华东师范大学软件学院课程作业

课程名称：计算机网络	年级：2023 级本科	姓名：张梓卫
作业主题：第六章作业	学号：10235101526	作业日期：2024/12/31
指导老师：刘献忠	组号：	

一 6.1

题目

In our example transport primitives of Fig. 6-2, LISTEN is a blocking call. Is this strictly necessary? If not, explain how a nonblocking primitive could be used. What advantage would this have over the scheme described in the text?

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

Figure 6-2. The primitives for a simple transport service.

解答

LISTEN 作为阻塞调用并非严格必要。事实上，LISTEN 可以被设计为非阻塞调用。在非阻塞方案中，LISTEN 调用仅表示建立新连接的意愿，并立即返回。当有连接尝试时，程序可以通过信号通知或回调机制处理连接请求。例如，可以执行“OK”或“REJECT”操作来接受或拒绝连接。

Advantages — .1

- **提高效率**：允许程序在等待连接时继续执行其他任务，例如处理日志或管理其他网络活动，从而更高效地利用系统资源。
- **增强灵活性**：开发者可以灵活设计程序逻辑，而不受阻塞机制的限制。
- **避免死锁风险**：在某些复杂的多线程或多进程环境中，非阻塞操作可以减少死锁或资源竞争的风险。

Warning — .2

- **实现复杂度增加**：非阻塞模式需要额外的轮询或事件通知机制，增加了开发和维护的复杂性。
- **潜在性能问题**：轮询可能导致系统开销增加，特别是在高负载场景中。如果使用事件驱动机制，则需要支持底层事件通知接口。

非阻塞 LISTEN 原语适用于需要同时处理多项任务的复杂应用程序，能够提高系统的并发能力和灵活性。

若是对于简单的单任务程序，阻塞 LISTEN 可能更容易实现和维护。

二 6.2

题目

In the underlying model of Fig. 6-4, it is assumed that packets may be lost by the network layer and thus must be individually acknowledged. Suppose that the network layer is 100 percent reliable and never loses packets. What changes, if any, are needed to Fig. 6-4?

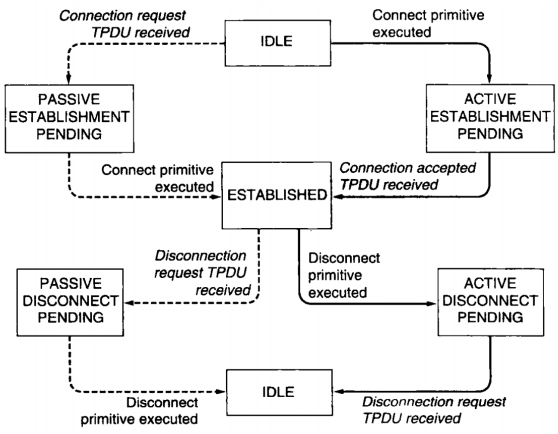

Figure 6-4. A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.

解答

如果网络层是 100%可靠且从不丢失分组，那么在图 6-4 中，不再需要对每个分组进行单独确认。可以简化协议，省略确认机制，或者使用更高效的确认方法，如累积确认。

设计可以做如下调整：

Tips 二 .1

- 取消对每个分组的单独确认：在网络层 100%可靠的情况下，发送方可以假设分组一定会被成功送达，无需为每个分组单独发送确认信息。可以简化协议，直接省略确认机制。
- 直接建立连接状态：由于网络层可靠性保证，连接可以直接从 'PASSIVE ESTABLISHMENT PENDING'状态跳转到 'ESTABLISHED'状态，因为可以确认对方肯定能收到消息。无需等待额外的确认即可建立连接。
- 保留断开连接的 PENDING 状态：虽然网络层可靠，但在断开连接时，收到客户端的断开请求只意味着不会再接收来自客户端的数据包。此时，服务器仍可能有未发送完的数据包需要发给客户端，因此在断开连接时仍需要保持 'PENDING'状态以完成数据传输。
- 使用累积确认或滑动窗口机制：如果仍需提高传输效率，可以使用累积确认来一次确认多个分组的接收情况，或者使用滑动窗口机制，在等待 ACK 之前允许发送多个分组，提高整体传输效率。

三 6.3

题目

Why does the maximum packet lifetime, T, have to be large enough to ensure that not only the packet but also its acknowledgements have vanished?

解答

为了理解这个问题，首先需要明确三次握手过程是如何解决延迟或重复引发的问题。

- 主机 1 发送连接请求时，选择一个序列号 x，并向主机 2 发送一个包含该序列号的连接请求 TPDU；
- 主机 2 收到请求后，回应该连接请求的 TPDU，确认 x，并声明自己选用的初始序列号 y；
- 最后，主机 1 在其发送的第一个数据 TPDU 中确认主机 2 所选择的初始序列号 y。

Notice

当出现延迟的重复控制 TPDU 时，例如，一个来自于已关闭连接的延迟重复连接请求（*Connection Request*）TPDU 可能在主机 1 毫不知情的情况下被主机 2 接收到，这会引发问题：

- 主机 2 会向主机 1 发送一个接受连接的 TPDU（*Connection Accepted*）来响应该连接请求；
- 然而，这个“接受连接”的 TPDU 的真正目的是主机 2 确认建立一个新的连接，而不是响应旧的连接请求。

关键在于主机 2 需要使用一个从主机 2 到主机 1 交互的初始序列号 y，这样可以确保该序列号所对应的旧 TPDU 或 ACK 已经不再存在于网络中。否则，可能会导致主机 1 对延迟的 TPDU 产生误解，从而试图建立新的连接。

最糟糕的情况是延迟的“连接请求”与“连接接受”的确认响应都在网络中存活。如果一个重发的分组到达，并且包含一个旧的序列号时：

- 如果在网络中还有旧的分组或 ACK，可能会混淆当前的连接过程，甚至破坏三次握手的正常工作。
- 错误的解析可能会导致故障性连接，从而引发难以定位的问题。

因此，最大分组寿命 T 必须足够大，以确保不仅数据分组本身已经从网络中消失，其对应的确认分组（ACK）也已经完全消失。这是为了避免由于分组或确认分组的延迟导致的混淆，例如新分组与旧确认分组的混淆，或者导致重复确认的错误解释。只有这样，才能确保协议的正确性，避免因旧分组或确认分组的存在而引发的传输错误，从而保证三次握手及连接管理的可靠性。

四 6.4

题目

Discuss the advantages and disadvantages of credits versus sliding window protocols.
讨论信用与滑动窗口协议的优缺点。

解答

滑动窗口协议

Advantages 四 .1

优点：滑动窗口协议实现较为简单，只需要管理窗口边界的一组参数（即窗口起始位置和结束位置），无需复杂的额外参数。不仅如此，其可靠性也较强，具体体现在可以连续发送多个分组，同时确保分组按序到达，避免了信用机制中窗口增加和减少时，可能出现分组乱序的问题。

Warning 四 .2

缺点：滑动窗口协议的灵活性较差，因为窗口大小的调整较为固定，无法根据接收方的具体缓冲区状态动态管理。在某些情况下，窗口资源可能未被充分利用，或导致传输效率降低。

信用机制协议

Advantages 四 .3

优点：信用机制允许动态管理缓冲区的使用。接收方可以根据自身缓冲区的状况，为发送方分配适当的信用额度，避免发送方发送过多数据导致接收方缓冲区溢出。信用机制将缓冲区的动态管理与分组确认机制分离开，使得协议更加灵活。

Warning 四 .4

缺点：信用机制需要额外维护和管理信用额度，这增加了协议的实现复杂性。还有一点是，信用额度信息的传递可能需要额外的通信开销。

五 6.5

题目

Why does UDP exist? Would it not have been enough to just let user processes send raw IP packets?

解答

仅仅让用户进程发送原始 IP 数据包当然远远不够。

IP 数据包仅包含目标计算机的 IP 地址，无法识别具体的目标进程。

UDP 的存在不仅解决了 IP 分组无法区分目标进程的问题，还通过引入端口号和提供简单的传输层接口，使开发者能够专注于应用逻辑，而无需处理底层网络层的复杂性。相比直接发送 IP 分组，UDP 提升了安全性、灵活性和标准化管理能力。

Notice

UDP 提供一种简单、无连接的传输层协议，允许应用程序在不建立连接、无需保证可靠性或顺序的情况下发送数据报。相比让用户进程直接发送原始 IP 分组，UDP 简化了传输过程，减少了协议开销，并提供了必要的接口和功能，使应用程序能够专注于自身逻辑而无需处理传输层的复杂性。此外，直接发送原始 IP 分组可能带来安全风险，且缺乏标准化的错误处理和流量控制机制，而 UDP 则通过标准化的接口和功能提供了更好的安全性和可管理性。

六 6.6

题目

Suppose that the TCP congestion window is set to 18 KB and a timeout occurs. How big will the window be if the next four transmission bursts are all successful? Assume that the maximum segment size is 1 KB.

解答

在 TCP 协议中，当发生超时时，拥塞窗口（cwnd）会被重置为 1 个最大报文段（MSS），并进入慢启动（Slow Start）阶段。此时，cwnd 会从 1 KB 开始，逐渐按指数增长。假设 MSS 为 1 KB，具体过程如下：

- cwnd 被重置为 1 KB。
- 阈值（ssthresh）被重置为 18 KB/2 = 9 KB。

解答 六 .1

慢启动过程

- **第 1 次传输成功**：cwnd 增加到 2 KB（发送 1 个分组）。
- **第 2 次传输成功**：cwnd 增加到 4 KB（发送 2 个分组）。

- **第 3 次传输成功**：cwnd 增加到 8 KB（发送 4 个分组）。
- **第 4 次传输成功**：cwnd 理论上增加到 16 KB（发送 8 个分组）。

阈值限制

- 由于阈值为 9 KB，当 cwnd 达到或超过阈值后，增长进入拥塞避免阶段，在这种情况下，cwnd 不能超过阈值。

因此，在经过四次成功的传输后，拥塞窗口的大小将增长至 9 KB（受到阈值的限制）。

七 6.7

题目

A TCP machine is sending full windows of 65,535 bytes over a 1-Gbps channel that has a 10-msec one-way delay. What is the maximum throughput achievable? What is the line efficiency?

解答

解答 七 .1

TCP 窗口大小为 65,535 字节。
最大吞吐量可以通过窗口大小除以 RTT 来计算：

$$\text{吞吐量} = \frac{\text{窗口大小}}{\text{RTT}} = \frac{65,535\,\text{字节}}{0.02\,\text{s}} = 3,276,750\,\text{字节/秒} \approx 26.21\,\text{Mbps}$$

$$\text{线路效率} = \frac{\text{最大吞吐量}}{\text{带宽}} = \frac{26.21\,\text{Mbps}}{1000\,\text{Mbps}} = 2.62\%$$

故最大吞吐量约为 26.21 Mbps，线路效率约为 2.62%。

八 6.8

题目

In a network whose max segment is 128 bytes, max segment lifetime is 30 sec, and has 8-bit sequence numbers, what is the maximum data rate per connection?

解答

解答 八 .1: 6.8

发送方最多可发送 255 个分段（由于序列号为 8 位，共有 256 个序列号，从 0 到 255），因此：

$$\text{最大数据量} = 255\,\text{段} \times 128\,\text{bytes/段} = 32,640\,\text{bytes}$$

分组的最大生命周期为 30 秒，因此最大数据速率为：

$$\text{最大数据速率} = \frac{32,640\,\text{bytes}}{30\,\text{s}} \approx 1,088\,\text{bytes/s}$$

换算为比特：

$$1,088\,\text{bytes/s} \times 8\,\text{bits/bytes} = 8,704\,\text{bits/s}$$

因此，每个连接的最大数据速率为 8,704 bps。

九 6.9

题目

A CPU executes instructions at the rate of 1000 MIPS. Data can be copied 64 bits at a time, with each word copied costing 10 instructions. If an incoming packet has to be copied four times, can this system handle a 1-Gbps line? For simplicity, assume that all instructions, even those instructions that read or write memory, run at the full 1000-MIPS rate.

解答

解答 九 .1

每次复制 64 bits（8 字节）数据需要 10 条指令，复制四次总共需要 40 条指令：

$$\text{每字节的指令数} = \frac{40}{8} = 5\,\text{指令/字节}$$

CPU 每秒执行 1000×10^6 条指令，处理数据速率为：

$$\text{数据速率} = \frac{1 \times 10^9\,\text{指令/秒}}{5\,\text{指令/字节}} = 200\,\text{MB/s} = 1600\,\text{Mbps}$$

目标数据速率为 1 Gbps，因此：

$$1600\,\text{Mbps} > 1000\,\text{Mbps}$$

结论：该系统可以处理 1 Gbps 的网络连接。

十 6.10

题目

To get around the problem of sequence numbers wrapping around while old packets still exist, one could use 64-bit sequence numbers. However, theoretically, an optical fiber can run at 75 Tbps. What maximum packet lifetime is required to make sure that future 75-Tbps networks do not have wraparound problems even with 64-bit sequence numbers? Assume that each byte has its own sequence number, as TCP does.

解答

解答 十 .1: 6.10

序列号空间的大小为 2^{64} 字节，这相当于约 2×10^{19} 字节。
假设网络以 75 Tbps 的速率传输，则每秒消耗的序列号数量为：

$$75\,\text{Tbps} = 75 \times 10^{12}\,\text{bits/second}$$

转换为字节速率：

$$\frac{75 \times 10^{12}}{8} = 9.375 \times 10^{12}\,\text{bytes/second}$$

传输完 2^{64} 字节所需的时间为：

$$T = \frac{2^{64}}{9.375 \times 10^{12}} \approx \frac{1.844 \times 10^{19}}{9.375 \times 10^{12}} \approx 1,962,653\,\text{seconds} \approx 22.7\,\text{days}$$

最大分组寿命应小于约 22.7 天，以确保不会出现序列号环绕问题。

总结：在未来 75 Tbps 的网络中，使用 64 位序列号可以有效避免序列号环绕问题，即使传输速率非常高。最大分组寿命的限制为约 22.7 天，这在实际中是一个非常宽裕的限制条件。