第三章第2题

主机 A 与主机 B 建立了一条 TCP 连接，用于可靠地传输数据。假设主机 A 要向主机 B 发送一个由 10 个报文段（Segment）组成的数据流，每个报文段包含 1000 字节的数据。已知初始序列号（ISS）为 5000，接收方主机 B 的初始接收窗口为 8000 字节。在传输过程中，忽略拥塞控制的影响。

（1）请写出第 1、第 5 和第 10 个报文段的数据所覆盖的字节序列号范围。

（2）当主机 B 成功接收并缓存了第 1 至第 3 个报文段，以及第 5 个报文段后（假设第 4 个报文段丢失），它发送出的下一个确认号（ACK）应该是多少？这个确认号的含义是什么？

（3）如果主机 B 的应用程序之后读取了 3000 字节的数据，主机 B 的接收窗口会发生什么变化？它如何通过报文告知主机 A 这一变化？

（4）主机 A 发送了第 1 个报文段（Seq=5000）。请描述在以下两种情况下，主机 A 的行为：

①该报文段在传输过程中丢失。

②该报文段成功到达主机 B，但主机 B 返回的确认报文（ACK=6000）丢失。

（5）以上两种情况的最终结果有何异同？这体现了 TCP 协议的什么特性？

（6）在初始状态下，主机 A 的可用发送窗口（Send Window）有多大？

（7）假设主机 A 已连续发送了 5 个报文段（Seq=5000, 6000, 7000, 8000, 9000），但只收到了主机 B 对前两个报文段的确认（ACK=7000）。同时，主机 B 在 ACK=7000 的这个报文里通告其接收窗口（rwnd）变为 6000 字节。请画出此时主机 A 的发送窗口示意图，并标出已确认、已发送未确认、可发送、不可发送的数据范围。

（8）根据（7）的条件，主机 A 接下来最多还能连续发送多少字节的数据？

（9）TCP 并非通过在传输前复制多份数据副本来实现可靠传输。请简要阐述 TCP 是如何主要依靠序列号、确认和重传这三个基本机制来共同保障数据可靠传输的。

（10）假设一个极其特殊的环境：网络传输非常可靠，几乎从不丢包，但延迟非常高（例如深空通信）。在这种环境下，TCP 的超时重传机制可能会面临什么挑战？能否提出一个简单的优化思路？（提示：从 RTO 的设置角度思考）

1

- 5000 ～ 5999
- 9000 ～ 9999
- 15000 ～ 15999

2

应该是 8000； ACK 是“期望收到的下一个字节的序列号”， 又由于 TCP 是累积确认，所以应该是 8000

3

- 这意味着缓存区**释放了3000字节的空间**。所以，新的接收窗口 rwnd 会在原来的基础上增加3000。
- 主机B会在它发送给主机A的下一个TCP报文段的首部中，填入最新的接收窗口大小（rwnd值）。主机A收到后，就会相应地调整自己的发送窗口。

4

Case 1

-

- 主机A在发送第1个报文段后，会启动一个**超时计时器 (RTO)**。
- 由于报文段丢失，主机B收不到任何东西，也不会发送ACK。
- 主机A在等待一段时间后，**计时器超时**。
- 主机A会认为报文段丢失，并**重传**第1个报文段（Seq=5000），然后重置计时器。

Case 2

-
- 主机A同样在发送后启动了超时计时器。
- 主机B收到了报文段，并发送了ACK=6000，但这个ACK在路上丢了。
- 主机A**没有收到**预期的确认，在等待一段时间后，**计时器同样会超时**。
- 主机A**无法区分**是自己发的包丢了，还是对方回的ACK丢了。从它的角度看，结果都是一样的——没有收到确认。
- 主机A重传第1个报文段（Seq=5000），并重置计时器。

5

两种情况的最终结果**相同**，这体现了TCP协议的**可靠性**

6

8000

7

- **已确认**: (-∞, 6999]
- **已发送未确认**: [7000, 9999]
- **可发送，未发送**: [10000, 12999] (可用窗口)
- **不可发送**: [13000, +∞)

8

3000

9

1. **序列号** : 发送方为每个字节打上唯一的编号，这为“确认”和“重排序”提供了基础。接收方可以根据序列号拼接乱序到达的数据段，并识别重复的数据。

2. **确认** : 接收方通过返回ACK号，明确告知发送方“我已经成功收到了哪些数据，接下来我需要哪个”，从而为发送方提供了**肯定的反馈**。
3. **重传** : 这是保障可靠性的最后一道防线。当发送方在规定时间内**没有收到**预期的“肯定反馈”（ACK）时，该机制（超时重传或快重传）被触发。它假定发生了丢包，并重新发送数据，直到收到确认为止。

- **共同作用**: 这三者形成了一个“**发送-确认-超时/重传**”的循环。序列号是识别身份的ID，确认是成功的信号，重传是失败的补救措施。它们共同确保了每一个字节的数据最终都能、且仅一次地被正确交付。

10

- 主要问题在于 **无法精确估算重传超时时间 (RTO)**。
    - 容易出现 过早超时导致的伪重传 ，过晚超时导致的错误恢复极慢
- 解决方案：
    - 引入一种轻量级的、独立于数据重传的探测机制来更准确地评估网络状态，并调整重传策略。
    - 发一些小包来作为一个类似 “Ping” 的过程，判断连接是否存活
    - 如果总是可以收到探测包，说明连接存活，可能只是延迟高，否则如何探测包多次丢包，连接可能似了，需要启动重传。