homework 4

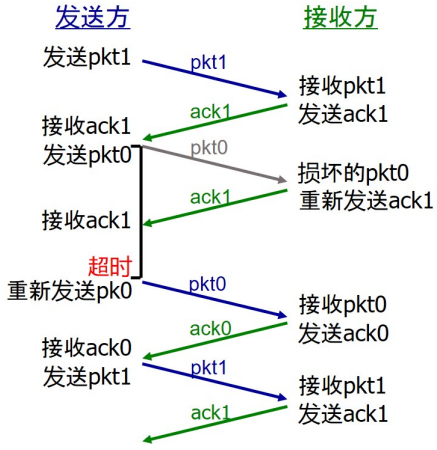
P1. 设主机A和B的socket端口号分别为$port_A$、$port_B$，则
a. 从A向S发送的报文段源端口号$port_A$，目的端口号23
b. 从B向S发送的报文段源端口号$port_B$，目的端口号23
c. 从S向A发送的报文段源端口号23，目的端口号$port_A$
d. 从S向B发送的报文段源端口号23，目的端口号$port_B$
e. 如果A和B是不同的主机，端口号可能相同。
f. 如果A和B是同一台主机，端口号不可能相同。

P3.
(a) 由于UDP反码计算中的溢出需要回卷，可得$01010011 + 01100110 = 10111001$，$10111001 + 01110100 = 00101110$，所以所求和的反码为11010001。
(b) 使用反码计算UDP校验和可使计算不受到机器的大小端属性影响，也可以提高性能。
(c) 在接收方，所有4个(8比特)字(包括检验和)加在一起。 如果该分组中没有引入差错，则在接收方处该和的每一位都会是1。如果这些比特之一是0，那么该分组中已出现了差错。
(d) 1比特的差错一定会被检测出来；2比特的差错不一定能检测出来(比如，如果两个字的最后一位对换了)。

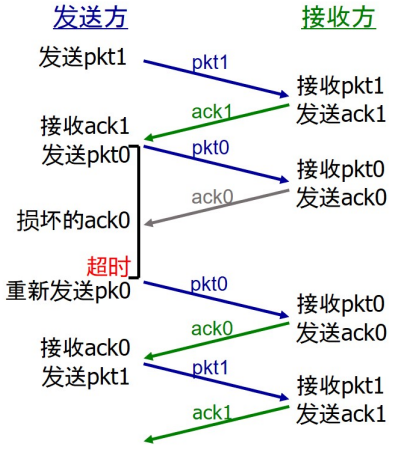
P6. 假设接收方收到了序号为0的未损坏的packet并向发送方回复`ACK`。但发送方收到的`ACK`是损坏的，于是当接收方等待来自下层的1时，收到了序号为0的未损坏的packet，则接收方会向发送方发送`NAK`的packet。而发送方收到`NAK`就会继续发送上述序号为0的未损坏的packet，导致接收方继续发送`NAK`的packet。为了进入下一状态，发送方期待接收方回复`ACK`，而接收方期待发送方发送序列号为1的packet，从而陷入死循环。

P7. 由于我们假设在rdt3.0的语境中不会出现(`ACK`) packet乱序的情况，所以发送方在处理接收方发来的`ACK`时仅需要了解发来的`ACK`是否与上一个`ACK`重复。由于这些`ACK` packet中包括了它们正在确认的packet序号，通过这些序号即可判断是否有重复，因此这些`ACK` packet不再需要自己的序号了。

P9.



被篡改的数据分组



被篡改的ACK确认分组

P10.

(a) 修改后的rdt2.1协议有一个计时器，其最大时间设定为已知的信道最大时延。在发送方的`Wait for ACK or NAK0`和`Wait for ACK or NAK1`状态上，我们各增加一个`timeout`事件。`timeout`事件发生后，发送方自转换，即`udt_send(sndpkt)`重传。

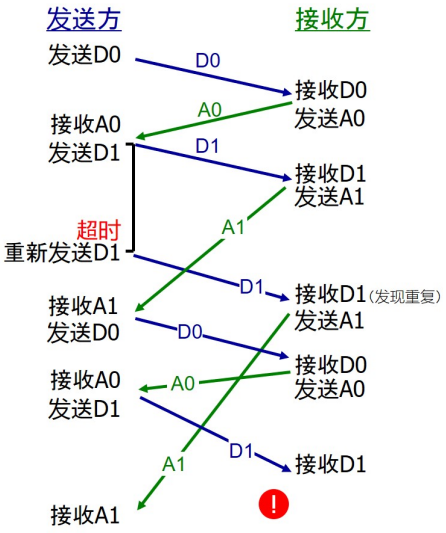
(b) 我们的新机制保证了当数据分组丢失时，由于接收方不会对丢失的分组作出反应，当发送方超时后就会重新发送当前分组；当确认分组丢失时，由于发送方无法接收到确认分组，超时后也会重新发送当前分组，而rdt2.1协议已经可以处理重复的数据分组。因此，我们的协议可以通过该信道正确通信。

P11.

(a) 如果仅仅去掉等待来自下层的1中的`sndpkt = mk_pkt(ACK, 1, checksum)`这一动作，由于在进入等待来自下层的1状态前一定会`sndpkt = mk_pkt(ACK, 1, checksum)`，因此不会影响这个协议的正确工作。

(b) 如果去掉等待来自下层的0中的发送`sndpkt`事件，那么如果发送方给接收方的`pkt0`是损坏的，接收方将不会有反应，会期待发送方发送序号为0的未损坏分组；而这时发送方会期待接收方作出回应，因此会陷入死锁状态，这个协议将无法正常工作。

P13.



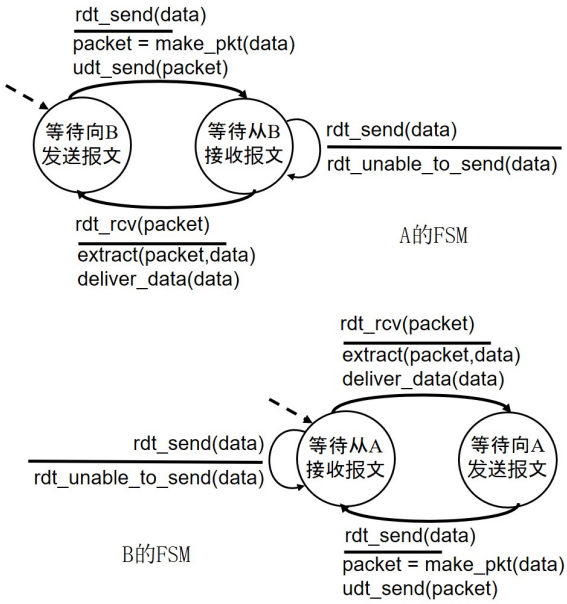
P14. 对一个只用NAK的协议来说，如果一个分组丢失，那么接收方要到接收下一个分组时才能发现有分组丢失了。

(a) 由于发送方仅仅是偶尔发送数据，发现分组丢失的时间间隔将很漫长，这意味着低下的传送效率。因此，在这种情况下，只用NAK的协议不如使用ACK的协议。

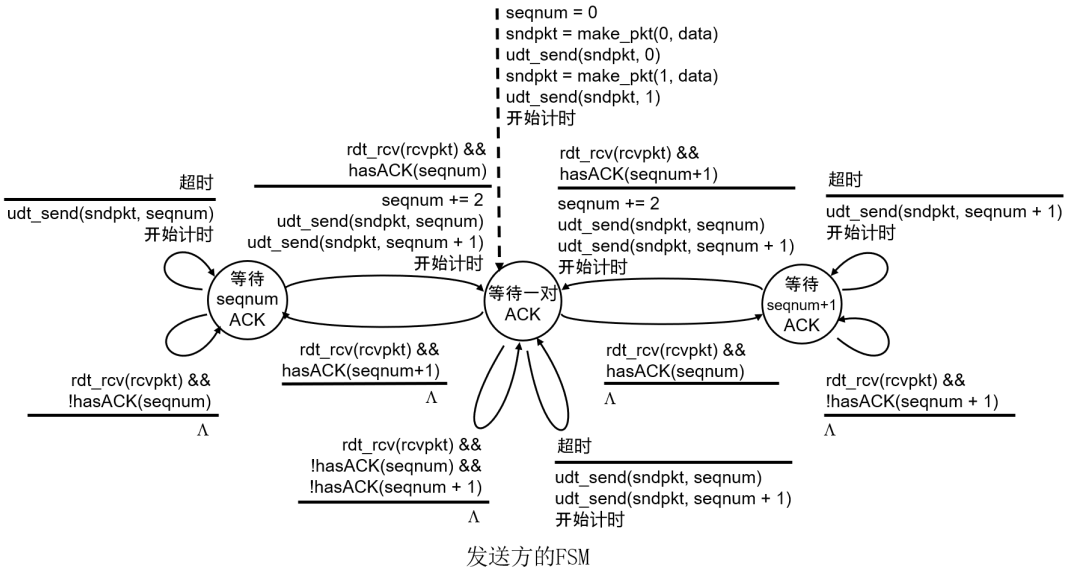
(b) 由于发送方要发送大量数据，且这一连接的丢包率很低，所以很快就能发现分组丢失，需要发送的NAK也较少，传送效率很高。在这种情况下，只用NAK的协议会比使用ACK的协议更好。

P15. 因为$U_{sender} = \dfrac{nL/R}{RTT + L/R} = \dfrac{.012n}{30 + .012} = 90\%$，解得窗口长度$n = 2250.9$。

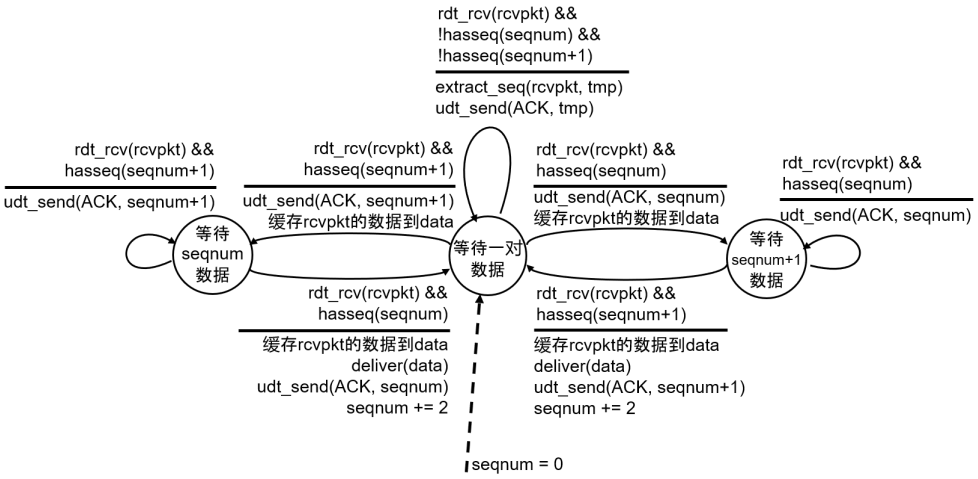
P17.



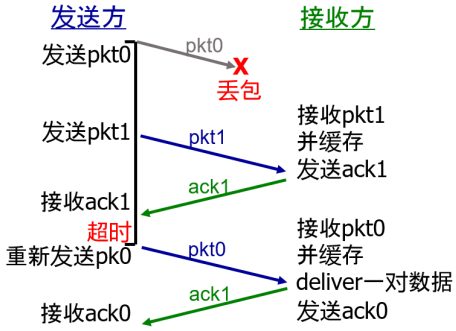
P18. 这一协议的报文包含一般的数据字段和一个(至少)2比特的序号字段。在下列FSM图中，`hasACK(x)`指的是判断收到的报文是否包含序号为x的ACK；`extract_seq(pkt)`指的是从`pkt`中获取其序号。



发送方的FSM



接收方的FSM



P22.

a. 考察两种边界情况：如果发送方已经收到接收方对$(k-1)$分组的确认，那么现在对发送方来说窗口中有序号$[k, k+N-1]$，即$[k, k+3]$；如果发送方尚未收到接收方对$(k-N), \ldots, (k-1)$分组的确认，那么现在对发送方来说窗口中有序号$[k-N, k-1]$，即$[k-4, k-1]$。因此，窗口中的序号是$[i, i+3]$，其中$i = k-4, \ldots, k$。如果考虑序号范围，对答案每一项取除以1024的余数即可。

b. 同样考察两种边界情况：如果发送方已收到接收方对$(k-1)$分组的确认，那么并无正在传播的ACK；如果发送方尚未收到接收方对$(k-N), \ldots, (k-1)$分组的确认，那么至多有$[k-N-1, k-1]$这些分组在传播。因此，ACK字段的所有可能值是$[k-4, k-1]$。类似a问，如果考虑序号范围，对答案每一项取除以1024的余数即可。