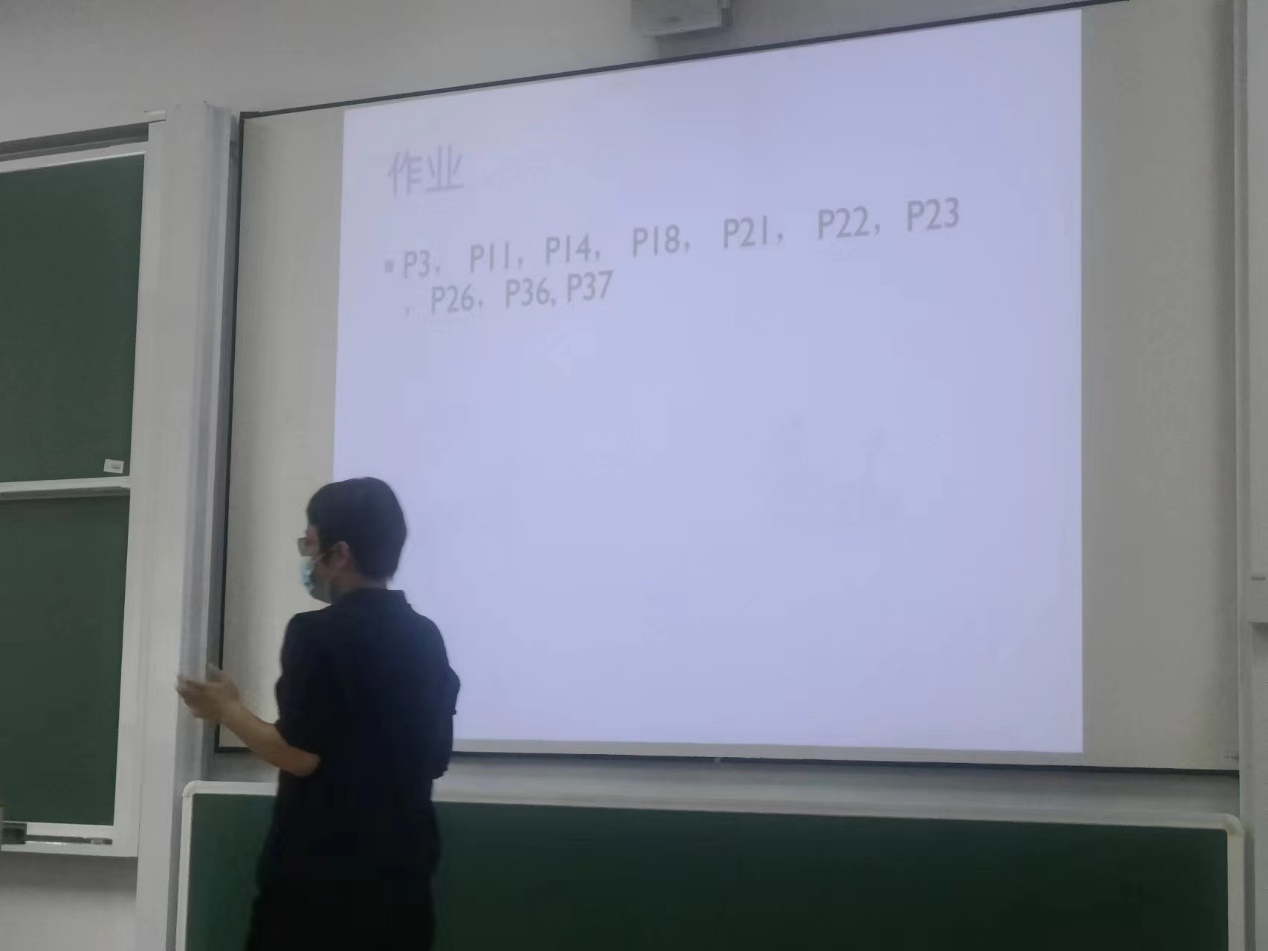
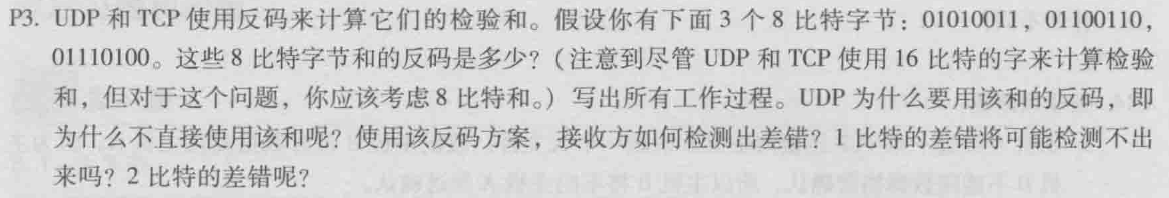
计算机网络第三章理论习题

--10215501435 杨茜雅





答：

01010011 + 01100110 = 10111001

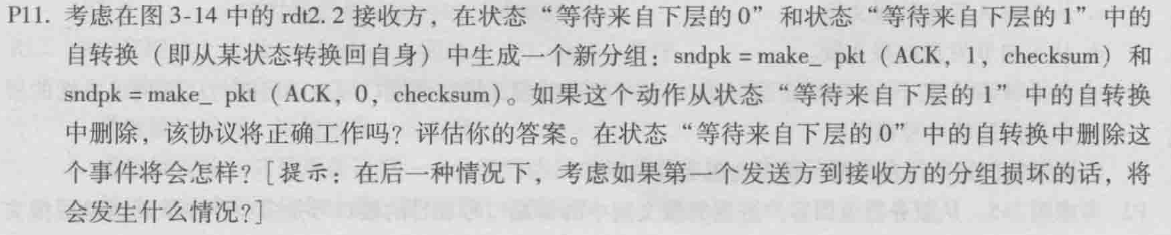
10111001 + 01110100 = 00101110 在溢出时应该向最低位进位

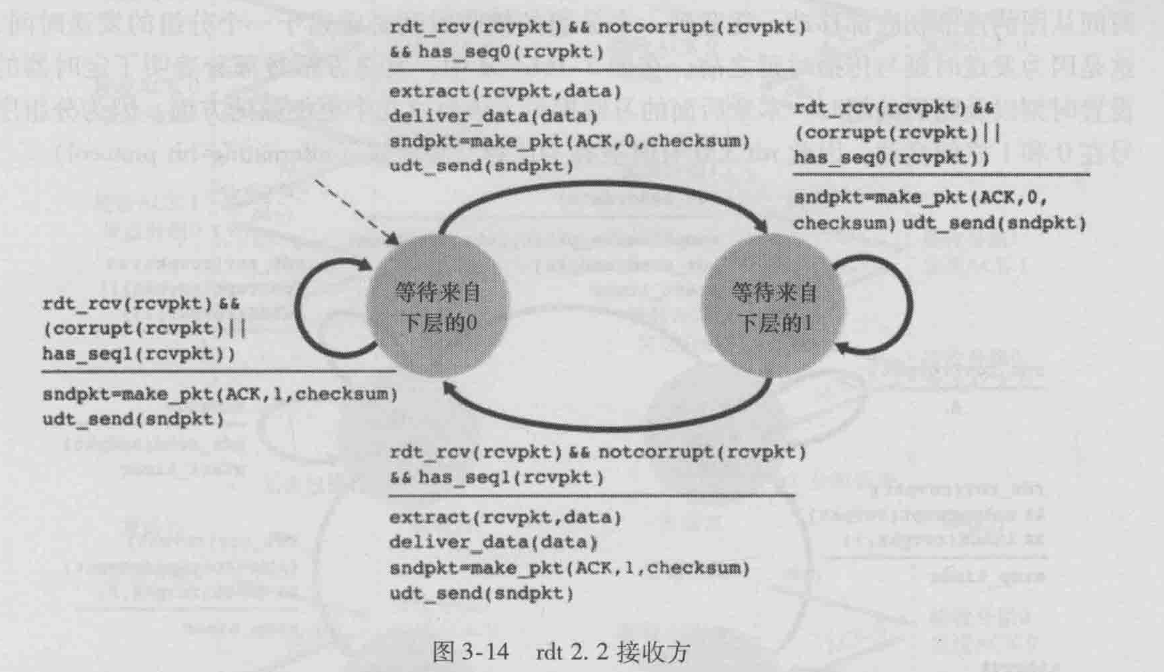
则反码为11010001

使用反码，不必依赖系统是大端还是小端，只需将所有数据包含校验码加起来，计算和为全1 即可，所以计算检验和比较简单快速。

差错检验方法：将三个字节与检验和相加，如果任何一个位为0，说明出错。

1 比特的差错一定能检测出，肯定会导致结果不同。2 比特的差错可能检测不出。比如题中第一、二字节变为01010010，01100111，即最后一个比特反转。

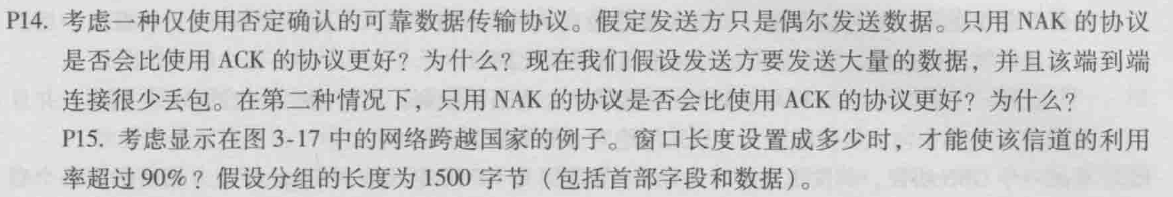


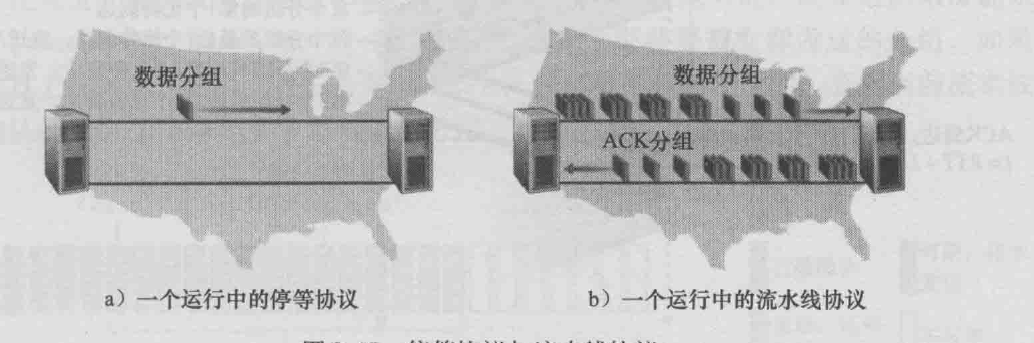


答：

前一种情况：如果从“等待来自下层的1”中删除，不会影响正常工作，因为sndpkt 已经被生成了。

后一种情况：但是如果从“等待来自下层的0”中删除，而且接收方刚刚启动（处于初始状态），sndpkt 是一个错误的值（很可能是一个随机值），那么发送方会认为ACK 损坏并重发分组，接收方会继续发送错误值，这将导致一个死锁。

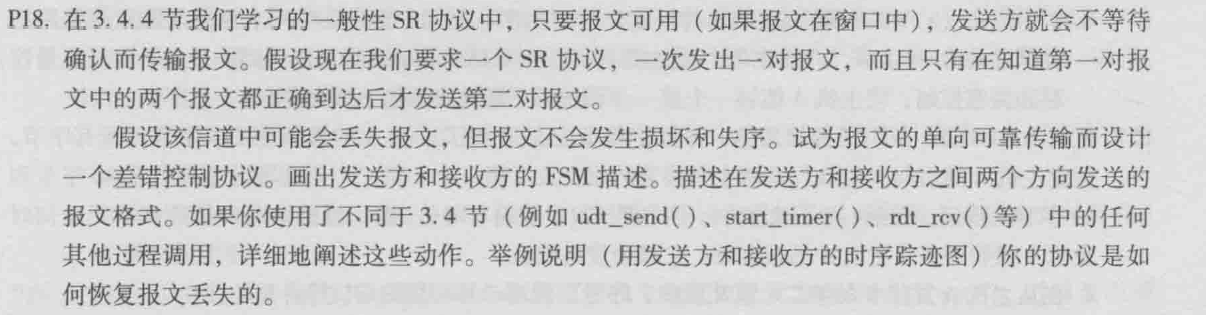




答：

只用NAK 协议不会比使用ACK 协议更好。因为接收方判断丢失是依据数据包的上下文，且只是偶尔发送数据，那么只有当丢失的包下一个包被接收时才会发现丢包，这个时间会更长，因此偶尔发送数据，只用NAK协议不会比使用ACK 协议更好。

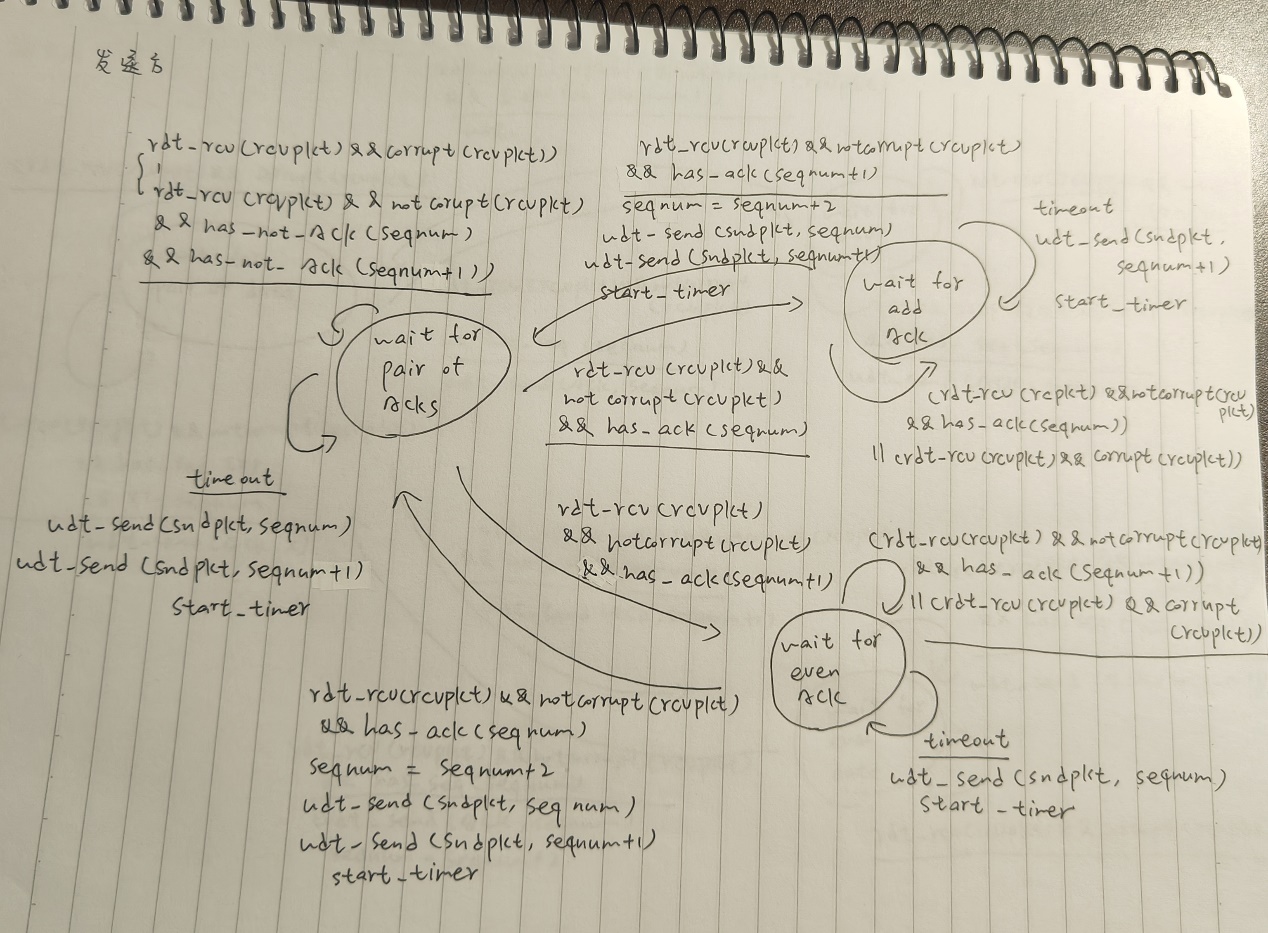
而当数据量较大且很少丢包时，用NAK 协议发送的数据包的数量明显比ACK 协议少，所以用NAK 协议会比使用ACK 协议更好，因为这样能够减小数据流量。

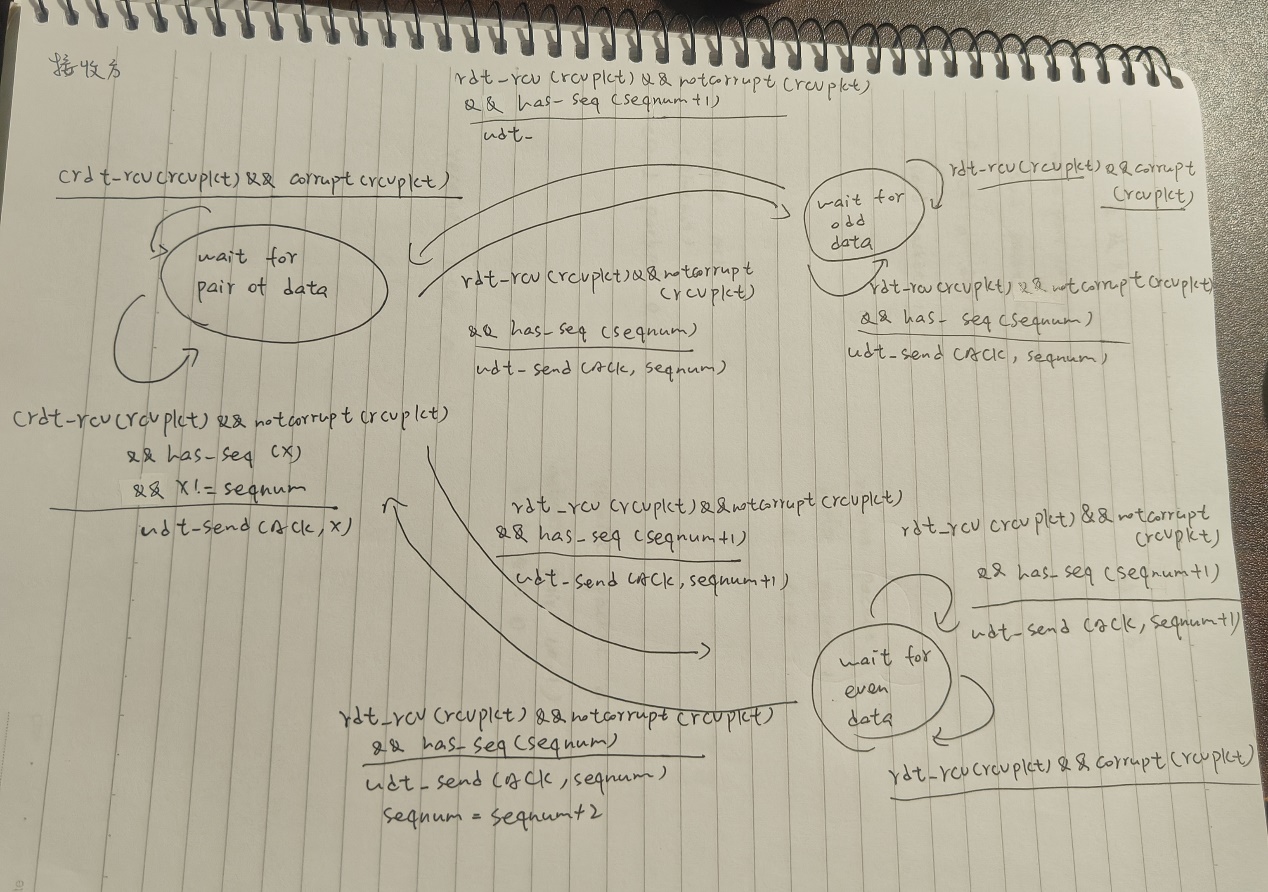


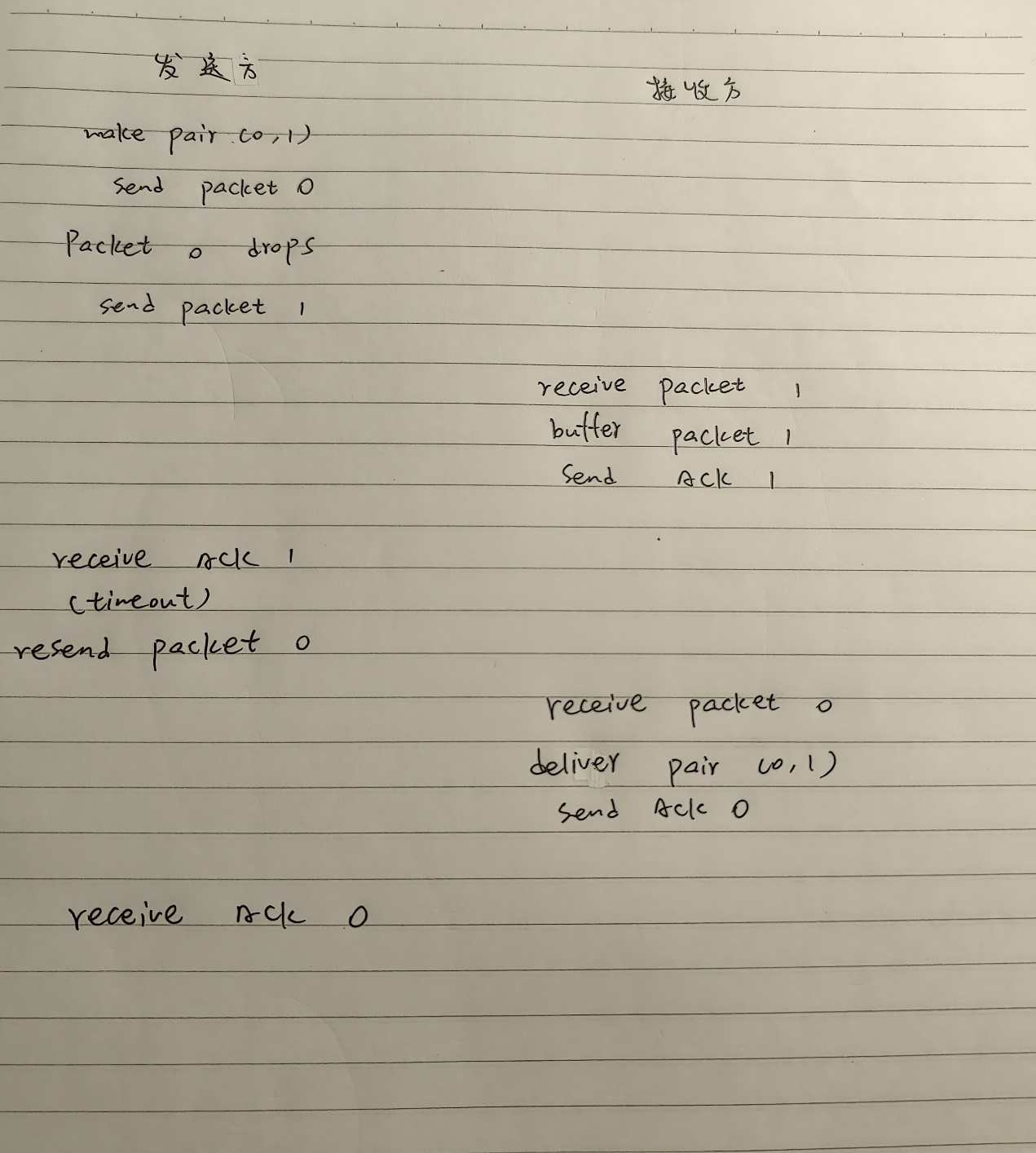
答：

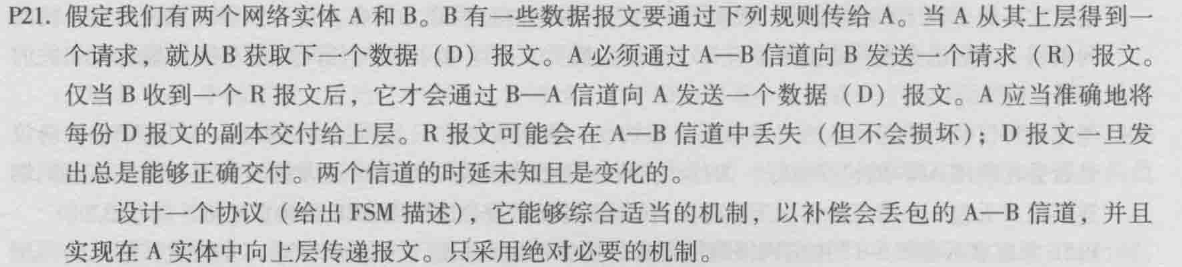
发送方将等待，直到它收到seqnum和seqnum+1，然后再进入下一对消息。有效的序列号是0、1、2和3。

发送方状态记录的是：当前一对数据包是否没有收到ACK， 是否有ACK？假设seqnum最初是 0，并且发送方已经发送了前两个数据信息。



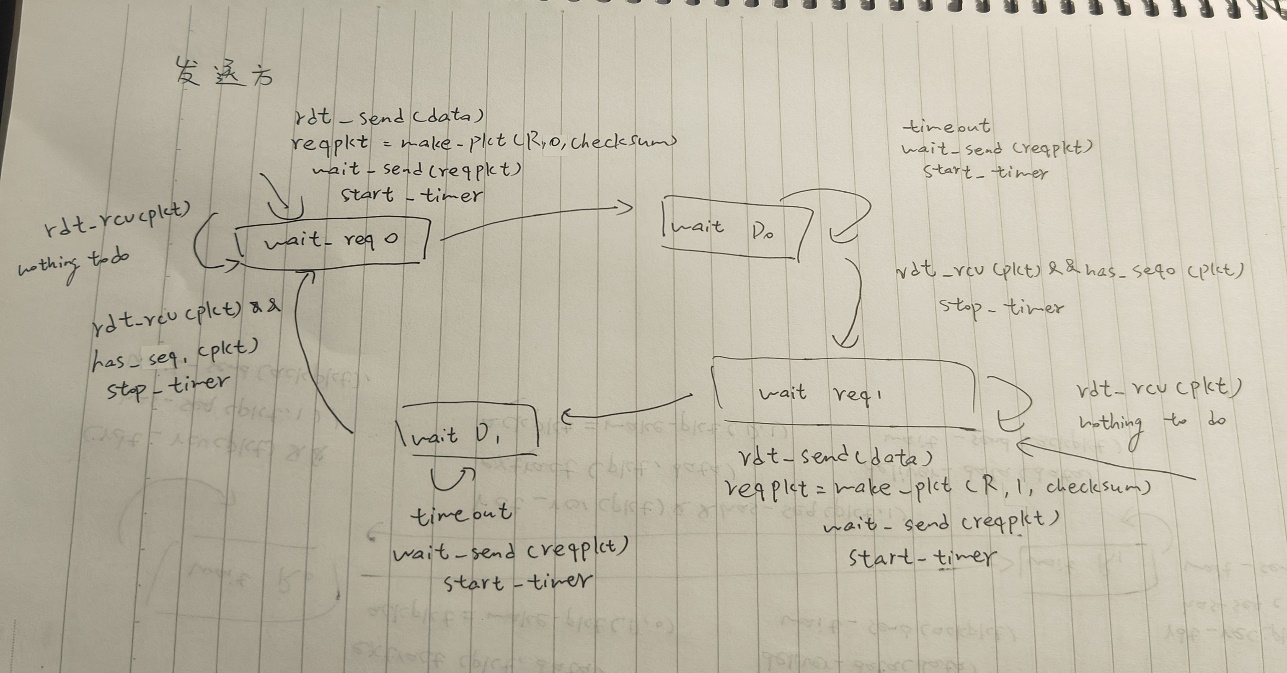




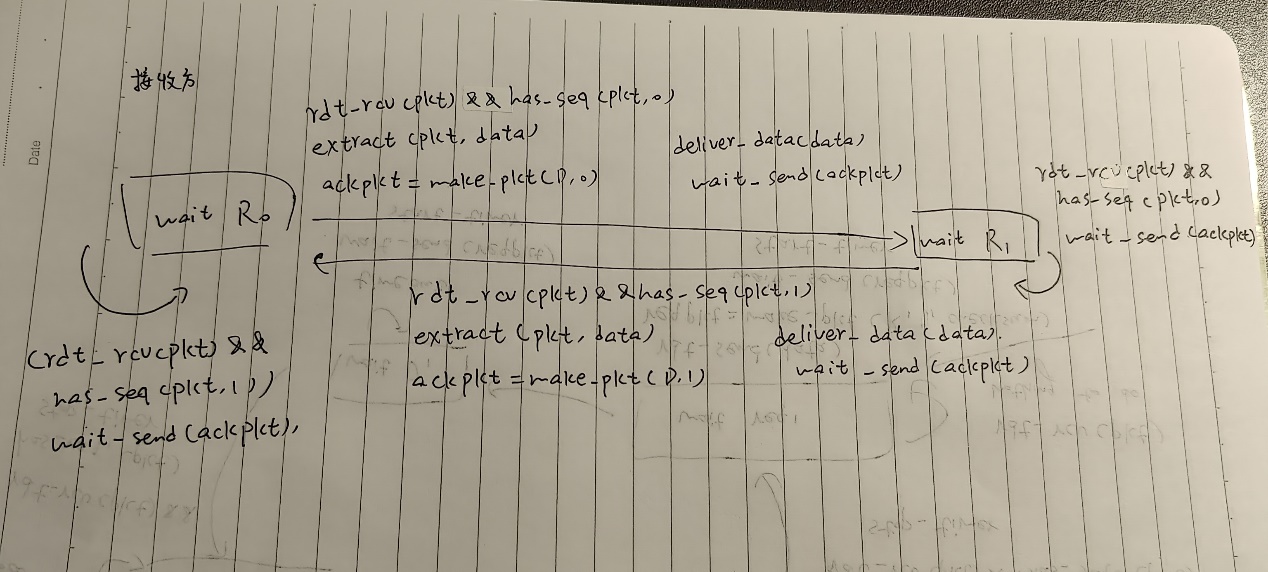


答：

发送方：



接收方：



由于A-to-B 通道可能会丢失请求消息，因此A 需要超时并重新传输其请求消息（以便能够从丢失中恢复）。由于信道延迟是可变的和未知的，因此A 可能会发送重复的请求（即，重新发送B 已经接收到的请求消息）。为了能够检测重复的请求消息，协议将使用序列号。1 位序列号足以满足停止和等待类型的请求/响应协议。

A（请求者）有4 种状态：

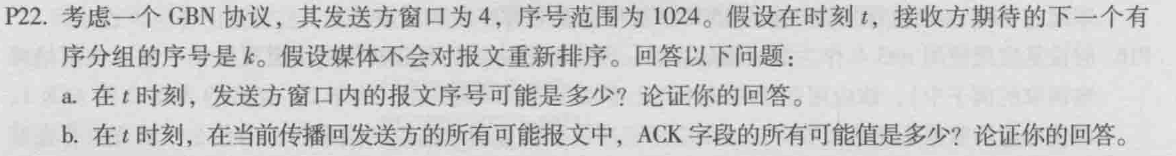
“ 等待来自上面的请求0。”在这里，请求者正在等待来自上面的调用以请求一个数据单元。当它接收到来自上面的请求时，它向B 发送一条请求消息R0，启动一个计时器并转换到“等待D0”状态。当处于“等待来自上面的请求0”状态时，A 将忽略从B 接收到的任何内容。“ 等待D0”。在这里，请求者正在等待来自B 的D0 数据消息。计时器始终在此状态下运行。如果计时器过期，A 将发送另一条R0 消息，重新启动计时器并保持此状态。如果从B 接收到D0消息，a 将停止时间并过渡到“等待来自上述的请求1”状态。如果A 在此状态下收到D1 数据消息，则会忽略该消息。

“ 等待来自上面的请求1。”在这里，请求者再次等待来自上面的调用以请求一个数据单元。当它接收到来自上面的请求时，它向B 发送一条请求消息R1，启动一个计时器并转换到“等待D1”状态。当处于“等待来自上面的请求1”状态时，A 忽略从B 接收到的任何内容。

“ 等待D1”。在这里，请求者正在等待来自B 的D1 数据消息。计时器始终在此状态下运行。如果计时器过期，A 将发送另一条R1 消息，重新启动计时器并保持此状态。如果从B 接收到D1消息，a 将停止计时器并过渡到“等待来自上面的请求0”状态。如果在该状态下接收到D0 数据消息，则忽略该消息。

数据供应商（B）只有两种状态：

“发送D0。”在此状态下，B 通过发送D0 继续响应接收到的R0 消息，然后保持在此状态。如果B 收到R1 消息，则它知道其D0 消息已正确接收。因此，它将丢弃该D0 数据（因为它已在另一端接收到），然后转换到“发送D1”状态，在该状态下，它将使用D1 发送下一条请求的数据。“发送D1。”在此状态下，B 通过发送D1 继续响应接收到的R1 消息，然后保持此状态。如果B 接收到R1 消息，则它知道其D1 消息已被正确接收，并因此过渡到“发送D1”状态。



答：

a.

考虑两种极端情况：

发送方发送k-4，k-3，k-2，k-1，接收方都完整得接收并发送ACK，但ACK 全都未传到发送方，接收方的期待序号为k，而发送方窗口序号为[k-4, k-1]

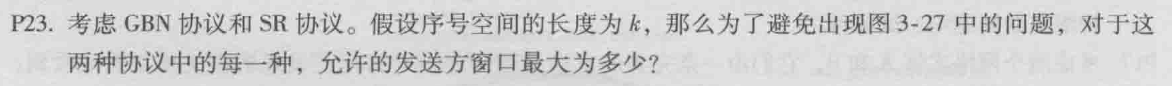
如果ACK 全都传回，则发送方更新base，其序号为[k, k+3]

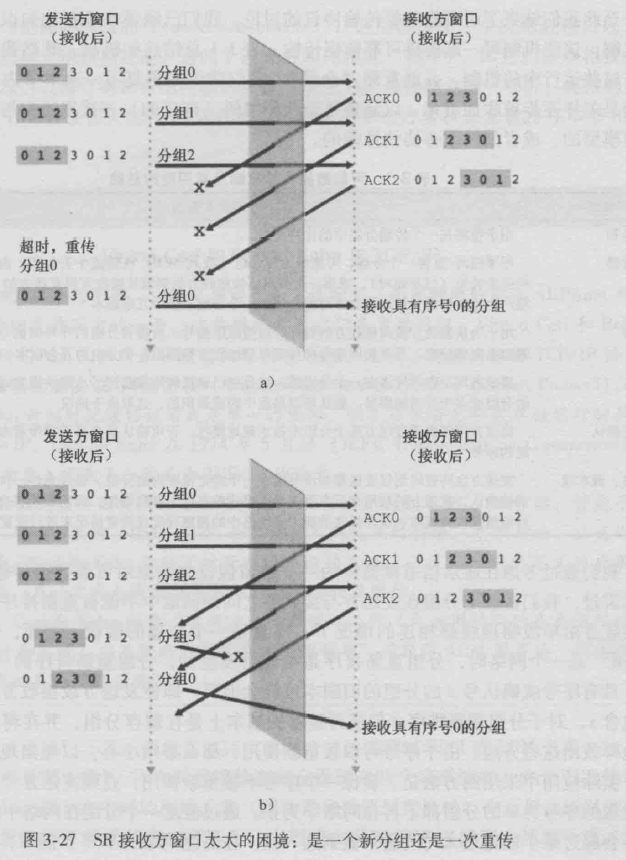
因此序号可能是[k-4, k+3]

b.

如果接收方期待k，则它一定将比k-1 小的ACK 发送出去了，如果要使发送方发送k-1，那么它至少已经接收到了k-5 的ACK。

因此正在传播回发送方的ACK 序号可能是[k-4, k-1]





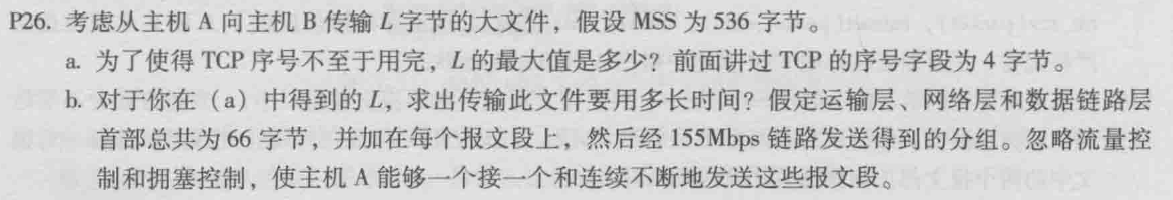
答：

设序号为0（第一个0），1，…，k-1，0（第二个0）

**SR协议**：要使其序号发生混杂，至少是当接收方刚刚包含第二个0，即接收方窗口为[k-N-1, 0]，也就是说k-N (包括k-N) 之前的都接收过了。要使序号混杂还有一个条件就是第一个0 在发送方窗口且恰好其ACK 丢失，需要重发。0(第一个0)~k-N 为k-N+1 个值，如果窗口长度不足k-N+1，则第一个0 和第二个0 不会同时包含在发送或接收窗口中。

所以窗口长度N <= k-N，即N <= k/2,即允许的发送方窗口大小最大为k/2。

**GBN协议**： 类似，N <= k 即可，即允许的发送方窗口大小最大为k。



答：

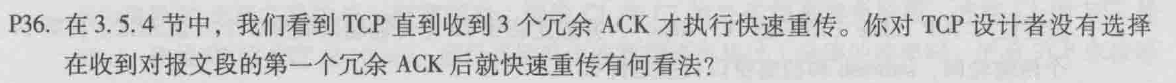
1. TCP是字节流编号的，L 的最大值是 byte。
2. 设N为报文数：

N = ·= 8012999

总头部长 = N\* 66 byte = 528857934 byte

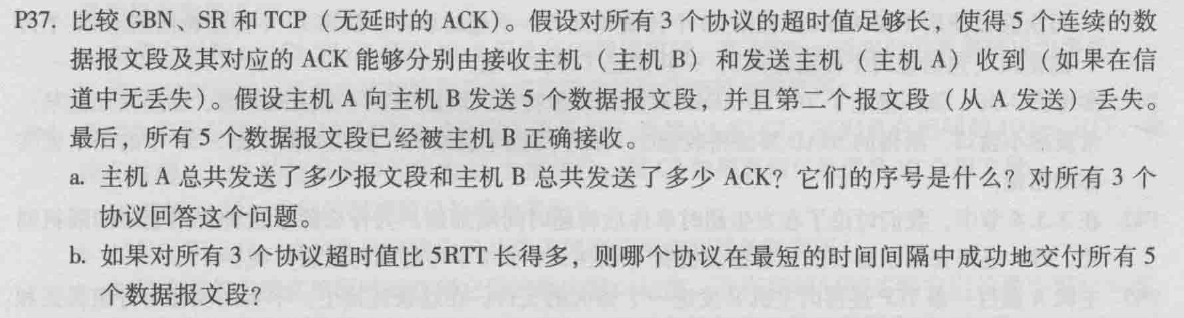
总字节数 = byte + 528857934 byte = 4.824 \* byte

t = =249s



答：

如果到达顺序错误，例如分组1正确到达，目的地发送ACK，分组3提前于分组2到达，发送一个冗余ACK，立即快速重传分组2，但实际上分组2并没有丢失，这没有必要。



答：

**a:**

GBN：

A：首先发送分组12345，后来重发2345，总共9个分组

B：首先发送ACK 1111，后来发送2345，总共5个ACK

SR：

A：首先发送分组12345，后来重发2，总共6个分组

B：首先发送ACK 1345，后来发送2，总共5个ACK

TCP：

A：首先发送分组12345，后来重发2，总共6个分组

B：首先发送ACK 2222，后来发送6，总共5个ACK

**b:**

GBN、SR 需要等待超时，而TCP使用快速重传，故TCP最快