段的网络层数据报中，IP 地址是多少？

P3. UDP 和 TCP 使用反码来计算它们的检验和。假设你有下面 3 个 8 比特字节：01010011，01100110，01110100。这些 8 比特字节和的反码是多少？（注意到尽管 UDP 和 TCP 使用 16 比特的字来计算检验和，但对于这个问题，你应该考虑 8 比特和。）写出所有工作过程。UDP 为什么要用该和的反码，即为什么不直接使用该和呢？使用该反码方案，接收方如何检测出差错？1 比特的差错将可能检测不出来吗？2 比特的差错呢？

P4. a. 假定你有下列 2 个字节：01011100 和 01100101。这 2 个字节之和的反码是什么？

b. 假定你有下列 2 个字节：11011010 和 01100101。这 2 个字节之和的反码是什么？

c. 对于（a）中的字节，给出一个例子，使得这 2 个字节中的每一个都在一个比特反转时，其反码不会改变。

P5. 假定某 UDP 接收方对接收到的 UDP 报文段计算因特网检验和，并发现它与承载在检验和字段中的值相匹配。该接收方能够绝对确信没有出现过比特差错吗？试解释之。

P6. 考虑我们改正协议 rdt2.1 的动机。试说明如图 3-57 所示的接收方与如图 3-11 所示的发送方运行时，接收方可能会引起发送方和接收方进入死锁状态，即双方都在等待不可能发生的事件。

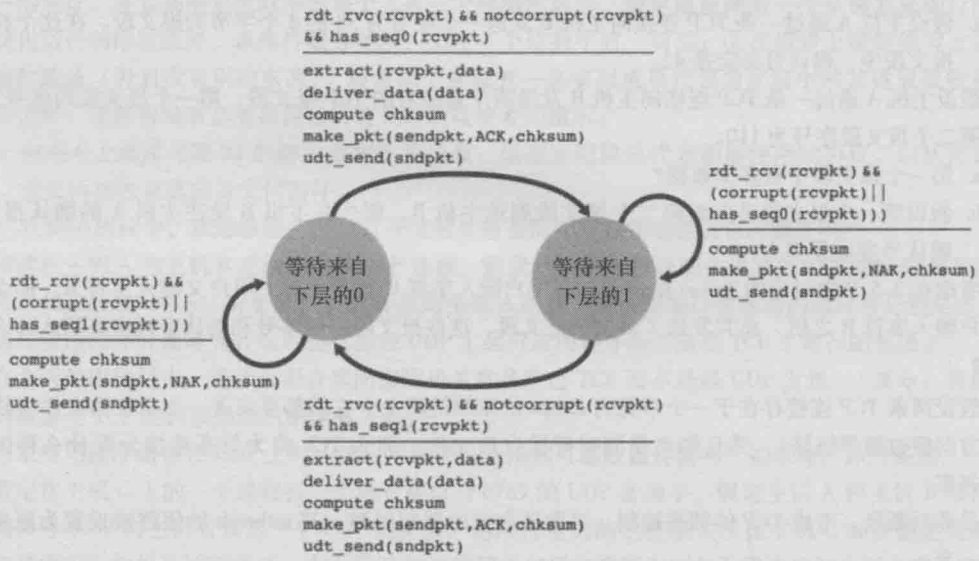

图 3-57 协议 rdt2.1 的一个不正确的接收方

P7. 在 rdt3.0 协议中，从接收方向发送方流动的 ACK 分组没有序号（尽管它们具有 ACK 字段，该字段包括了它们正在确认的分组的序号）。为什么这些 ACK 分组不需要序号呢？

P8. 画出协议 rdt3.0 中接收方的 FSM。

P9. 当数据分组和确认分组发生篡改时，给出 rdt3.0 协议运行的轨迹。你画的轨迹应当类似于图 3-16 中所用的图。

P10. 考虑一个能够丢失分组但其最大时延已知的信道。修改协议 rdt2.1，以包括发送方超时和重传机制。非正式地论证：为什么你的协议能够通过该信道正确通信？

P11. 考虑在图 3-14 中的 rdt2.2 接收方，在状态“等待来自下层的 0”和状态“等待来自下层的 1”中的自转换（即从某状态转换回自身）中生成一个新分组：sndpk = make_ pkt（ACK，1，checksum）和 sndpk = make_ pkt（ACK，0，checksum）。如果这个动作从状态“等待来自下层的 1”中的自转换中删除，该协议将正确工作吗？评估你的答案。在状态“等待来自下层的 0”中的自转换中删除这个事件将会怎样？[提示：在后一种情况下，考虑如果第一个发送方到接收方的分组损坏的话，将会发生什么情况？]