动进行的，对用户来说是看不见这种调用过程的。

3) 端口号就是同一操作系统内为区别不同套接字而设置的，因此无法将 1 个端口号分配给不同的套接字，端口号由 16 位组成，可分配的端口号范围是 0-65535。但 0-1023 是知名端口，一般分配给特定的应用程序。TCP 和 UDP 不会共用端口号，所以他们可以使用相同端口号。

8. 子网掩码有什么用?

子网掩码是一种用来指明一个 IP 地址所标示的主机处于哪个子网中。子网掩码不能单独存在，它必须结合 IP 地址一起使用。子网掩码只有一个作用，就是将某个 IP 地址划分成网络地址和主机地址两部分。

9. 子网划分的方法

1) 传统子网划分，ip 地址结构=网络号+主机号
2) 子网掩码
3) CIDR，减少了传统分法的 ip 浪费。(适当分配多个合适的 IP 地址，使得这些地址能够进行聚合，减少这些地址在路由表中的表项数)

10. IP 报文经过路由器的转发过程及变化

1) 防火墙收到数据包后，解封以太网帧头部，提取目的 MAC 地址，查看目的 MAC 地址是不是自己本身的 MAC 地址。
2) 如果不是自己的 MAC 地址则丢弃。
3) 如果是自己的 MAC 地址，上传到上层解析，解析 IP 层。
4) 假设是自己的 MAC 地址，解析 IP 层，提取目的 IP 地址，判断目的 IP 地址是不是指向本机，
5) 如果是指向本机，则上传到上层，有上层解析
6) 如果不是指向自己而是转发，则去查路由表，匹配出接口。
7) 假设数据包是转发，则根据路由的最长匹配原则，匹配路由表，找到出接口。
8) 如果匹配的路由是直连路由(与路由器直接连接，中间没有其他设备）则使用目的地址查 ARP 表。
9) 如果匹配的路由不是直连路由则使用下一跳的 IP 地址查 ARP 表。
10) 假设数据包不是直连路由，去 ARP 表中查下一跳的 IP 地址对应的 MAC 地址。
11) 如果查到了下一跳的 IP 地址对应的 MAC 地址，则把 MAC 地址封装到帧的目的 MAC 中，然后封装物理层发送出去
12) 如果没查到就发 ARP 请求，查找 IP 地址对应的 MAC 地址
13) 假设没有查到下一跳的 IP 地址对应的 MAC 地址，发送 ARP 请求报文获取 IP 对应的 MAC 地址。
14) 发送 ARP 请求后，如果没收到 ARP 响应则丢弃数据包。
15) 发送 ARP 请求后，如果收到 ARP 响应，提取 ARP 响应中的源 MAC 地址存放到 ARP 表中形成映射关系。
16) 假设收到的 ARP 响应，把下一跳的 MAC 地址放到目的 MAC 地址中，封装，发送。

11. TCP 协议有几大计时器?

1) 重传计时器

目的：为了控制丢失的报文段或者丢弃的报文段。这段时间为对报文段的等待确认时间。

创建时间：在 TCP 发送报文段时，会创建对次特定报文段的重传计时器。

可能发生的两种情况：在截止时间（通常为 60 秒）到之前，已经收到了对此特定报文段的确认，则撤销计时器；在截止时间到了，但为收到对此特定报文段的确认，则重传报文段，并且将计时器复位。

重传时间：2*RTT（Round Trip Time，为往返时间

2) 持续计时器

目的：主要解决零窗口大小通知可能导致的死锁问题

死锁问题的产生：当接收端的窗口大小为 0 时，接收端向发送端发送一个零窗口报文段，发送端即停止向对端发送数据。此后，如果接收端缓存区有空间则会重新给发送端发送一个窗口大小，即窗口更新。但接收端发送的这个确认报文段有可能会丢失，而此时接收端不知道已经丢失并认为自己已经发送成功，则一直处于等待数据的状态；而发送端由于没有收到该确认报文段，就会一直等待对方发来新的窗口大小，这样一来，双方都处在等待对方的状态，这样就形成了一种死锁问题。如果没有应对措施，这种局面是不会被打破的。为了解决这种问题，TCP 为每一个连接设置了坚持计时器。

工作原理：当发送端 TCP 收到接收端发来的零窗口通知时，就会启动坚持计时器。当计时器的期限到达时，发送端就会主动发送一个特殊的报文段告诉对方确认已经丢失，必须重新发送。【这个特殊的报文段就称为探测报文段，探测报文段只有 1 个字节的大小，里边只有一个序号，该序号不需要被确认，甚至在计算其他部分数据的确认时该序号会被忽略。】

截止期的设置：设置为重传时间的值。但如果没有收到接收端的响应，则会发送另一个探测报文段，并将计时器的值加倍并复位，直到大于门限值（一般为 60 秒）。在此之后，发送端会每隔 60 秒发送一个探测报文段，直到窗口重新打开。

3) 保活计时器

目的：主要是为了防止两个 TCP 连接出现长时间的空闲。当客户端与服务器端建立 TCP 连接后，很长时间内客户端都没有向服务器端发送数据，此时很有可能是客户端出现故障，而服务器端会一直处于等待状态。保活计时器就是解决这种问题而生的。

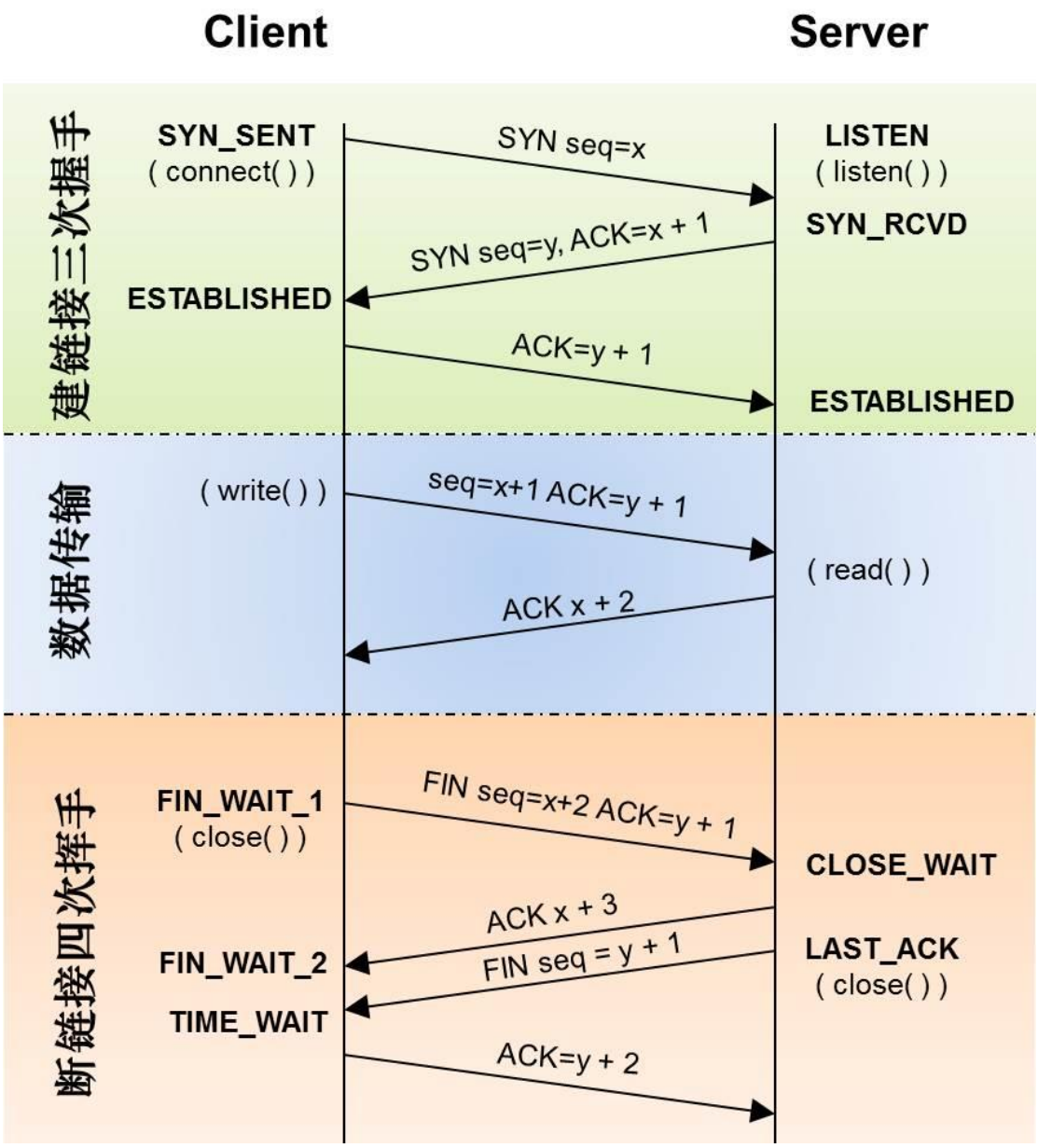
工作原理：每当服务器端收到客户端的数据时，都将保活计时器重新设置（通常设置为 2 小时）。过了 2 小时后，服务器端如果没有收到客户端的数据，会发送探测报文段给客户端，并且每隔 75 秒发送一个，当连续发送 10 次以后，仍没有收到

对端的来信，则服务器端认为客户端出现故障，并会终止连接。

4) 时间等待计时器

时间等待计时器是在连接终止期间使用的。当 TCP 关闭一个连接时，它并不认为这个连接马上就真正地关闭了。在时间等待期间中，连接还处于一种中间过渡状态。这就可以使重复的 FIN 报文段（如果有的话）可以到达目的站因而可将其丢弃。这个计时器的值通常设置为一个报文段的寿命期待值的两倍。

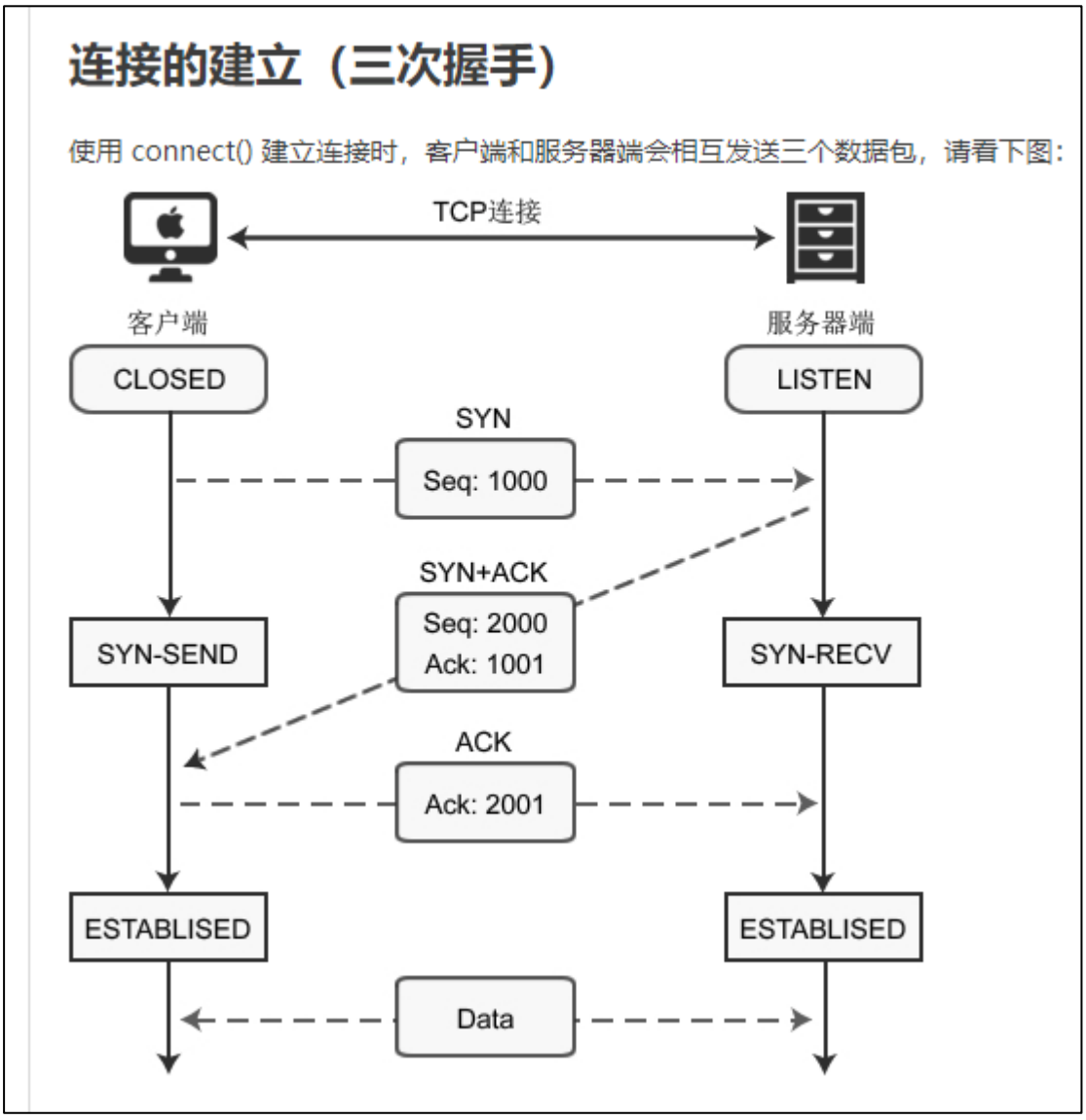
12. 详细说一下 TCP 协议，三次握手传输的内容？13 种状态



1) 第一次握手：建立连接。客户端发送连接请求报文段，将 SYN 位置为 1，Sequence Number 为 x；然后，客户端进入 SYN_SEND 状态，等待服务器的确认；

2) 第二次握手：服务器收到 SYN 报文段。服务器收到客户端的 SYN 报文段，需要对这个 SYN 报文段进行确认，设置 Acknowledgment Number 为 x+1(Sequence Number+1)；同时，自己自己还要发送 SYN 请求信息，将 SYN 位置为 1，Sequence Number 为 y；服务器端将上述所有信息放到一个报文段（即 SYN+ACK 报文段）中，一并发送给客户端，此时服务器进入 SYN_RECV 状态；

3) 第三次握手：客户端收到服务器的 SYN+ACK 报文段。然后将 Acknowledgment Number 设置为 y+1，向服务器发送 ACK 报文段，这个报文段发送完毕以后，客户端和服务器端都进入 ESTABLISHED 状态，完成 TCP 三次握手。



客户端调用 socket() 函数创建套接字后，因为没有建立连接，所以套接字处于 CLOSED 状态；服务器端调用 listen() 函数后，套接字进入 LISTEN 状态，开始监听客户端请求。

这个时候，客户端开始发起请求：

1）当客户端调用 connect() 函数后，TCP 协议会组建一个数据包，并设置 SYN 标志位，表示该数据包是用来建立同步连接的。同时生成一个随机数字 1000，填充“序号（Seq）”字段，表示该数据包的序号。完成这些工作，开始向服务器端发送数据包，客户端就进入了 SYN-SEND 状态。

2）服务器端收到数据包，检测到已经设置了 SYN 标志位，就知道这是客户端发来的建立连接的“请求包”。服务器端也会组建一个数据包，并设置 SYN 和 ACK 标志位，SYN 表示该数据包用来建立连接，ACK 用来确认收到了刚才客户端发送的数据包。服务器生成一个随机数 2000，填充“序号（Seq）”字段。2000 和客户端数据包没有关系。服务器将客户端数据包序号（1000）加 1，得到 1001，并用这个数字填充“确认号（Ack）”字段。服务器将数据包发出，进入 SYN-RECV 状态。

3）客户端收到数据包，检测到已经设置了 SYN 和 ACK 标志位，就知道这是服务器发来的“确认包”。客户端会检测“确认号（Ack）”字段，看它的值是否为 1000+1，如果是就说明连接建立成功。

接下来，客户端会继续组建数据包，并设置 ACK 标志位，表示客户端正确接收了服务器发来的“确认包”。同时，将刚才服务器发来的数据包序号（2000）加 1，得到 2001，并用这个数字来填充“确认号（Ack）”字段。

客户端将数据包发出，进入 ESTABLISED 状态，表示连接已经成功建立。

4）服务器端收到数据包，检测到已经设置了 ACK 标志位，就知道这是客户端发来的“确认包”。服务器会检测“确认号（Ack）”字段，看它的值是否为 2000+1，如果是就说明连接建立成功，服务器进入 ESTABLISED 状态。

至此，客户端和服务器都进入了 ESTABLISED 状态，连接建立成功，接下来就可以收发数据了。

注：三次握手的关键是要确认对方收到了自己的数据包，这个目标就是通过 “确认号（Ack）”字段实现的。计算机会记录下自己发送的数据包序号 Seq，待收到对方的数据包后，检测“确认号（Ack）”字段，看 Ack = Seq + 1 是否成立，如果成立说明对方正确收到了自己的数据包

那四次挥手呢?

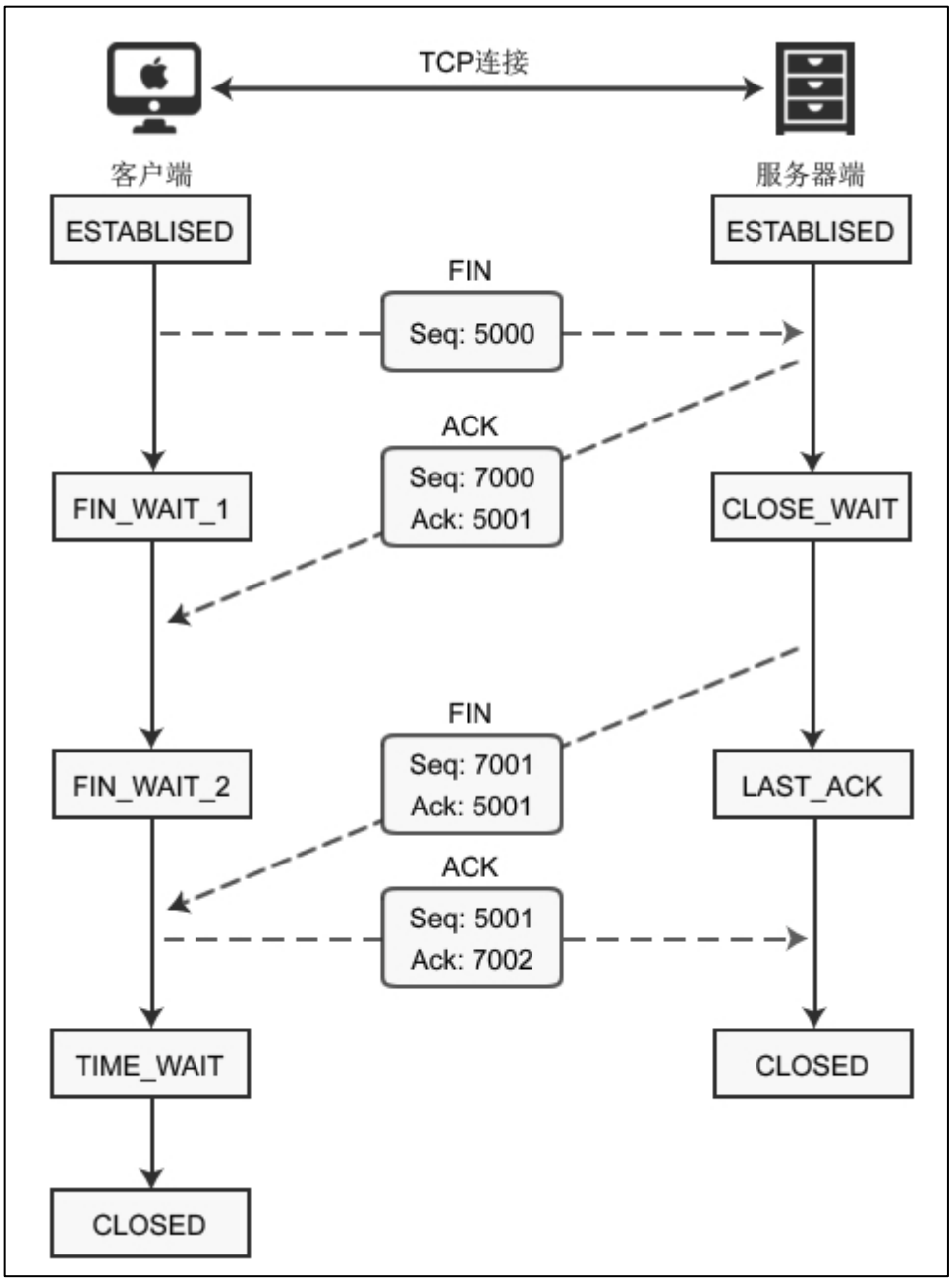
当客户端和服务器通过三次握手建立了 TCP 连接以后，当数据传送完毕，肯定是要断开 TCP 连接的啊。那对于 TCP 的断开连接，这里就有了神秘的“四次分手”。

1) 第一次挥手：主机 1（可以使客户端，也可以是服务器端），设置 Sequence Number 和 Acknowledgment Number，向主机 2 发送一个 FIN 报文段；此时，主机 1 进入 FIN_WAIT_1 状态；这表示主机 1 没有数据要发送给主机 2 了；

2) 第二次挥手：主机 2 收到了主机 1 发送的 FIN 报文段，向主机 1 回一个 ACK 报文段，Acknowledgment Number 为 Sequence Number 加 1；主机 1 进入 FIN_WAIT_2 状态；主机 2 告诉主机 1，我“同意”你的关闭请求；

3) 第三次挥手：主机 2 向主机 1 发送 FIN 报文段，请求关闭连接，同时主机 2 进入 LAST_ACK 状态；

4) 第四次挥手：主机 1 收到主机 2 发送的 FIN 报文段，向主机 2 发送 ACK 报文段，然后主机 1 进入 TIME_WAIT 状态；主机 2 收到主机 1 的 ACK 报文段以后，就关闭连接；此时，主机 1 等待 2MSL 后依然没有收到回复，则证明 Server 端已正常关闭，那好，主机 1 也可以关闭连接了。



建立连接后，客户端和服务器都处于 ESTABLISED 状态。这时，客户端发起断开连接的请求：

1) 客户端调用 close() 函数后，向服务器发送 FIN 数据包，进入 FIN_WAIT_1 状态。FIN 是 Finish 的缩写，表示完成任务需要断开连接。

2) 服务器收到数据包后，检测到设置了 FIN 标志位，知道要断开连接，于是向客户端发送“确认包”，进入 CLOSE_WAIT 状态。

注意：服务器收到请求后并不是立即断开连接，而是先向客户端发送“确认包”，告诉它我知道了，我需要准备一下才能断开连接。

3) 客户端收到“确认包”后进入 FIN_WAIT_2 状态，等待服务器准备完毕后再次发送数据包。

4) 等待片刻后，服务器准备完毕，可以断开连接，于是再主动向客户端发送 FIN 包，告诉它我准备好了，断开连接吧。然后进入 LAST_ACK 状态。

5) 客户端收到服务器的 FIN 包后，再向服务器发送 ACK 包，告诉它你断开连接吧。然后进入 TIME_WAIT 状态。

6) 服务器收到客户端的 ACK 包后，就断开连接，关闭套接字，进入 CLOSED 状态。

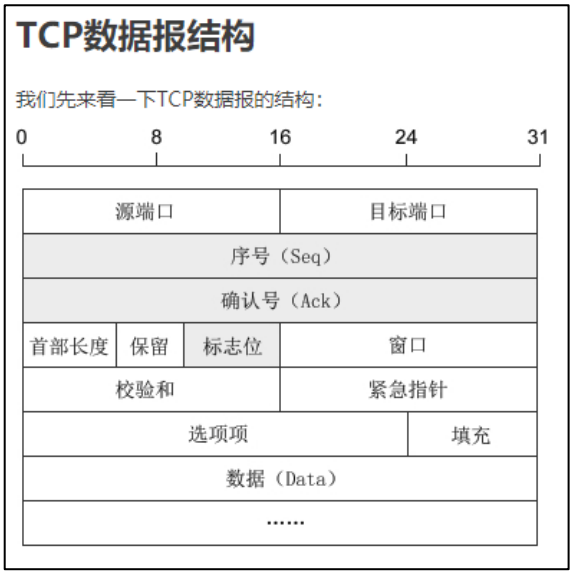
客户端最后一次发送 ACK 包后进入 TIME_WAIT 状态，而不是直接进入 CLOSED 状态关闭连接，这是为什么呢?

TCP 是面向连接的传输方式，必须保证数据能够正确到达目标机器，不能丢失或出错，而网络是不稳定的，随时可能会毁坏数据，所以机器 A 每次向机器 B 发送数据包后，都要求机器 B”确认“，回传 ACK 包，告诉机器 A 我收到了，这样机器 A 才能知道数据传送成功了。如果机器 B 没有回传 ACK 包，机器 A 会重新发送，直到机器 B 回传 ACK 包。

客户端最后一次向服务器回传 ACK 包时，有可能会因为网络问题导致服务器收不到，服务器会再次发送 FIN 包，如果这时客户端完全关闭了连接，那么服务器无论如何也收不到 ACK 包了，所以客户端需要等待片刻、确认对方收到 ACK 包后才能进入 CLOSED 状态。那么，要等待多久呢?

数据包在网络中是有生存时间的，超过这个时间还未到达目标主机就会被丢弃，并通知源主机。这称为报文最大生存时间（MSL，Maximum Segment Lifetime）。TIME_WAIT 要等待 2MSL 才会进入 CLOSED 状态。ACK 包到达服务器需要 MSL 时间，服务器重传 FIN 包也需要 MSL 时间，2MSL 是数据包往返的最大时间，如果 2MSL 后还未收到服务器重传的 FIN 包，就说明服务器已经收到了 ACK 包。

六大标志位



1）序号：Seq（Sequence Number）序号占 32 位，用来标识从计算机 A 发送到计算机 B 的数据包的序号，计算机发送数据时对此进行标记。

2）确认号：Ack（Acknowledge Number）确认号占 32 位，客户端和服务器端都可以发送，Ack = Seq + 1。

3）标志位：每个标志位占用 1Bit，共有 6 个，分别为 URG、ACK、PSH、RST、SYN、FIN，具体含义如下：

URG：紧急指针（urgent pointer）有效。

ACK：确认序号有效。

PSH：接收方应该尽快将这个报文交给应用层。

RST：重置连接。

SYN：建立一个新连接。(Synchronous)

FIN：断开一个连接。(Finish)

13. TCP 为啥挥手要比握手多一次?

因为当处于 LISTEN 状态的服务器端收到来自客户端的 SYN 报文(客户端希望新建一个 TCP 连接)时，它可以把 ACK(确认应答)和 SYN(同步序号)放在同一个报文里来发送给客户端。但在关闭 TCP 连接时，当收到对方的 FIN 报文时，对方仅仅表示对方已经没有数据发送给你了，但是你自己可能还有数据需要发送给对方，则等你发送完剩余的数据给对方之后，再发送 FIN 报文给对方来表示你数据已经发送完毕，并请求关

闭连接，所以通常情况下，这里的 ACK 报文和 FIN 报文都是分开发送的。

14. 为什么一定进行三次握手?

当客户端向服务器端发送一个连接请求时，由于某种原因长时间驻留在网络节点中，无法达到服务器端，由于 TCP 的超时重传机制，当客户端在特定的时间内没有收到服务器端的确认应答信息，则会重新向服务器端发送连接请求，且该连接请求得到服务器端的响应并正常建立连接，进而传输数据，当数据传输完毕，并释放了此次 TCP 连接。若此时第一次发送的连接请求报文段延迟了一段时间后，到达了服务器端，本来这是一个早已失效的报文段，但是服务器端收到该连接请求后误以为客户端又发出了一次新的连接请求，于是服务器端向客户端发出确认应答报文段，并同意建立连接。如果没有采用三次握手建立连接，由于服务器端发送了确认应答信息，则表示新的连接已成功建立，但是客户端此时并没有向服务器端发出任何连接请求，因此客户端忽略服务器端的确认应答报文，更不会向服务器端传输数据。而服务器端却认为新的连接已经建立了，并在一直等待客户端发送数据，这样服务器端一直处于等待接收数据，直到超出计数器的设定值，则认为服务器端出现异常，并且关闭这个连接。在这个等待的过程中，浪费服务器的资源。如果采用三次握手，客户端就不会向服务器发出确认应答消息，服务器端由于没有收到客户端的确认应答信息，从而判定客户端并没有请求建立连接，从而不建立该连接。

15. TCP 与 UDP 的区别？应用场景都有哪些?

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) 若通信数据完整性需让位与通信实时性，则应该选用 TCP 协议（如文件传输、重要状态的更新等）；反之，则使用 UDP 协议（如视频传输、实时通信等）。
7) UDP:DNS　SNMP

8) TCP 面向字节流，UTP 面向数据包；

16. 为什么 UDP 有时比 TCP 更有优势?

1) 网速的提升给 UDP 的稳定性提供可靠网络保障，丢包率很低，如果使用应用层重传，能够确保传输的可靠性。

2) TCP 为了实现网络通信的可靠性，使用了复杂的拥塞控制算法，建立了繁琐的握手过程，由于 TCP 内置的系统协议栈中，极难对其进行改进。

3) 采用 TCP，一旦发生丢包，TCP 会将后续的包缓存起来，等前面的包重传并接收到后再继续发送，延时会越来越大，基于 UDP 对实时性要求较为严格的情况下，采用自定义重传机制，能够把丢包产生的延迟降到最低，尽量减少网络问题对游戏性造成影响。

17. UDP 中一个包的大小最大能多大

1) 以太网(Ethernet)数据帧的长度必须在 46-1500 字节之间，这是由以太网的物理特性决定的。这个 1500 字节被称为链路层的 MTU(最大传输单元)。但这并不是指链路层的长度被限制在 1500 字节，其实这这个 MTU 指的是链路层的数据区。

2) 并不包括链路层的首部和尾部的 18 个字节。所以，事实上，这个 1500 字节就是网络层 IP 数据报的长度限制。因为 IP 数据报的首部为 20 字节，所以 IP 数据报的数据区长度最大为 1480 字节。

3) 而这个 1480 字节就是用来放 TCP 传来的 TCP 报文段或 UDP 传来的 UDP 数据报的。又因为 UDP 数据报的首部 8 字节，所以 UDP 数据报的数据区最大长度为 1472 字节。这个 1472 字节就是我们可以使用的字节数。

18. TCP 粘包

1) 在 socket 网络程序中，TCP 和 UDP 分别是面向连接和非面向连接的。因此 TCP 的 socket 编程，收发两端（客户端和服务器端）都要有成对的 socket，因此，发送端为了将多个发往接收端的包，更有效的发到对方，使用了优化方法（Nagle 算法），将多次间隔较小、数据量小的数据，合并成一个大的数据块，然后进行封包。这样，接收端，就难于分辨出来了，必须提供科学的拆包机制。

2) 对于 UDP，不会使用块的合并优化算法，这样，实际上目前认为，是由于 UDP 支持的是一对多的模式，所以接收端的 skbuff(套接字缓冲区）采用了链

式结构来记录每一个到达的 UDP 包，在每个 UDP 包中就有了消息头（消息来源地址，端口等信息），这样，对于接收端来说，就容易进行区分处理了。所以 UDP 不会出现粘包问题

1) TCP 粘包是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；

2) 发送方原因

我们知道，TCP 默认会使用 Nagle 算法。而 Nagle 算法主要做两件事：1）只有上一个分组得到确认，才会发送下一个分组；2）收集多个小分组，在一个确认到来时一起发送。所以，正是 Nagle 算法造成了发送方有可能造成粘包现象。

3) 接收方原因

TCP 接收到分组时，并不会立刻送至应用层处理，或者说，应用层并不一定会立即处理；实际上，TCP 将收到的分组保存至接收缓存里，然后应用程序主动从缓存里读收到的分组。这样一来，如果 TCP 接收分组的速度大于应用程序读分组的速度，多个包就会被存至缓存，应用程序读时，就会读到多个首尾相接粘到一起的包。

4) 解决方法

① **发送方**

对于发送方造成的粘包现象，我们可以通过关闭 Nagle 算法来解决，使用 TCP_NODELAY 选项来关闭 Nagle 算法。

② **接收方**

遗憾的是 TCP 并没有处理接收方粘包现象的机制，我们只能在应用层进行处理。

③ **应用层处理**

应用层的处理简单易行！并且不仅可以解决接收方造成的粘包问题，还能解决发送方造成的粘包问题。

19. 传输层功能

传输进程到进程的逻辑通信，即所说的端到端的通信，而网络层完成主机到主机之间

的逻辑通信；

20. TCP 可靠性保证

1. 序号

   TCP 首部的序号字段用来保证数据能有序提交给应用层，TCP 把数据看成无结构的有序的字节流。数据流中的每一个字节都编上一个序号字段的值是指本报文段所发送的数据的第一个字节序号。

2. 确认

   TCP 首部的确认号是期望收到对方的下一个报文段的数据的第一个字节的序号；

3. 重传

   超时重传

   冗余 ACK 重传

4. 流量控制

   TCP 采用大小可变的滑动窗口进行流量控制，窗口大小的单位是字节。

   发送窗口在连接建立时由双方商定。但在通信的过程中，接收端可根据自己的资源情况，随时动态地调整对方的发送窗口上限值(可增大或减小)。

   1) 窗口

      接受窗口 rwnd，接收端缓冲区大小。接收端将此窗口值放在 TCP 报文的首部中的窗口字段，传送给发送端。

      拥塞窗口 cwnd，发送缓冲区大小。

      发送窗口 swnd，发送窗口的上限值 = Min [rwnd, cwnd]

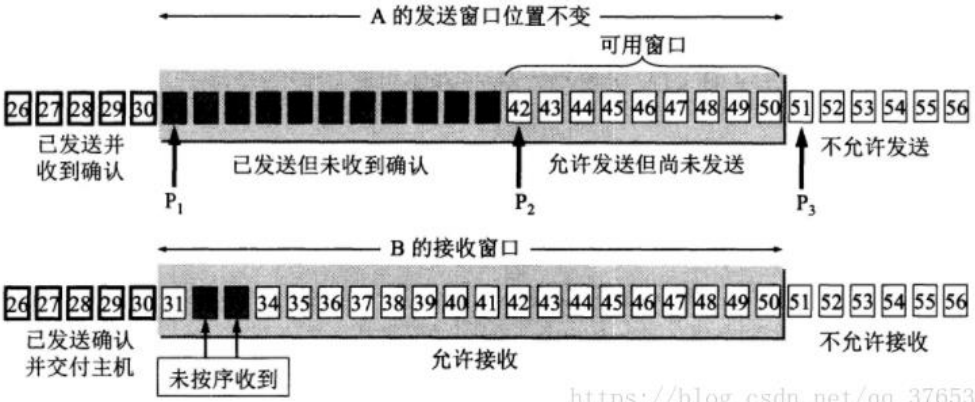
5. 拥塞控制

6. 流量控制与拥塞控制的区别

   所谓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。拥塞控制所要做的都有一个前提，就是网络能承受现有的网络负荷。流量控制往往指的是点对点通信量的控制，是个端到端的问题。流量控制所要做的就是控制发送端发送数据的速率，以便使接收端来得及接受。

21. TCP 滑动窗口

TCP 连接的两端都可收发数据，连接的收发数据量是通过一组窗口结构（window structure）来维护的



每个 TCP 连接的两端都维护一组窗口：发送窗口结构（send window structure）和接收窗口结构（receive window structure）。TCP 以字节为单位维护其窗口结构。随着发送端接收到返回的数据 ACK，滑动窗口也随之右移。发送端根据接收端返回的 ACK 可以得到两个重要的信息：一是接收端期望收到的下一个字节序号；二是当前的窗口大小

22. 拥塞控制

TCP 拥塞控制的目标是最大化利用网络上瓶颈链路的带宽。

网络内尚未被确认收到的数据包数量 = 网络链路上能容纳的数据包数量 = 链路带宽 × 往返延迟。拥塞窗口的大小取决于网络的拥塞程度，并且动态地在变化。常见的 TCP 拥塞控制算法：Reno 和 BBR

Reno 分为慢启动、拥塞避免、快重传和快恢复

1) 慢开始

发送方维持一个叫做**拥塞窗口 cwnd (congestion window)** 的状态变量。拥塞窗口的大小取决于网络的拥塞程度，并且动态地在变化。发送方让自己的发送窗口等于拥塞窗口，另外考虑到接受方的接收能力，发送窗口可能小于拥塞窗口。慢开始算法的思路就是，不要一开始就发送大量的数据，先探测一下网络的拥塞程度，也就是说由小到大逐渐增加拥塞窗口的大小。

当然收到单个确认但此确认多个数据报的时候就加相应的数值。所以一次传输轮次之后拥塞窗口就加倍。这就是乘法增长，和后面的拥塞避免算法的加法增长比较。

为了防止 cwnd 增长过大引起网络拥塞，还需设置一个慢开始门限 ssthresh 状态变量。ssthresh 的用法如下：

当 cwnd<ssthresh 时，使用慢开始算法。

当 cwnd>ssthresh 时，改用拥塞避免算法。

当 cwnd=ssthresh 时，慢开始与拥塞避免算法任意。

拥塞避免算法让拥塞窗口缓慢增长，即每经过一个往返时间 RTT 就把发送方的拥塞窗口 cwnd 加 1，而不是加倍。这样拥塞窗口按线性规律缓慢增长。

无论是在**慢开始阶段**还是在**拥塞避免阶段**，只要发送方判断网络出现拥塞（其根据就是没有收到确认，虽然没有收到确认可能是其他原因的分组丢失，但是因为无法判定，所以都当做拥塞来处理），就把慢开始门限设置为出现拥塞时的发送窗口大小的一半。然后把拥塞窗口设置为 1，执行慢开始算法。如下图：

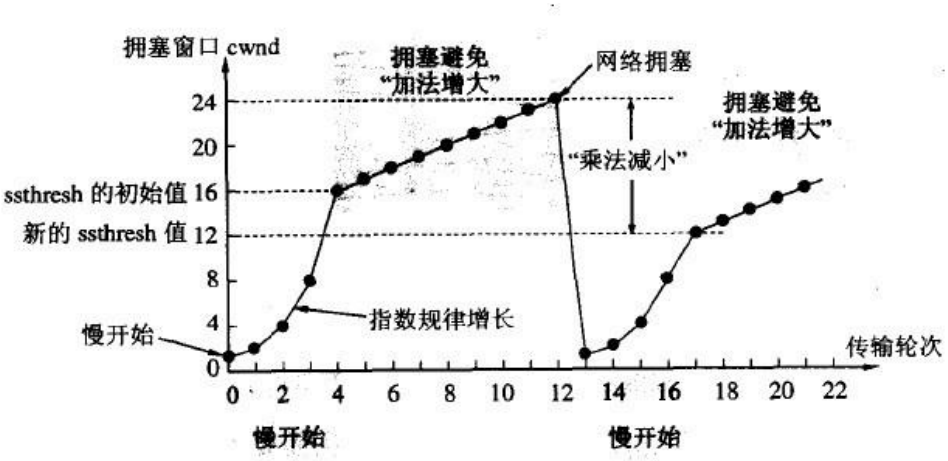


图 5-25　慢开始和拥塞避免算法的实现举例 net/sicofield

2)　快重传和快恢复

快重传要求接收方在收到一个失序的报文段后就立即发出重复确认（为的是使发送方及早知道有报文段没有到达对方）而不要等到自己发送数据时捎带确认。快重传算法规定，发送方只要一连收到三个重复确认就应当立即重传对方尚未收到的报文段，而不必继续等待设置的重传计时器时间到期。

快重传配合使用的还有快恢复算法，有以下两个要点:

①当发送方连续收到三个重复确认时，就执行“乘法减小”算法，把 ssthresh 门限减半。但是接下去并不执行慢开始算法。

②考虑到如果网络出现拥塞的话就不会收到好几个重复的确认，所以发送方现在认为网络可能没有出现拥塞。所以此时不执行慢开始算法，而是将 cwnd 设置为 ssthresh 的大小，然后执行拥塞避免算法。如下图：

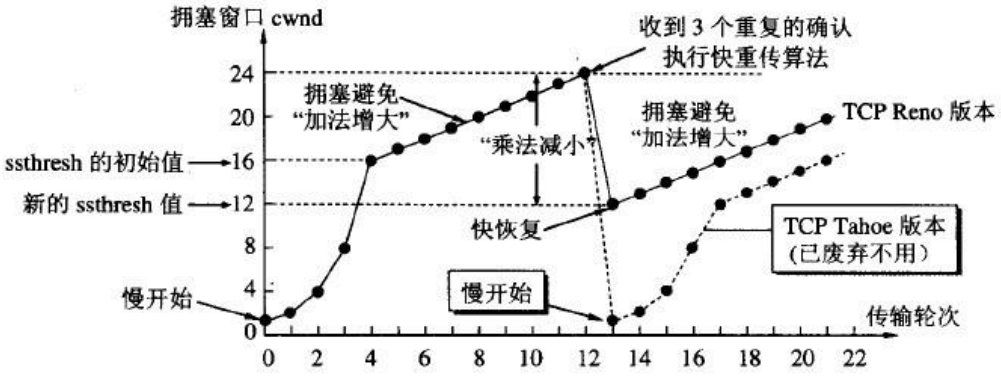


图 5-27　从连续收到三个重复的确认转入拥塞避免 net/sicofield

BBR

BBR 算法不将出现丢包或时延增加作为拥塞的信号，而是认为当网络上的数据包总量大于瓶颈链路带宽和时延的乘积时才出现了拥塞，所以 BBR 也称为基于拥塞的拥塞控制算法（Congestion-Based Congestion Control），其适用网络为高带宽、高时延、有一定丢包率的长肥网络，可以有效降低传输时延，并保证较高的吞吐量。BBR 算法周期性地探测网络的容量，交替测量一段时间内的带宽极大值和时延极小值，将其乘积作为作为拥塞窗口大小，使得拥塞窗口始的值始终与网络的容量保持一致。所以 BBR 算法解决了两个比较主要的问题：①在有一定丢包率的网络链路上充分利用带宽。适合高延迟、高带宽的网络链路。②降低网络链路上的 buffer 占用率，从而降低延迟。适合慢速接入网络的用户。

23. 三次握手过程中有哪些不安全性

SYN Flood：伪装的 IP 向服务器发送一个 SYN 请求建立连接，然后服务器向该 IP 回复 SYN 和 ACK，但是找不到该 IP 对应的主机，当超时时服务器收不到 ACK 会重复发送。当大量的攻击者请求建立连接时，服务器就会存在大量未完成三次握手的连接，服务器主机 backlog 被耗尽而不能响应其它连接。

防范措施：①采用 SYN cookie 设置，如果短时间内连续收到某个 IP 的重复 SYN 请求，

则认为受到了该 IP 的攻击，丢弃来自该 IP 的后续请求报文；②在网关处设置过滤，拒绝将一个源 IP 地址不属于其来源子网的包进行更远的路由③无效连接监视释放，不停的监视系统中半开连接和不活动连接，当达到一定阈值时拆除这些连接，释放系统资源④Syn Cache 技术，在收到 SYN 时不急着去分配 TCB，而是先回应一个 ACK 报文，并在一个专用的 HASH 表中（Cache）中保存这种半开连接，直到收到正确的 ACK 报文再去分配 TCB。

24. TCP 流量控制

1) 如果发送方把数据发送得过快，接收方可能会来不及接收，这就会造成数据的丢失。TCP 的流量控制是利用滑动窗口机制实现的，接收方在返回的 ACK 中会包含自己的接收窗口的大小，以控制发送方的数据发送。

2) 当某个 ACK 报文丢失了，就会出现 A 等待 B 确认，并且 B 等待 A 发送数据的死锁状态。为了解决这种问题，TCP 引入了持续计时器（Persistence timer），当 A 收到 rwnd=0 时，就启用该计时器，时间到了则发送一个 1 字节的探测报文，询问 B 是很忙还是上个 ACK 丢失了，然后 B 回应自身的接收窗口大小，返回仍为 0（A 重设持续计时器继续等待）或者会重发 rwnd=x。

25. 流量控制与拥塞控制的区别?

1) 拥塞控制就是防止过多的数据注入网络中，这样可以使网络中的路由器或链路不致过载。拥塞控制是一个全局性的过程，和流量控制不同，流量控制指点对点通信量的控制。
2) 所谓流量控制就是让发送速率不要过快，让接收方来得及接收。利用滑动窗口机制就可以实施流量控制。原理这就是运用 TCP 报文段中的窗口大小字段来控制，发送方的发送窗口不可以大于接收方发回的窗口大小。

26. time_wait 与 close_wait，time_wait 状态持续多长时间？为什么会有 time_wait 状态?

1) time_wait 另一边已经初始化一个释放，close_wait 连接一端被动关闭；
2) 首先调用 close()发起主动关闭的一方，在发送最后一个 ACK 之后会进入 time_wait 的状态，也就说该发送方会保持 2MSL 时间之后才会回到初始状态。MSL 指的是数据包在网络中的最大生存时间。产生这种结果使得这个 TCP 连接在 2MSL 连接等待期间，定义这个连接的四元组（客户端 IP 地址和端口，服务端 IP 地址和端口号）不能被使用。

3) 为什么存在 time_wait

① TCP 协议在关闭连接的四次握手过程中，最终的 ACK 是由主动关闭连接的一端（后面统称 A 端）发出的，如果这个 ACK 丢失，对方（后面统称 B 端）将重发出最终的 FIN，因此 A 端必须维护状态信息（TIME_WAIT）允许它重发最终的 ACK。如果 A 端不维持 TIME_WAIT 状态，而是处于 CLOSED 状态，那么 A 端将响应 RST 分节，B 端收到后将此分节解释成一个错误。因而，要实现 TCP 全双工连接的正常终止，必须处理终止过程中四个分节任何一个分节的丢失情况，主动关闭连接的 A 端必须维持 TIME_WAIT 状态 。

为实现 TCP 全双工连接的可靠释放

由 TCP 状态变迁图可知，假设发起主动关闭的一方（client）最后发送的 ACK 在网络中丢失，由于 TCP 协议的重传机制，执行被动关闭的一方（server）将会重发其 FIN，在该 FIN 到达 client 之前，client 必须维护这条连接状态，也就说这条 TCP 连接所对应的资源（client 方的 local_ip,local_port）不能被立即释放或重新分配，直到另一方重发的 FIN 达到之后，client 重发 ACK 后，经过 2MSL 时间周期没有再收到另一方的 FIN 之后，该 TCP 连接才能恢复初始的 CLOSED 状态。如果主动关闭一方不维护这样一个 TIME_WAIT 状态，那么当被动关闭一方重发的 FIN 到达时，主动关闭一方的 TCP 传输层会用 RST 包响应对方，这会被对方认为是有错误发生，然而这事实上只是正常的关闭连接过程，并非异常。

② TCP segment 可能由于路由器异常而“迷途”，在迷途期间，TCP 发送端可能因确认超时而重发这个 segment，迷途的 segment 在路由器修复后也会被送到最终目的地，这个迟到的迷途 segment 到达时可能会引起问题。在关闭“前一个连接”之后，马上又重新建立起一个相同的 IP 和端口之间的“新连接”，“前一个连接”的迷途重复分组在“前一个连接”终止后到达，而被“新连接”收到了。为了避免这个情况，TCP 协议不允许处于 TIME_WAIT 状态的连接启动一个新的可用连接，因为 TIME_WAIT 状态持续 2MSL，就可以保证当成功建立一个新 TCP 连接的时候，来自旧连接重复分组已经在网络中消逝。

为使旧的数据包在网络因过期而消失

为说明这个问题，我们先假设 TCP 协议中不存在 TIME_WAIT 状态的限制，再假设当前有一条 TCP 连接：(local_ip, local_port, remote_ip,remote_port)，因某些原因，我们先关闭，接着很快以相同的

四元组建立一条新连接。本文前面介绍过，TCP 连接由四元组唯一标识，因此，在我们假设的情况中，TCP 协议栈是无法区分前后两条 TCP 连接的不同的，在它看来，这根本就是同一条连接，中间先释放再建立的过程对其来说是“感知”不到的。这样就可能发生这样的情况：前一条 TCP 连接由 local peer 发送的数据到达 remote peer 后，会被该 remot peer 的 TCP 传输层当做当前 TCP 连接的正常数据接收并向上传递至应用层（而事实上，在我们假设的场景下，这些旧数据到达 remote peer 前，旧连接已断开且一条由相同四元组构成的新 TCP 连接已建立，因此，这些旧数据是不应该被向上传递至应用层的），从而引起数据错乱进而导致各种无法预知的诡异现象。作为一种可靠的传输协议，TCP 必须在协议层面考虑并避免这种情况的发生，这正是 TIME_WAIT 状态存在的第 2 个原因。

4) 如果 time_wait 维持的时间过长，主动关闭连接端迟迟无法关闭连接，占用程序资源。

5) 如果服务器程序 TCP 连接一直保持在 CLOSE_WAIT 状态，那么只有一种情况，就是在对方关闭连接之后服务器程序自己没有进一步发出 ack 信号。换句话说，就是在对方连接关闭之后，程序里没有检测到，或者程序压根就忘记了这个时候需要关闭连接，于是这个资源就一直被程序占着。

6) time_wait 状态如何避免

首先服务器可以设置 SO_REUSEADDR 套接字选项来通知内核，如果端口忙，但 TCP 连接位于 TIME_WAIT 状态时可以重用端口。在一个非常有用的场景就是，如果你的服务器程序停止后想立即重启，而新的套接字依旧希望使用同一端口，此时 SO_REUSEADDR 选项就可以避免 TIME_WAIT 状态。

Close_wait:

1) 产生原因

在被动关闭连接情况下，在已经接收到 FIN，但是还没有发送自己的 FIN 的时刻，连接处于 CLOSE_WAIT 状态。通常来讲，CLOSE_WAIT 状态的持续时间应该很短，正如 SYN_RCVD 状态。但是在一些特殊情况下，就会出现连接长时间处于 CLOSE_WAIT 状态的情况。出现大量 close_wait 的现象，主要原因是某种情况下对方关闭了 socket 链接，但是我方忙与读或者写，没有关闭连接。代码需要判断 socket，一旦读到 0，断开连接，read 返回负，检查一下 errno，如果不是 AGAIN，就断开连接。对方关闭连接之后服务器程序自己没有进一步发出 ack 信号。换句话说，就是在对方连接关闭之后，程序里没有检测到，或者程序压根就忘记了这个时候需要关闭连接，于是这个资源就一直被程序占着。

2) 解决方法

要检测出对方已经关闭的 socket，然后关闭它。

27. Time_wait 为什么是 2MSL 的时间长度

TIME_WAIT 的状态是为了等待连接上所有的分组的消失。单纯的想法，发送端只需要等待一个 MSL 就足够了。这是不够的，假设现在一个 MSL 的时候，接收端需要发送一个应答，这时候，我们也必须等待这个应答的消失，这个应答的消失也是需要一个 MSL，所以我们需要等待 2MSL。

28. 介绍一下 ping 的过程，分别用到了哪些协议

PING，用于测试网络连通性，基于 ICMP（使用时加 IP 报头）。工作原理：利用网络上机器 IP 地址的唯一性，给目标 IP 地址发送一个数据包，再要求对方返回一个同样大小的数据包来确定两台网络机器是否连接相通，时延是多少。

1. ping localhost:

localhost 的 IP 地址一般为 127.0.0.1，也称 loopback(环回路由)；如果此时 ping 不通，则表示协议栈有问题；ping 该地址不经过网卡，仅仅是软件层面

2. ping 本机 IP：

ping 本机 IP 其实是从驱动到网卡，然后原路返回；所以如果此时 ping 不通，则表示网卡驱动有问题，或者 NIC 硬件有问题；

3. ping 网关：

所谓网关，就是连接到另外一个网络的“关卡”， 一般为离我们终端最近的路由器；可以使用 ipconfig (windows)或 ifconfig (Linux)查看；若此时 ping 不通，则为主机到路由器间的网络故障；

4. ping 目的 IP:

若此步骤不成功，应该就是路由器到目的主机的网络有问题

同一网段

ping 通知系统建立一个固定格式的 ICMP 请求数据包。ICMP 协议打包这个数据包和机器 B 的 IP 地址转交给 IP 协议层（一组后台运行的进程，与 ICMP 类似）IP 层协议将以机器 B 的 IP 地址为目的地址，本机 IP 地址为源地址，加上一些其他的控制信息，构建一个 IP 数据包

获取机器 B 的 MAC 地址 IP 层协议通过机器 B 的 IP 地址和自己的子网掩码，发现它跟自己属同一网络，就直接在本网络查找这台机器的 MAC。若两台机器之前有过通信，在机器 A 的 ARP 缓存表应该有 B 机 IP 与其 MAC 的映射关系。若没有，则发送 ARP 请求广播，

得到机器 B 的 MAC 地址，一并交给数据链路层。数据链路层构建一个数据帧，目的地址是 IP 层传过来的 MAC 地址，源地址是本机的 MAC 地址，再附加一些控制信息，依据以太网的介质访问规则，将他们传送出去。机器 B 收到这个数据帧后，先检查目的地址，和本机 MAC 地址对比。符合，接收。接收后检查该数据帧，将 IP 数据包从帧中提取出来，交给本机的 IP 协议层协议。IP 层检查后，将有用的信息提取交给 ICMP 协议，后者处理后，马上构建一个 ICMP 应答包，发送给主机 A，其过程和主机 A 发送 ICMP 请求包到主机 B 类似（这时候主机 B 已经知道了主机 A 的 MAC 地址，不需再发 ARP 请求）。不符合，丢弃。

不同网段

ping 通知系统建立一个固定格式的 ICMP 请求数据包。ICMP 协议打包这个数据包和机器 B 的 IP 地址转交给 IP 协议层（一组后台运行的进程，与 ICMP 类似）。IP 层协议将以机器 B 的 IP 地址为目的地址，本机 IP 地址为源地址，加上一些其他的控制信息，构建一个 IP 数据包。获取主机 B 的 MAC 地址。IP 协议通过计算发现主机 B 与自己不在同一网段内，就直接交给路由处理，就是将路由的 MAC 取过来，至于怎么得到路由的 MAC 地址，和之前一样，先在 ARP 缓存表中寻找，找不到可以利用广播。路由得到这个数据帧之后，再跟主机 B 联系，若找不到，就向主机 A 返回一个超时信息。

29. socket 编程

TCP 过程：

客户端：

1) 创建 socket
2) 绑定 ip、端口号到 socket 字
3) 连接服务器，connect()
4) 收发数据，send()、recv()
5) 关闭连接

服务器端：

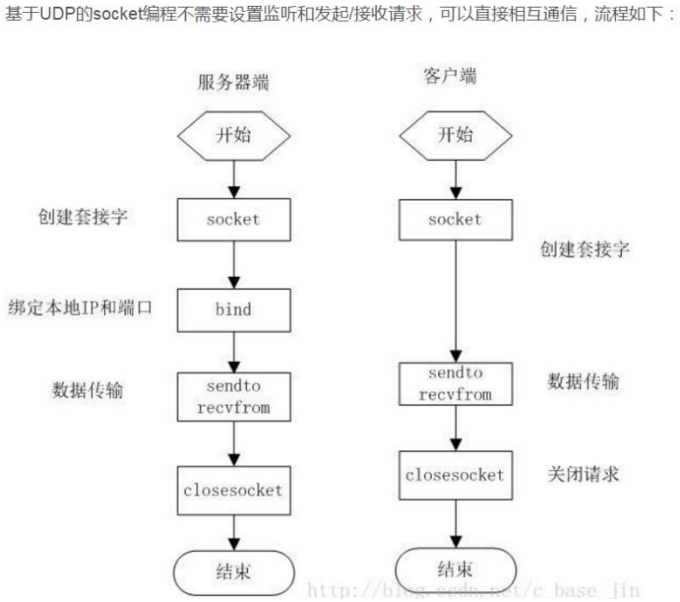
1) 创建 socket 字
2) 设置 socket 属性
3) 绑定 ip 与端口号
4) 开启监听，listen()
5) 接受发送端的连接 accept()
6) 收发数据 send()、recv()
7) 关闭网络连接
8) 关闭监听

对应关系：

客户端的 connect()指向服务器端的 accept()

客户端、服务器端的 send()/recv()是双向箭头的关系。

UDP 过程：



30. 客户端为什么不需要 bind

在很多场景下， 我们要在一个 pc 上开启多个客户端进程， 如果指定固定端口， 必然会造成端口冲突， 影响通信！所以， 我们就不要费力不讨好了， 客户端就不要指定端口了， 让操作系统来搞。这样，实际上就是操作系统对客户端的 socket 进行了隐式的命名（名称中的端口是随机的）。

31. send 和 recv 的缺点

四、 应用层

1. 常用的网络协议?

1) DHCP

动态主机设置协议（Dynamic Host Configuration Protocol, DHCP）是一个局域网的网络协议，使用 UDP 协议工作，主要有两个用途：给内部网络或网络服务供应商自动分配 IP 地址给用户给内部网络管理员作为对所有计算机作中央管理的手段

2) ARP

将 32 位的 IP 地址转换为 48 位的物理地址。当路由器或主机选择了某条路由时，首先会查找 ARP 缓存，若缓存中有对应 IP 地址的物理地址，则以此封装以太帧，否则会广播（为二层广播）ARP 报文，每个主机接收到 ARP 请求报文后，会缓存发送源的 IP——MAC 对到 ARP 缓存中，目的主机会发送 ARP 回应（此时为单播），当发送源接收到回应时，会将目的方的 IP——MAC 对存放在 ARP 缓存中。在点到点的物理连接中，是不会用到 ARP 报文的，在启动时双方都会通告对方自己的 IP 地址，此时物理层的封装不需要 MAC 地址。windows 上可以使用 arp -a 查看本机的 ARP 缓存。ARP 缓存中的每个条目的最大存活时间为 20 分钟

3) ICMP

ICMP（Internet Control Message Protocol）因特网控制报文协议。它是 IPv4 协议族中的一个子协议，用于 IP 主机、路由器之间传递控制消息。控制消息是在网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然不传输用户数据，但是对于用户数据的传递起着重要的作用。

ICMP 协议与 ARP 协议不同，ICMP 靠 IP 协议来完成任务，所以 ICMP 报文中要封装 IP 头部。它与传输层协议（如 TCP 和 UDP）的目的不同，一般不用来在端系统之间传送数据，不被用户网络程序直接使用，除了想 Ping 和 Tracert 这样的诊断程序。

2. 网络协议各个层的网络设备?

一、集线器

集线器也称 HUB，工作在 OSI 七层结构的第一层物理层，属于共享型设备，接收数据广播发出，在局域网内一般都是星型连接拓扑结构，每台工作站都连接到集线器上。由于集线器的带宽共享特性导致网络利用效率极低，一般在大中型的网络中不会使用到集线器。现在的集线器基本都是全双工模式，市面上常见的集线器传输速率普遍都为 100Mbps。

二、中继器

中继器（Repeater）工作于 OSI 的第一层（物理层），中继器是最简单的网络互联设备，连接同一个网络的两个或多个网段，主要完成物理层的功能，负责在两个网

络节点的物理层上按位传递信息，完成信号的复制、调整和放大功能，以此从而增加信号传输的距离，延长网络的长度和覆盖区域，支持远距离的通信。

一般来说，中继器两端的网络部分是网段，而不是子网。中继器只将任何电缆段上的数据发送到另一段电缆上，并不管数据中是否有错误数据或不适于网段的数据。大家最常接触的是网络中继器，在通讯上还有微波中继器、激光中继器、红外中继器等等，机理类似，触类旁通。

三、交换机

交换机顾名思义以交换为主要功能，工作在 OSI 第二层（数据链路层），根据 MAC 地址进行数据转发。交换机的每一个端口都属于一个冲突域，而集线器所有端口属于一个冲突域。交换机通过分析 Ethernet 包的包头信息 (其中包含了源 MAC 地址、目标 MAC 地址、信息长度等），取得目标 MAC 地址后，查找交换机中存储的地址对照表 (MAC 地址对应的端口)，确认具有此 MAC 地址的网卡连接在哪个端口上，然后将信包送到对应端口，有效的抑制 IP 广播风暴。并且信息包处于并行状态，效率较高。

交换机的转发延迟非常小，主要的得益于其硬件设计机理非常高效，为了支持各端口的最大数据传输速率，交换机内部转发信包的背板带宽都必须远大于端口带宽，具有强大的整体吞吐率，才能为每台工作站提供更高的带宽和更高的网络利用率，可以满足大型网络环境大量数据并行处理 的要求。

四、网桥

网桥和交换机一样都是工作在 OSI 模型的第二层（数据链路层），可以看成是一个二层路由器（真正的路由器是工作在网络层，根据 IP 地址进行信包转发）。网桥可有效的将两个局域网（LAN）连起来，根据 MAC 地址（物理地址）来转发帧，使本地通信限制在本网段内，并转发相应的信号至另一网段，网桥通常用于联接数量不多的、同一类型的网段。

五、路由器

路由器跟集线器和交换机不同，是工作在 OSI 的第三层（网络层），根据 IP 进行寻址转发数据包。路由器是一种可以连接多个网络或网段的网络设备，能将不同网络或网段之间（比如局域网——大网）的数据信息进行转换，并为信包传输分配最合适的路径，使它们之间能够进行数据传输，从而构成一个更大的网络。

路由器具有最主要的两个功能，即数据通道功能和控制功能。数据通道功能包括转发决定、背板转发以及输出链路调度等，一般由特定的硬件来完成；控制功能一般用软件来实现，包括与相邻路由器之间的信息交换、系统配置、系统管理等。

六、网关

网关（Gateway）又叫协议转换器，网关的概念实际上跟上面的设备型不是一类问题，但是为了方便参考还是放到这里一并介绍。

网关是一种复杂的网络连接设备，可以支持不同协议之间的转换，实现不同协议网络之间的互连。网关具有对不兼容的高层协议进行转换的能力，为了实现异构设备之间的通信，网关需要对不同的链路层、专用会话层、表示层和应用层协议进行翻译和转换。所以网关兼有路由器、网桥、中继器的特性。

若要使两个完全不同的网络（异构网）连接在一起，一般使用网关，在 Internet 中两个网络也要通过一台称为网关的计算机实现互联。这台计算机能根据用户通信目标计算机的 IP 地址，决定是否将用户发出的信息送出本地网络，同时，它还将外界发送给属于本地网络计算机的信息接收过来，它是一个网络与另一个网络相联的通道。为了使 TCP/IP 协议能够寻址，该通道被赋予一个 IP 地址，这个 IP 地址称为网关地址。

所以，网关的作用就是将两个使用不同协议的网络段连接在一起的设备，对两 个网络段中的使用不同传输协议的数据进行互相的翻译转换。在互连设备中，由于协议转换的复杂性，一般只能进行一对一的转换，或是少数几种特定应用协议的转换。

3. 讲讲浏览器输入地址后发生的全过程，以及对应的各个层次的过程

1、域名解析：浏览器获得 URL 地址，向操作系统请求该 URL 对应的 IP 地址，操作系统查询 DNS（首先查询本地 HOST 文件，没有则查询网络）获得对应的 IP 地址

解释：

把 URL 分割成几个部分：协议、网络地址、资源路径

协议：指从该计算机获取资源的方式，常见的是 HTTP、FTP

网络地址：可以是域名或者是 IP 地址，也可以包括端口号，如果不注明端口号，默认是 80 端口

如果地址不是一个 IP 地址，则需要通过 DNS（**UDP**）（域名系统）将该地址解析成 IP 地址，IP 地址对应着网络上的一台计算机，DNS 服务器本身也有 IP，你的网络设置包含 DNS 服务器的 IP，例如，www.abc.com 不是一个 IP，则需要向 DNS 询问请求 www.abc.com 对应的 IP，获得 IP，在这个过程中，你的电脑直接询问 DNS 服务器可能没有发现 www.abc.com 对应的 IP，就会向它的上级服务器询问，这样依次一层层向上级找，最高可达根节点，直到找到或者全部找不到为止

端口号就相当于银行的窗口，不同的窗口负责不同的服务，如果输入 www.abc.com:8080/，则表示不使用默认的 80 端口，而使用指定的 8080 端口

2、确认好了 IP 和端口号，则可以向该 IP 地址对应的服务器的该端口号发起 TCP 连接请求

3、服务器接收到 TCP 连接请求后，回复可以连接请求，

4、浏览器收到回传的数据后，还会向服务器发送数据包，表示三次握手结束

5、三次握手成功后，开始通讯，根据 HTTP 协议的要求，组织一个请求的数据包，里面包含请求的资源路径、你的身份信息等，例如，www.abc.com/images/1/表示的资源路径是 images/1/，发送后，服务器响应请求，将数据返回给浏览器，数据可以是根据 HTML 协议组织的网页，里面包含页面的布局、文字等等，也可以是图片或者脚本程序等，如果资源路径指定的资源不存在，服务器就会返回 404 错误，如果返回的是一个页面，则根据页面里的一些外链 URL 地址，重复上述步骤，再次获取

6、渲染页面，并开始响应用户的操作

7、窗口关闭时，浏览器终止与服务器的连接

4. http 与 https 工作方式

1) http 包含如下动作：

① 浏览器打开一个 TCP 连接；

② 浏览器发送 HTTP 请求到服务器；

③ 服务器发送 HTTP 回应信息到服务器；

④ TCP 连接关闭；

2) SSL 包含如下动作：

① 验证服务器端；

② 允许客户端和服务器端选择加密算法和密码，确保双方都支持；

③ 验证客户端；

④ 使用公钥加密技术来生成共享加密数据；

⑤ 创建一个加密的 SSL 连接；

⑥ 基于该 SSL 连接传递 HTTP 请求；

5. http 协议，http 和 https 的区别

1) HTTP 和 HTTPS 的基本概念

HTTP：是互联网上应用最为广泛的一种网络协议，是一个客户端和服务器端请求和应答的标准（TCP），用于从 WWW 服务器传输超文本到本地浏览器的传输协议，它可以使浏览器更加高效，使网络传输减少。

HTTPS：是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版，即 HTTP 下加入 SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。

HTTPS 协议的主要作用可以分为两种：一种是建立一个信息安全通道，来保证数据传输的安全；另一种就是确认网站的真实性。

2) HTTPS 和 HTTP 的区别主要如下：

a) https 协议需要到 ca 申请证书，一般免费证书较少，因而需要一定费用。

b) http 是超文本传输协议，信息是明文传输，https 则是具有安全性的 ssl 加密传输协议。

c) http 和 https 使用的是完全不同的连接方式，用的端口也不一样，前者是 80，后者是 443。

d) http 的连接很简单，是无状态的；HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，比 http 协议安全。

e) 在 OSI 模型中，HTTP 工作于应用层，而 HTTPS 工作于传输层；

6. http 状态码

1) 1XX 信息码，服务器收到请求，需要请求者继续执行操作；

2) 2XX 成功码，操作被成功接收并处理；

3) 3XX 重定向，需要进一步的操作以完成请求；

4) 4XX 客户端错误，请求包含语法错误或无法完成请求；

5) 5XX 服务器错误，服务器在处理请求的过程中发生了错误

404 服务器无法根据客户端的请求找到资源（网页）。通过此代码，网站设计人员可设置"您所请求的资源无法找到"的个性页面

7. HTTP1.0 与 HTTP1.1 的区别?

1) HTTP1.0 需要使用 keep-alive 参数来告知服务器要建立一个长连接，而 HTTP1.1 默认支持长连接；

2) HTTP 1.1 支持只发送 header 信息(不带任何 body 信息)，如果服务器认为客户端有权限请求服务器，则返回 100，否则返回 401；

3) HTTP1.0 是没有 host 域的，HTTP1.1 才支持这个参数；

4) HTTP2.0 使用了多路复用的技术，做到同一个连接并发处理多个请求，而且并发请求的数量比 HTTP1.1 大了好几个数量级；

5) HTTP1.1 不支持 header 数据的压缩，HTTP2.0 使用 HPACK 算法对 header 的数据进行压缩，这样数据体积小了，在网络上传输就会更快；

6) 对支持 HTTP2.0 的 web server 请求数据的时候，服务器会顺便把一些客户端需要的资源一起推送到客户端，免得客户端再次创建连接发送请求到服务器端获取。这种方式非常合适加载静态资源

8. cookie 和 session 的区别?

Session 是服务器用来跟踪用户的一种手段，每个 Session 都有一个唯一标识：Session ID。当服务器创建了一个 Session 时，给客户端发送的响应报文就包含了 Set-Cookie 字段，其中有一个名为 sid 的键值对，这个键值对就是 Session ID。客户端收到后就把 Cookie 保存在浏览器中，并且之后发送的请求报文都包含 Session ID。HTTP 就是 Session 和 Cookie 这两种方式一起合作来实现跟踪用户状态的，而 Session 用于服务器端，Cookie 用于客户端

9. OSI 7 层网络模型中各层的名称及其作用？

OSI 层	功能	TCP/IP 协议
应用层(Application layer)	文件传输，电子邮件，文件服务，虚拟终端	TFTP，HTTP，SNMP，FTP，SMTP，DNS，Telnet
表示层(Presentation layer)	数据格式化，代码转换，数据加密	没有协议
会话层(Session layer)	解除或建立与其他接点的联系	没有协议
传输层(Transport layer)	提供端对端的接口	TCP，UDP
网络层(Network layer)	为数据包选择路由	IP，ICMP，RIP，OSPF，BGP，IGMP
数据链路层(Data link layer)	传输有地址的帧，错误检测功能	SLIP，CSLIP，PPP，ARP，RARP，MTU
物理层(Physical layer)	以二进制数据形式在物理媒体上传输数据	ISO2110，IEEE802，IEEE802.2

10. TCP/IP 4 层网络模型名称及其作用？

OSI七层网络模型		Linux TCP/IP概念层	对应网络协议	相应措施
应用层（Application）			TFTP、FTP、NFS、WAIS	
表示层(Presentation)		应用层	Telnet、rlogin、SNMP、Gopher	Linux应用命令测试
会话层(Session)			SMTP、DNS	
传输层(Transport)	⟺	传输层	TCP、UDP	TCP、UDP协议分析
网络层(Network)	⟺	网际层	IP、ICMP、ARP、RARP、AKP、UUCP	检查IP地址、路由设置
数据链路层(Data Link)	⟺	网络接口	FDDI、Ethernet、Arpanet、PDN、SLP、PPP	ARP地址检测、物理连接检测
物理层(Physical)			IEEE 802.1A、IEEE 802.2到IEEE 802.11	

11. OSI 7 层网络中各层的常见协议以及协议作用？设备

第一层：物理层(PhysicalLayer)

规定通信设备的机械的、电气的、功能的和过程的特性，用以建立、维护和拆除物理链路连接。具体地讲，机械 特性规定了网络连接时所需接插件的规格尺寸、引脚数

量和排列情况等;电气特性规定了在物理连接上传输 bit 流时线路上信号电平的大小、阻抗匹配、传输速率 距离限制等;功能特性是指对各个信号先分配确切的信号含义，即定义了 DTE 和 DCE 之间各个线路的功能;规程特性定义了利用信号线进行 bit 流传输的一组 操作规程，是指在物理连接的建立、维护、交换信息是，DTE 和 DCE 双放在各电路上的动作系列。在这一层，数据的单位称为比特(bit)。属于物理层定义的典型规范代表包括：EIA/TIA RS-232、EIA/TIA RS-449、V.35、RJ-45 等。

第二层：数据链路层(DataLinkLayer)

在物理层提供比特流服务的基础上，建立相邻结点之间的数据链路，通过差错控制提供数据帧(Frame)在信道上无差错的传输，并进行各电路上的动作系列。数据链路层在不可靠的物理介质上提供可靠的传输。该层的作用包括：物理地址寻址、数据的成帧、流量控制、数据的检错、重发等。在这一层，数据的单位称为帧(frame)。数据链路层协议的代表包括：SDLC、HDLC、PPP、STP、帧中继等。

第三层是网络层

在 计算机网络中进行通信的两个计算机之间可能会经过很多个数据链路，也可能还要经过很多通信子网。网络层的任务就是选择合适的网间路由和交换结点， 确保数据及时传送。网络层将数据链路层提供的帧组成数据包，包中封装有网络层包头，其中含有逻辑地址信息- -源站点和目的站点地址的网络地址。如 果你在谈论一个 IP 地址，那么你是在处理第 3 层的问题，这是“数据包”问题，而不是第 2 层的“帧”。IP 是第 3 层问题的一部分，此外还有一些路由协议和地 址解析协议(ARP)。有关路由的一切事情都在这第 3 层处理。地址解析和路由是 3 层的重要目的。网络层还可以实现拥塞控制、网际互连等功能。在这一层，数据的单位称为数据包(packet)。网络层协议的代表包括：IP、IPX、RIP、OSPF 等。

第 四层是处理信息的传输层

第 4 层的数据单元也称作数据包(packets)。但是，当你谈论 TCP 等具体的协议时又有特殊的叫法，TCP 的数据单元称为段 (segments)而 UDP 协议的数据单元称为“数据报(datagrams)”。这个层负责获取全部信息，因此，它必须跟踪数据单元碎片、乱序到达的 数据包和其它在传输过程中可能发生的危险。第 4 层为上层提供端到端(最终用户到最终用户)的透明的、可靠的数据传输服务。所为透明的传输是指在通信过程中 传输层对上层屏蔽了通信传输系统的具体细节。传输层协议的代表包括：TCP、UDP、SPX 等。

第五层是会话层

这一层也可以称为会晤层或对话层，在会话层及以上的高层次中，数据传送的单位不再另外命名，而是统称为报文。会话层不参与具体的传输，它提供包括访问验证和会话管理在内的建立和维护应用之间通信的机制。如服务器验证用户登录便是由会话层完成的。

第六层是表示层

这一层主要解决拥护信息的语法表示问题。它将欲交换的数据从适合于某一用户的抽象语法，转换为适合于 OSI 系统内部使用的传送语法。即提供格式化的表示和转换数据服务。数据的压缩和解压缩， 加密和解密等工作都由表示层负责。

第七层应用层

应用层为操作系统或网络应用程序提供访问网络服务的接口。应用层协议的代表包括：Telnet、FTP、HTTP、SNMP 等

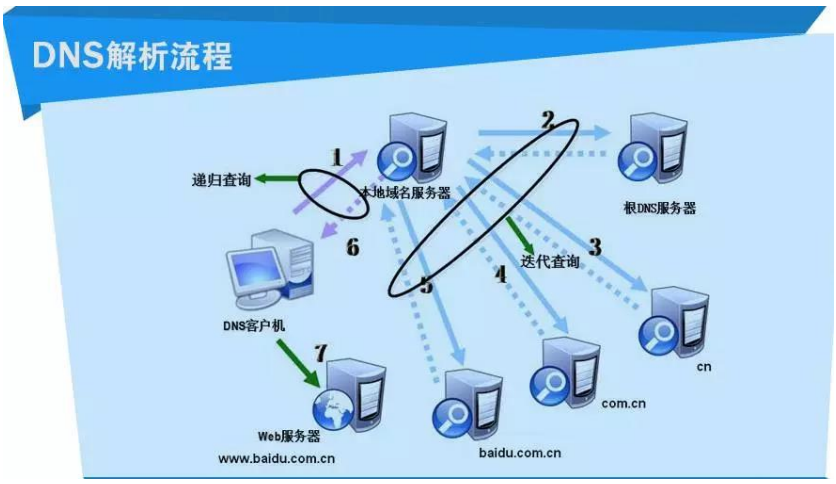
12. OSI 与 TCP 模型的区别?

1) TCP/IP 协议中的应用层处理开放式系统互联模型中的第五层、第六层和第七层的功能；

2) TCP/IP 协议中的传输层并不能总是保证在传输层可靠地传输数据包，而开放式系统互联模型可以做到。TCP/IP 协议还提供一项名为 UDP(用户数据报协议)的选择。UDP 不能保证可靠的数据包传输。

13. DNS 是干什么的？？

1) 主机解析域名的顺序
   找缓存、找本机的 hosts 文件、找 DNS 服务器
2) DNS 协议运行在 UDP 协议之上，使用端口号 53
3) 根服务器：ISP 的 DNS 服务器还找不到的话，它就会向根服务器发出请求，进行递归查询（DNS 服务器先问根域名服务器.com 域名服务器的 IP 地址，然后再问.com 域名服务器，依次类推）



十个过程：

① 浏览器先检查自身缓存中有没有被解析过这个域名对应的 ip 地址；

② 如果浏览器缓存没有命中，浏览器会检查操作系统缓存中有没有对应的已解析过的结果。在 windows 中可通过 c 盘里 hosts 文件来设置；

③ 还没命中，请求本地域名服务器来解析这个域名，一般都会在本地域名服务器找到；

④ 本地域名服务器没有命中，则去根域名服务器请求解析；

⑤ 根域名服务器返回给本地域名服务器一个所查询域的主域名服务器；

⑥ 本地域名服务器向主域名服务器发送请求；

⑦ 接受请求的主域名服务器查找并返回这个域名对应的域名服务器的地址；

⑧ 域名服务器根据映射关系找到 ip 地址，返回给本地域名服务器；

⑨ 本地域名服务器缓存这个结果；

⑩ 本地域名服务器将该结果返回给用户；

14. get/post 区别

1) 后退按钮或刷新，Get 无害，post 数据会被重新提交；
2) Get 所使用的 URL 可以被设置为书签，而 post 不可以；
3) Get 能够被缓存，而 post 不可以；
4) Get 参数保留在浏览器历史中，而 post 参数不会保留在浏览器历史中；
5) 当发生数据时，get 方法向 URL 添加数据，URL 的数据长度是受限的，而 post 没有数据长度限制；
6) Get 只允许 ASCII 编码，而 post 没有限制；
7) Get 安全性没有 post 安全性好；
8) Get 数据在 URL 中对所有人是可见的，而在 post 中数据不会显示在 URL 中。
9) Get 产生一个 TCP 数据包，post 产生两个 TCP 数据包；对于 get 方式的请求，浏览器会把 header 和 data 一并发送出去；对于 post，浏览器先发送 header 再发送

data；

10) GET 和 POST 本质上就是 TCP 链接，并无差别。但是由于 HTTP 的规定和浏览器/服务器的限制，导致他们在应用过程中体现出一些不同；

15. 说一下网卡从接收到数据后发生了什么

16. send 函数什么情况下会阻塞

17. 常用端口号

FTP 21;TELNET 23;SMTP 25;DNS 53;HTTP 80;HTTPS 443

18. https 的过程

19. 4 种 IO 模型

IO 有两种操作，同步 IO 和异步 IO。同步 IO 指的是，必须等待 IO 操作完成后，控制权才返回给用户进程。异步 IO 指的是，无须等待 IO 操作完成，就将控制权返回给用户进程。

4 种网络 IO 模型：①阻塞 IO 模型②非阻塞 IO 模型③多路 IO 复用模型④异步 IO 模型

①②③都属于同步 IO。

真实的 IO 操作，就是例子中的 recvfrom 这个系统调用。非阻塞 IO 在执行 recvfrom 这个系统调用的时候，如果内核的数据没有准备好，这时候不会阻塞进程。但是当内核中数据准备好时，recvfrom 会将数据从内核拷贝到用户内存中，这个时候进程则被阻塞。而异步 IO 则不一样，当进程发起 IO 操作之后，就直接返回，直到内核发送一个信号，告诉进程 IO 已完成，则在这整个过程中，进程完全没有被阻塞。非阻塞 I O 和异步 IO 的区别还是很明显的。在非阻塞 IO 中，虽然进程大部分时间都不会被阻塞，但是它仍然要求进程去主动检查，并且当数据准备完成以后，也需要进程主动地再次调用 r ecvfrom 来将数据拷贝到用户内存中。而异步 IO 则完全不同，它就像是用户进程将整个 IO 操作交给了他人（内核) 完成，然后内核做完后发信号通知。在此期间，用户进程不需要去检查 IO 操作的状态，也不需要主动地拷贝数据。（非阻塞 IO 和异步 IO 的区别）

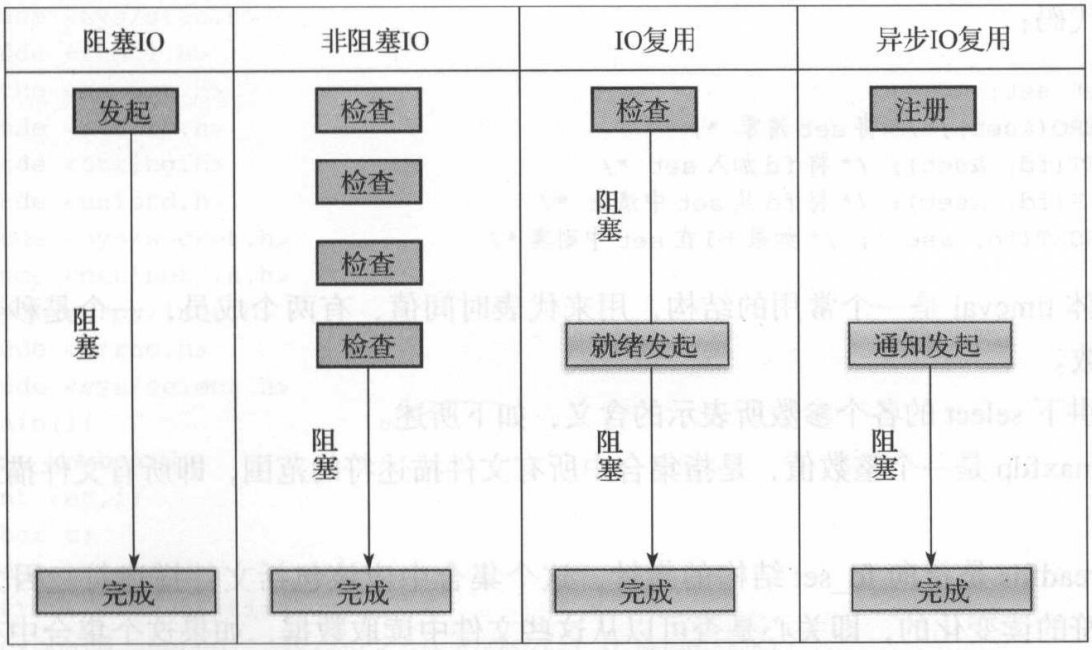


图 7-7　各种 IO 模型的比较

20.　Select、poll 和 epoll 的区别

https://www.cnblogs.com/aspirant/p/9166944.html

五、　其他相关面试题

1. 我想要检查目前我这部主机启动在网络端口口监听的服务有哪些，并且关闭不要的程序

Netstat -tulnp　　lsof 展示文件描述符，展示特定端口 展示进程占用

六、　HTTP

1.　URI 和 URL

URI 用字符串标识某一互联网资源，而 URL 表示资源的地点（互联网上所处的位置）。可见 URL 是 URI 的子集

2. HTTP Method

GET：获取资源，GET 方法用来请求访问已被 URI 识别的资源。指定的资源经服务器端解析后返回响应内容。也就是说，如果请求的资源是文本，那就保持原样返回；如果是像 CGI（Common Gateway Interface，通用网关接口）那样的程序，则返回经过执行后的输出结果。

POST：用来传输实体的主体

PUT：传输文件

HEAD：获取报文头部，只是不返回报文主体部分。用于确认 URI 的有效性及资源更新的日期时间等。

DELETE：删除文件

OPTIONS：询问支持的方法

3. 持久连接

HTTP 协议的初始版本中，每进行一次 HTTP 通信就要断开一次 TCP 连接。持久连接的特点是，只要任意一端没有明确提出断开连接，则保持 TCP 连接状态

持久连接的好处在于减少了 TCP 连接的重复建立和断开所造成的额外开销，减轻了服务器端的负载。另外，减少开销的那部分时间，使 HTTP 请求和响应能够更早地结束，这样 Web 页面的显示速度也就相应提高了。 在 HTTP/1.1 中，所有的连接默认都是持久连接。

4. 管线化

持久连接使得多数请求以管线化（pipelining）方式发送成为可能。从前发送请求后需等待并收到响应，才能发送下一个请求。管线化技术出现后，不用等待响应亦可直接发送下一个请求。

即持久化连接是一个请求对应一个响应，在响应未收到之前不会发送下一个请求；而管线化可持续发送请求，而不用等待响应

5. Cookie

HTTP 是无状态协议，它不对之前发生过的请求和响应的状态进行管理。也就是说，无法根据之前的状态进行本次的请求处理。

Cookie 技术通过在请求和响应报文中写入 Cookie 信息来控制客户端的状态。

发生 Cookie 交互的情景：Cookie 会根据从服务器端发送的响应报文内的一个叫做 Set-Cookie 的首部字段信息，通知客户端保存 Cookie。当下次客户端再往该服务器发送请求时，客户端会自动在请求报文中加入 Cookie 值后发送出去。

服务器端发现客户端发送过来的 Cookie 后，会去检查究竟是从哪一个客户端发来的连接请求，然后对比服务器上的记录，最后得到之前的状态信息。

6. 范围请求

为了实现一种资源可恢复的机制。所谓恢复是指能从之前下载中断处恢复下载。

要实现该功能需要指定下载的实体范围。像这样，指定范围发送的请求叫做范围请求（Range Request）执行范围请求时，会用到首部字段 Range 来指定资源的 byte 范围

7. HTTP 状态码

	类别	原因短语
1XX	Informational(信息性状态码)	接收到的请求正在处理
2XX	Success（成功状态码）	请求正常处理完毕
3XX	Redirection(重定向状态码)	需要进行附加操作以完成请求
4XX	Client Error（客户端错误状态码）	服务器无法处理请求
5XX	Server Error（服务端错误状态码）	服务器处理请求出错

200 OK　表明请求被正常处理

204 No Content　服务器接收的请求已成功处理，但在返回的响应报文中不含实体的主体部分

206 Partial Content　表示客户端进行了范围请求，而服务器成功执行了这部分的 GET 请求。

301 Moved Permanently　表示请求的资源已被分配了新的 URI，以后应使用资源现在所指的 URI （永久性重定向）

302 Found　表示请求的资源已被分配了新的 URI，希望用户（本次）能使用新的 URI

访问 （临时性重定向）

303 See Other 由于请求对应的资源存在着另一个 URI，应使用 GET 方法定向获取请求的资源

303 状态码和 302 Found 状态码有着相同的功能，但 303 状态码明确表示客户端应当采用 GET 方法获取资源，这点与 302 状态码有区别。

当 301、302、303 响应状态码返回时，几乎所有的浏览器都会把 POST 改成 GET，并删除请求报文内的主体，之后请求会自动再发送。301、302 标准是禁止将 POST 方法改变成 GET 方法的，但实际使用时大家都会这么做。

304 Not Modified 表示客户端发送附带条件的请求时，服务器端允许请求访问资源，但未满足条件的情况

400 Bad Request 该状态码表示请求报文中存在语法错误。

401 Unauthorized 表示发送的请求需要有通过 HTTP 认证（BASIC 认证、DIGEST 认证）的认证信息

403 Forbidden 表明对请求资源的访问被服务器拒绝了。服务器端没有必要给出拒绝的详细理由

404 Not Found 表明服务器上无法找到请求的资源。除此之外，也可以在服务器端拒绝请求且不想说明理由时使用。

500 Internal Server Error 表明服务器端在执行请求时发生了错误。也有可能是 Web 应用存在的 bug 或某些临时的故障

503 Service Unavailable 表明服务器暂时处于超负载或正在进行停机维护，现在无法处理请求。