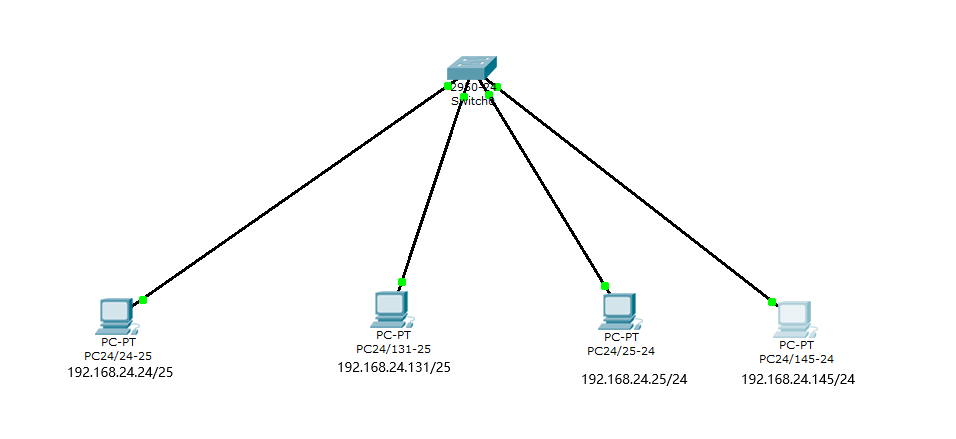
综合验证性实验：

实验一、网络命令的使用

第一部分：子网划分与连通性测试



某单位一个局域网中，使用交换机连接了4台计算机，它们的网络参数（IP地址/子网掩码）配置如图所示，请完成如下问题：

1. 对四个IP地址进行分析，哪些地址位于同一个子网，哪些地址网络地址相同，但是子网掩码不同。

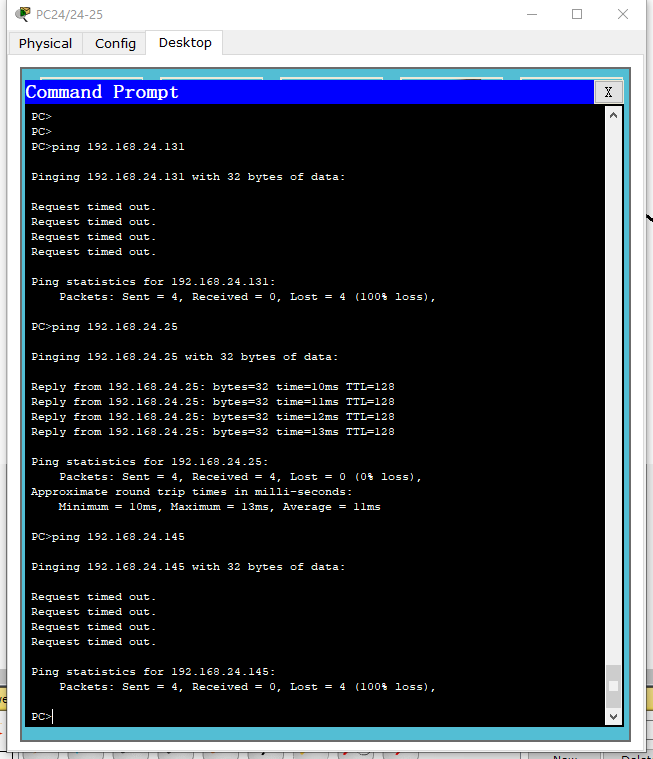
主机pc24/24-25与主机pc24/131-25位于同一子网

主机pc24/25-24与主机pc24/145-24位于同一子网

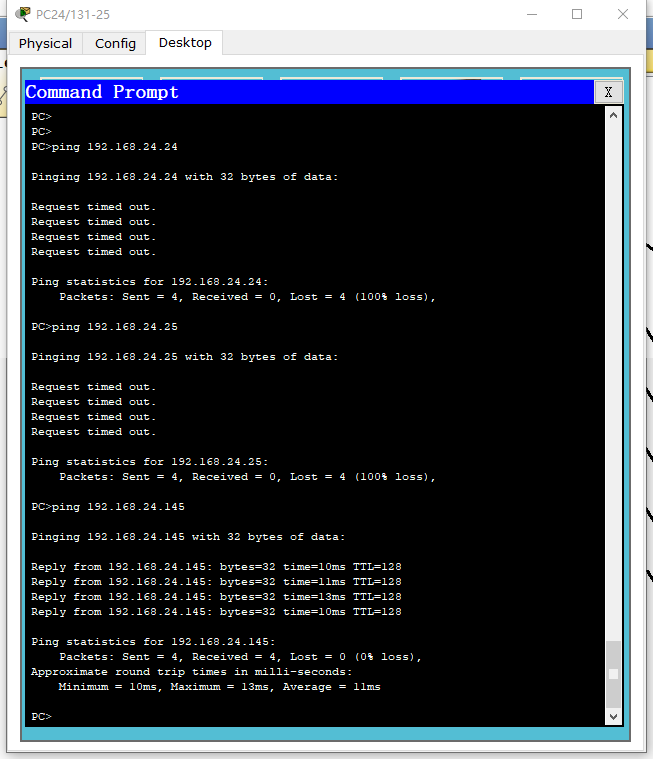
主机pc24/24-25与主机pc24/25-24网络地址相同，但是子网掩码不同

主机pc24/131-25与主机pc24/145-24网络地址相同，但是子网掩码不同

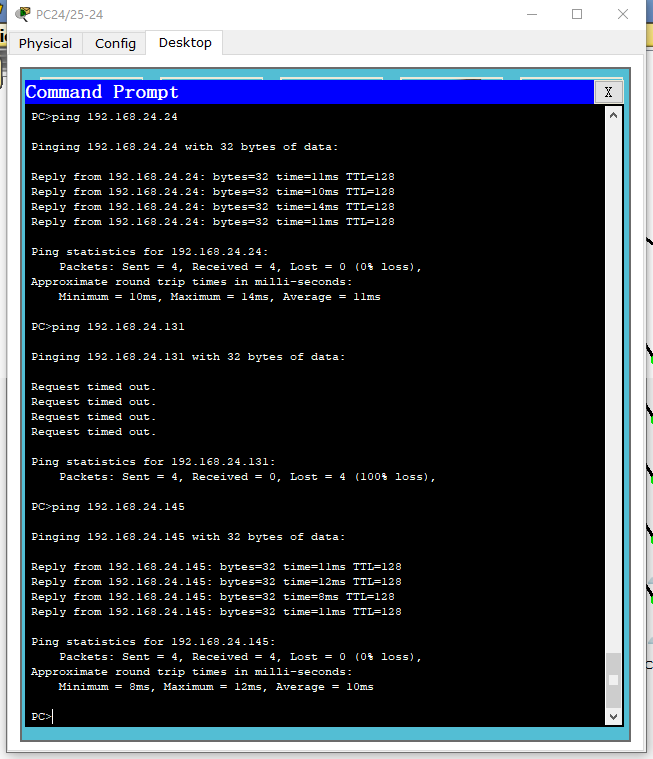
1. 主机之间互相ping，列出结果，并认真分析原因。



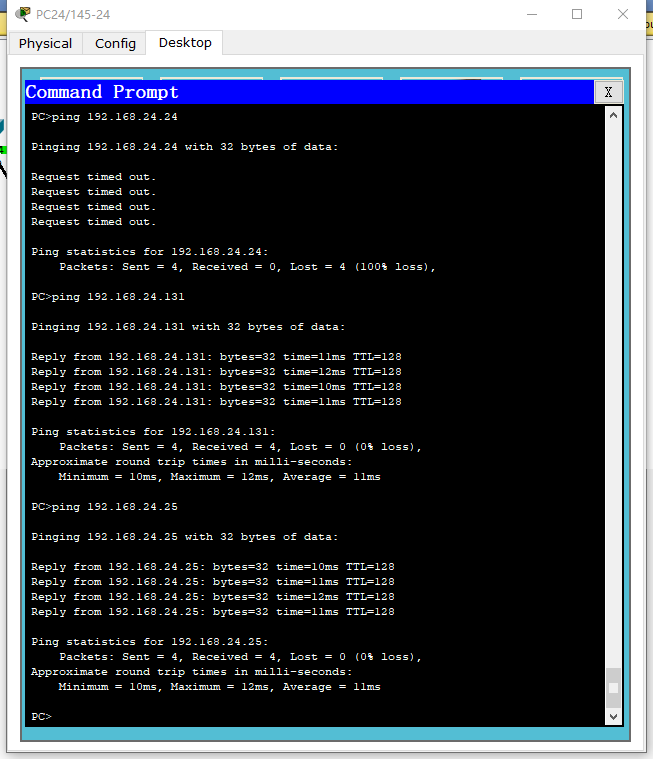
主机pc24/24-25只能与主机pc24/25-24进行ping通，原因是，虽然与主机pc24/131-25有着相同的子网掩码，但是主机pc24/24-25的第25位网络地址并不是1，与主机pc24/131-25不在同一个网段，所以无法ping通。它只能与第25位ip地址位为1的主机相互通信。



该主机的原因与上面的原因相似，就不在过多的赘述。



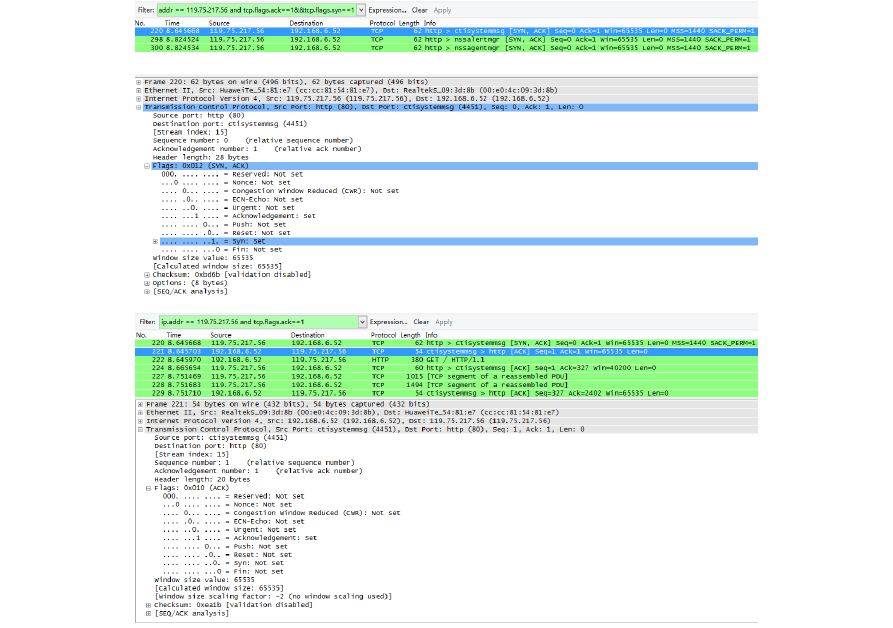
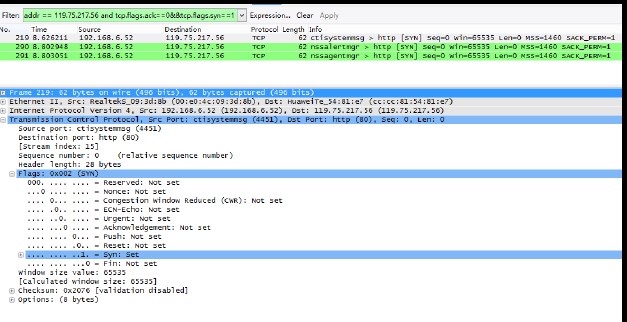
该主机与主机pc24/24-25虽然有着不同的子网掩码，但是他们属于相同的网段，所以可以通信，他与主机pc24/145-24有着相同的子网掩码和网络地址，他们也数以同一个网段，故也可以相互通信。



该主机的原因与上面的原因相似，就不在过多的赘述。

第二部分：深刻理解运输层协议

运输层是负责数据通信的最高层，充分理解和验证运输层协议有助于对数据通信、数据传输的深刻理解，以及对其下层协议的理解。请用Wireshark软件完成如下要求：



1. TCP三次握手全过程。三次交互。首部关键字段理解：序号、确认号、同步/确认开关量、接收窗口等。（知识点：TCP报文首部、TCP连接管理）

下面是TCP报文格式图：



图1 TCP报文格式

上图中有几个字段需要重点介绍下：

（1）序号：Seq序号，占32位，用来标识从TCP源端向目的端发送的字节流，发起方发送数据时对此进行标记。

（2）确认序号：Ack序号，占32位，只有ACK标志位为1时，确认序号字段才有效，Ack=Seq+1。

（3）标志位：共6个，即URG、ACK、PSH、RST、SYN、FIN等，具体含义如下：

（A）URG：紧急指针（urgent pointer）有效。

（B）ACK：确认序号有效。

（C）PSH：接收方应该尽快将这个报文交给应用层。

（D）RST：重置连接。

（E）SYN：发起一个新连接。

（F）FIN：释放一个连接。

需要注意的是：

（A）不要将确认序号Ack与标志位中的ACK搞混了。

（B）确认方Ack=发起方Req+1，两端配对。

二、三次握手

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：



（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。

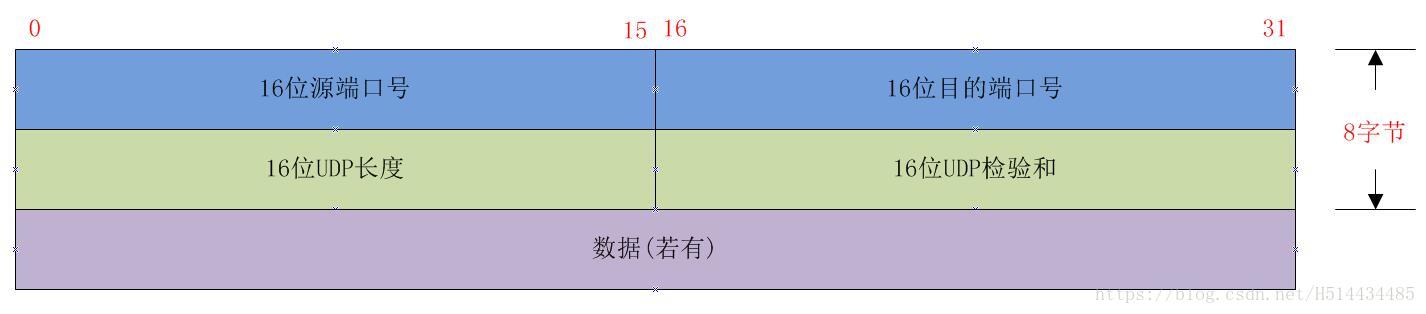
（2）第二次握手：Server收到数据包后由标志位SYN=1知道Client请求建立连接，Server将标志位SYN和ACK都置为1，ack=J+1，随机产生一个值seq=K，并将该数据包发送给Client以确认连接请求，Server进入SYN\_RCVD状态。

（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。

1. UDP报文传输，首部内容分析，端口等字段。（知识点：UDP协议）



UDP数据报文格式及首部中各字段



16位源端口、目的端口：

用来标识源端和目标端的应用进程。

16位UDP长度：

该字段表示UDP首部和UDP数据的字节长度。该字段的最小值为8字节（发送0字节的UDP数据报是OK的）。

16位UDP校验和：

UDP校验和覆盖了UDP首部和UDP数据。UDP的校验和是可选的，而TCP的校验和是必需的。

1. TCP停等协议，发送-确认交互，尤其观测是累计确认还是单一确认。（知识点：TCP可靠传输-停等协议）

确认重传，简单来说就是接收方收到报文以后给发送方一个 ACK 回复，说明自己已经收到了发送方发过来的数据。如果发送方等待了一个特定的时间还没有收到接收方的 ACK 他就认为数据包丢了，接收方没有收到就会重发这个数据包。

好的，上面的机制还是比较好理解的，但是我们会发现一个问题，那就是如果接收方已经收到了数据然后返回的 ACK 丢失，发送方就会误判导致重发。而此时接收方就会收到冗余的数据，但是接收方怎么能判定这个数据是冗余的还是新的数据呢？

这就涉及到了 TCP 的另外一个机制就是采用序号和确认号，也就是每次发送数据的时候这个报文段里面包括了当前报文段的序号和对上面的报文的确认号，这样我们的接收方可以根据自己接受缓存中已经有的数据来确定是否接受到了重复的报文段。这时候如果出现上面所说的 ACK 丢失，导致接受重复的报文段时客户端丢弃这个冗余的报文段。

好现在我们大致了解了确认重传机制，但是还有些东西还没有弄清楚，也就是 TCP 真正的实现究竟是怎样的。

确认是每发一个报文段就确认一次还是一次确认多个呢？

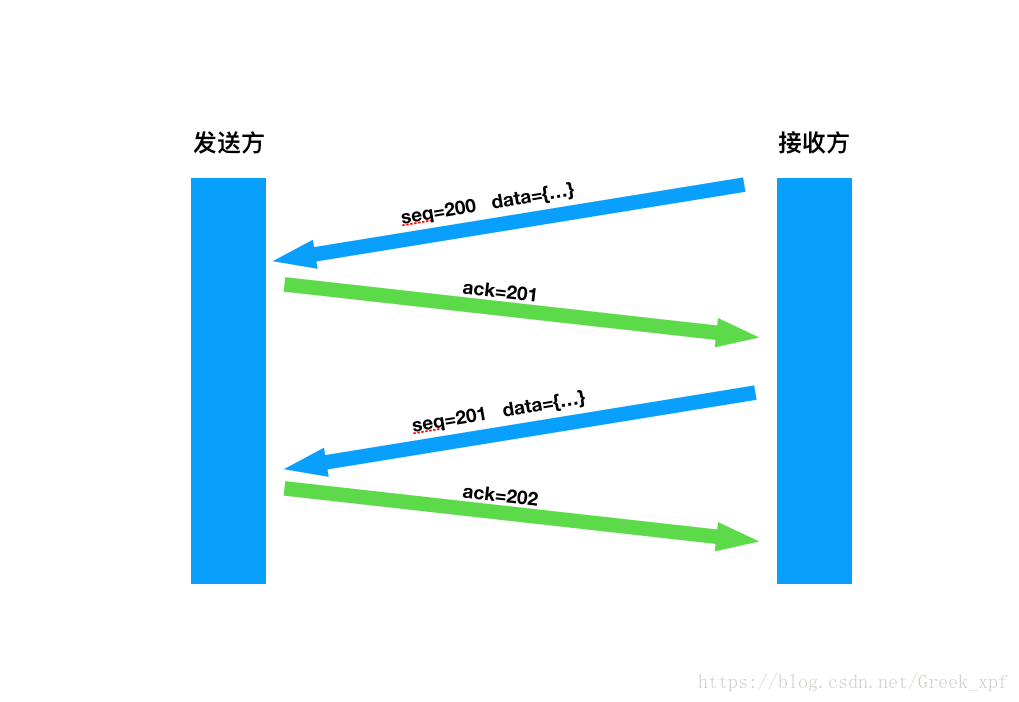
还有上面所说的发送方等待一个特定的时间，这个时间究竟等多长比较合适？

重传的时候是只重传那个没收到的报文还是重传那个报文段及它以后的报文段？

1.累计确认/单停等协议

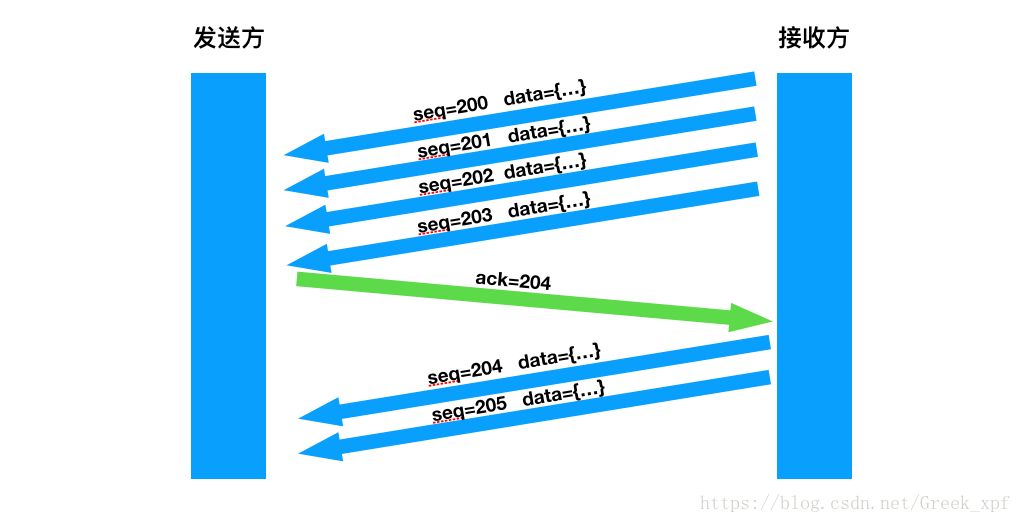
这就是我们要解决的第一个问题就是如何确认。这里涉及到两种确认方式，分别称为累计确认（捎带确认） 和 单停等协议 。

**单停等协议**



用一张图来快速理解，就是每发送一次数据，就进行一次确认。等发送方收到了 ACK 才能进行下一次的发送。

**累计确认**



一样的也是采用的 ACK 机制，但是注意一点的是，并非对于每一个报文段都进行确认，而仅仅对最后一个报文段确认，捎带的确认了上图中的 203 号及以前的报文。

1. TCP超时重传，能否设计出发送后超时重传，观察计时时间，重传报文段与第一次发送的报文段时间差。（知识点：TCP可靠传输-停等协议）

由于下层网络层（IP）可能出现丢失、重复或失序包的情况，TCP协议提供可靠数据传输服务。为保证数据传输的正确性，TCP重传其认为已丢失的包。TCP根据接收端返回发送端的一系列信息来判断是否出现丢包。当数据段丢失或确认信息丢失，TCP启动重传操作，重传尚未确认的数据。TCP拥有两套操作来确认重传，一是基于时间的，而是基于确认信息的构成。

一、超时重传

  超时重传是指，发送数据包在一定的时间周期内没有收到相应的ACK，超过设定的等待时间后就认为这个数据包丢失，就会重新发送，这个等待时间称为重传超时时间（RTO，Retransmission TimeOut）。

1.重传超时时间

  RTO（Retransmission TimeOut）即重传超时时间。TCP超时与重传中一个很重要的部分是对一个给定的连接的往返时间（RTT）的测量。由于网络流量的变化，这个时间会相应地发生变化，TCP需要跟踪这些变化并动态调整超时时间RTO。

  RTT（Round Trip Time）由三部分组成：链路的传播时间、末端系统的处理时间、路由器缓存中的排队和处理时间。其中，前两个部分对于一个TCP连接相对固定，路由器缓存中的排队和处理时间会随着整个网络的拥塞程度的变化而变化。所以RTT的变化在一定程度上反映了网络的拥塞程度。

  TCP超时和重传的基础是怎样根据给定连接的RTT设置RTO。RTO的值被设置过大过小都会对协议造成不良影响。如果RTO设置过大将会使发送端经过较长时间的等待才能发现报文段丢失，降低了连接数据传输的吞吐量；RTO设置过小，则会造成不必要的重传，浪费网络资源。

2.怎样根据RTT设置RTO

  对一个连接而言，如能够了解端点间的传输往返时间（RTT，Round Trip Time），则可根据RTT来设置一个合适的RTO。显然，在任何时刻连接的RTT都是随机的，无法事先预知。TCP通过测量来获得当前RTT的一个估计值，并以该RTT估计值为基准来设置当前的RTO。

RTO=srtt+4\*sttvar

rttvar ：平均偏差

srtt：平滑的RTT估计值

3.重传二义性与Karn算法（RTT的精确估值测量）

  当接收到重复传输（即至少重传一次）数据的确认信息时，不进行该数据包的RTT测量，可以避免二义性问题。另外，对该数据之后的包采取退避策略，仅当接收到未经重传的数据时，该SRTT才用于计算RTO。

二、快速重传

  在超时重传中，定时器超时后才会认为发送的数据包丢失了。快速重传机制实现了另外的一种丢包评定标准，即如果连续收到三次ACK，发送方就认为这个包丢失了，需要重新传递。这个机制不需要等到重传计时器溢出，所以叫做快速重传。它可以避免发送端因等待重传计数器的超时而空闲较长时间，以此增加网络吞吐量。而重新传递以后，因为走的不是慢启动而是拥塞避免算法，所以这又叫做快速恢复算法。

1. TCP连接释放，能否捕获完成双向释放请求和确认，注意观察主机中协议端口状态。（知识点：TCP连接管理）

首先，先断开连接的一方（即数据先传送完毕，以下称为“A”）,会向另一方（以下称为 “B”）发送TCP断开连接包，连接释放报文段的首部FIN置“1”，序号为前面已传送过的数据的最后一个字节的序号加“1”(本次实验为“327”)；B收到连接报文段后发出确认，确认号为A发的序号+1（本次实验为“328”），这样，A到B这个方向的连接就释放了，TCP就处于半关闭状态，但B可以发送数据，直到B发送TCP连接断开包，B向A发送连接断开包的过程与A向B发送的TCP连接断开包过程类似。

1. 不同高层协议在运输层的反映，如http、QQ、FTP等应用。（知识点：TCP、UDP协议及应用层协议）

2，网络可以提供两种服务：  
1）可靠，面向连接；（TCP） 就像靠谱的快递，每一步都有反馈和监控，当然价格也是呵呵...  
2）不可靠，尽力而为的传输 （UDP） 就像某些不靠谱的快递或者听都没听过的XX快递，价格低，但是能不能到就靠运气了。  
3，两种服务无所谓好坏，TCP 的可靠是需要消耗很多资源的，效率低 （大块，重要的文件等）  
UDP 不保证可靠性，但是效率高（视频，语音，不重要的小文件等）

TCP：传输控制协议，面向连接的的协议，稳定可靠。当客户和服务器彼此交换数据前，必须先在双方之间建立一个TCP连接，之后才能传输数据。  
 UDP：广播式数据传输，UDP不提供可靠性，它只是把应用程序传给IP层的数据报发送出去，但是并不能保证它们能到达目的地。由于UDP在传输数据报前不用在客户和服务器之间建立一个连接，且没有超时重发等机制，故而传输速度很快。 QQ等应用就是使用的UDP协议。

优点：1.传输数据为字节级，传输数据可自定义，数据量小。相应的移动端开发，手机费用低

   2.传输数据时间短，性能高

   3.适合C/S之间信息实时交互

   4.可以加密，数据安全性高

缺点： 1.需要对传输的数据进行解析，转化为应用级的数据

    2.对开发人员的开发水平要求高

    3.相对于Http协议传输，增加了开发量

Http请求主要有http协议，基于http协议的soap协议，常见的http数据请求方式有get和post，web服务

优点：1.基于应用级的接口使用方便

    2.要求的开发水平不高，容错性强

缺点：1.传输速度慢，数据包大。

    2.如实现实时交互，服务器性能压力大

    3.数据传输安全性差