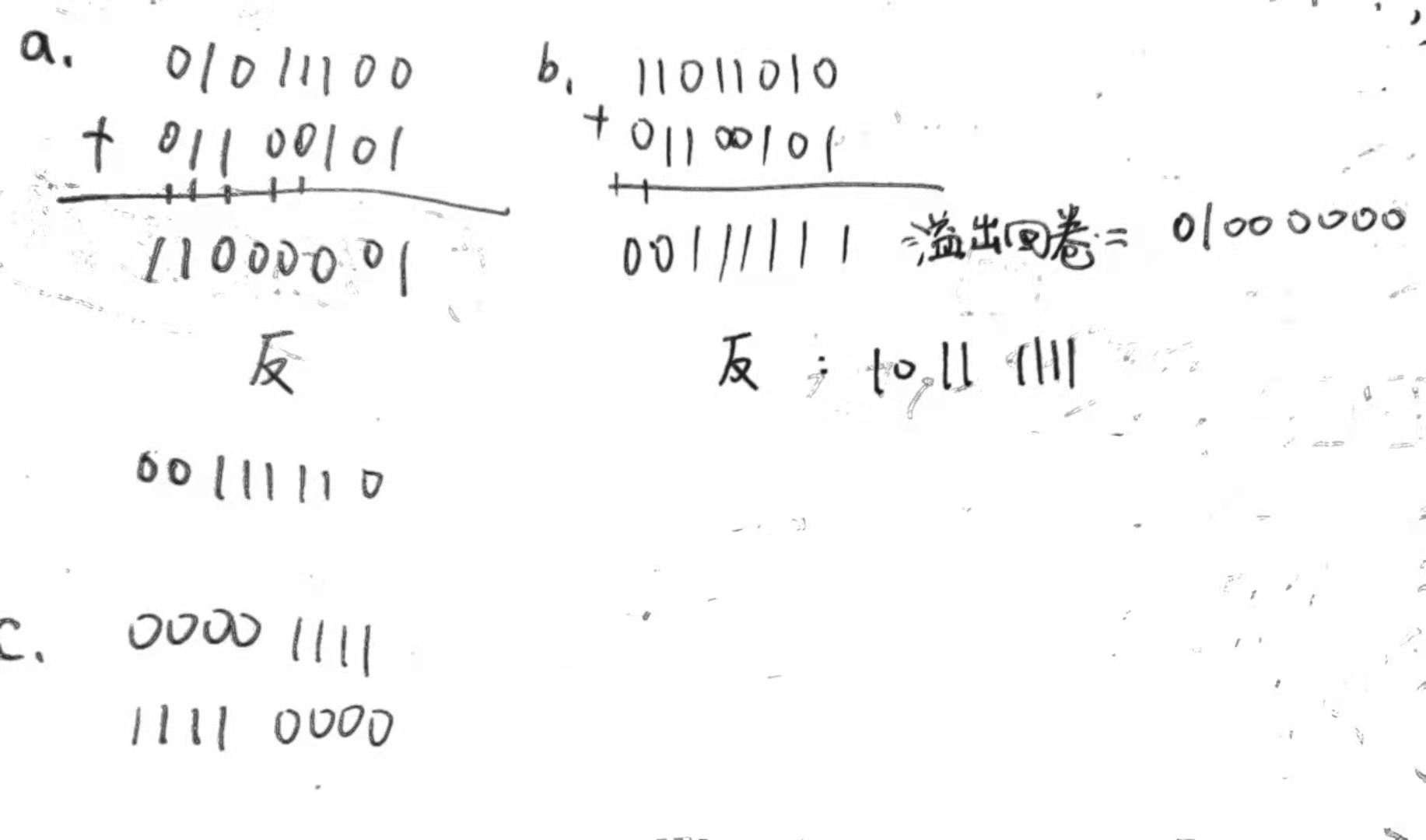
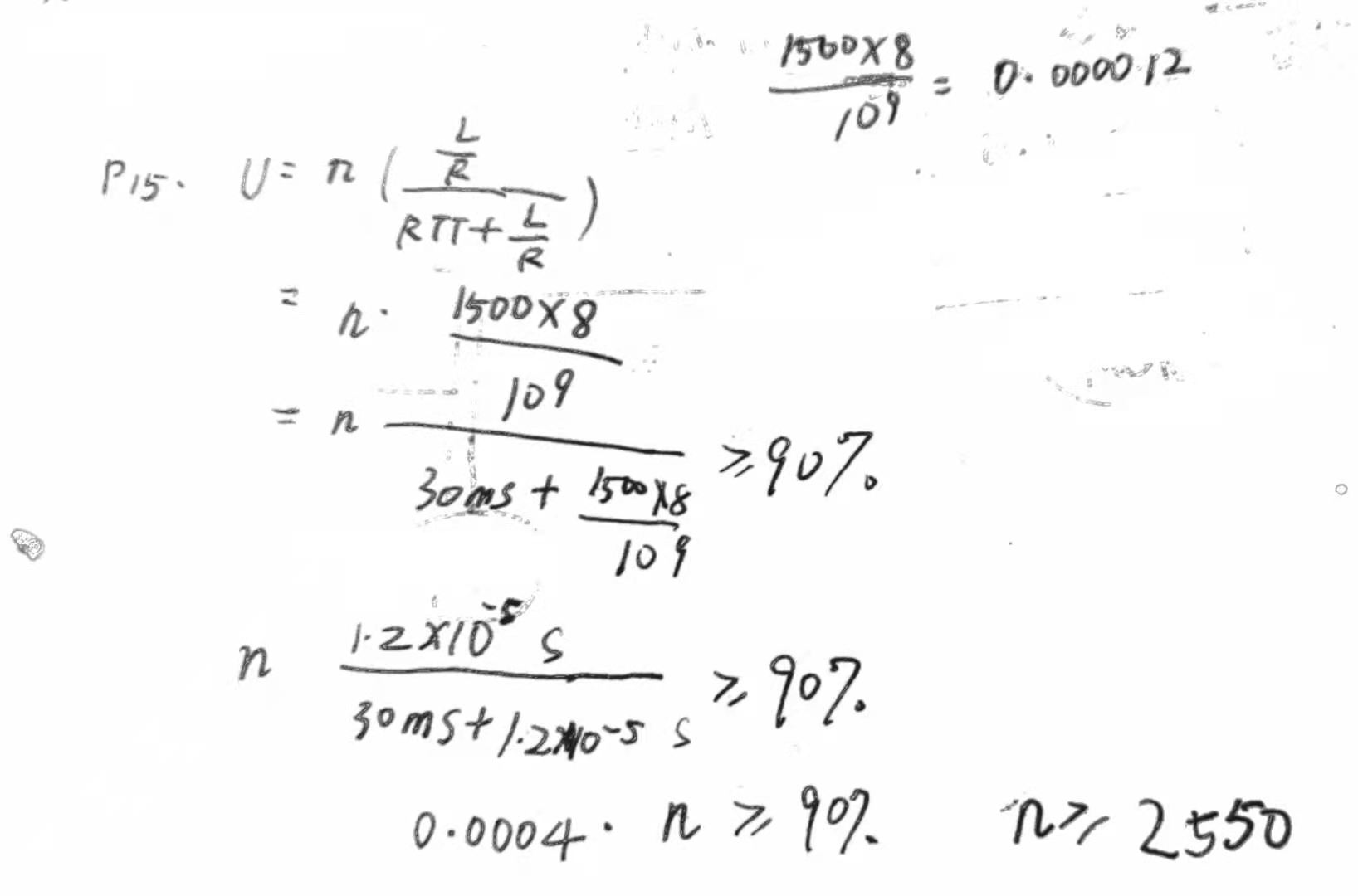
1a:rdt1.0->rdt2.0（新增了ARQ协议，包括差错检测（接收方的ACK NAK应答）、接收方反馈、重传，）->rdt2.1（针对接收方的ACK/NAK失效问题，引入了新的机制：序号。发送方在每个分组中加入序号，接收方通过序号判断是否重复接收同样的包，再进行下一次发送）->rdt2.2（功能同rdt2.1，但只使用编号的ACK，接收方对最后正确接收的分组发ACK以替代NAK）->rdt3.0（实现基于时间的重传机制，新增一个倒计数定时器）

1b:使用SR机制。发送方的发送窗口大小为4，从编号为0的组开始依次发送，其中0号和1号都顺利按顺序到达接收方，接收方窗口向前移动并向发送方发送ACK，但是第二个分组发送失败，接收方没有收到2号分组。与此同时发送方发送完3号分组后，发送窗口满了，需要等待来自接收方返回的0号、1号分组的ACK才能继续向前移动。收到ACK后，发送窗口向前移动，发送3、4、5号分组。而接收方没有收到2号分组，因此后面再接收到3、4、5号分组时，只是在接收窗口内部将它们缓存下来，后沿并不向前移动。因为发送方没有收到2号分组的ACK，所以一直等待到对2号分组内置的计时器超时，发送方重传2号分组，此时接收方成功收到二号分组并将其缓存，联合之前收到并缓存的3、4、5号分组，排序后按序交付上层。

P14：不会，从结果上来说，如果仅仅使用NAK会比ACK更好，现在学的应该是这种协议；如果分析一下的话，发送方偶尔发送数据，说明数据发送得不多，这种时候对发送窗口没什么流量限制，而还是要等到每个下一次发送才收到本次发送的NAK，永远差一个，效率上来说没必要；而如果发送大量数据，并且很少丢包，那由于是可靠数据传输，每次发送方都要等RTT，确认没有NAK应答才会发下一个包，会使发送速率降低。

P4

P15 

P24 a 是，因为接收方不知道该分组是否已经被接受过，仅仅会根据接收到的分组把对应分组的ACK传回去，所以如果发送方出问题的话，比如重置了发送窗口，是有可能的。

b 是，和a一样。

C 是， SR协议是特殊的比特交替协议，它的发送和接收窗口都扩大了，但仍然是每次只针对一个分组的协议。

d 是，GBN协议是特殊的比特交替协议，它的发送窗口都扩大了，但接收方的ACK仍然是每次只针对最大的一个分组的回应。

P26

