# 用自己的语言，总结传输层的作用、位置、主要功能和主要服务。

**作用**：主要作用是为运行在网络边缘的不同主机上的各应用程序之间提供通信服务。另一个主要作用是在应用层和网络层之间充当复用器。

**位置**：传输层位于网络层之上，在资源子网与通信子网之间。

**主要功能**:

①分段和重组数据: 将应用层传输的数据分割成适合网络传输的小块，同时在目的地重新组装这些数据。

②错误检测和恢复: 提供机制来检测和纠正在数据传输过程中出现的错误，确保数据的可靠性和完整性。

③流量控制: 管理数据的流动，避免目标设备因接收速度过快而无法处理，确保传输平稳进行。

④多路复用和多路分解: 通过端口号将不同的应用程序数据区分开来，确保多个应用程序能够共享网络连接。

**主要服务**:

①传输控制协议: 提供可靠的、面向连接的服务，确保数据可靠地按顺序到达目的地。TCP使用三次握手建立连接，实现数据的可靠传输和流量控制。

②用户数据报协议: 提供无连接的、不可靠的服务，主要用于实时应用和那些对数据丢失或乱序不敏感的应用程序。UDP更轻量级，不提供像TCP那样的可靠性和流量控制。

# 在传输层中，端口、传输层地址和套接字分别是指什么？相互之间有什么关系？

**端口**:是一个逻辑标识，用于区分同一设备上不同应用程序或服务之间的通信。在传输层中，端口分为两种类型：TCP端口和UDP端口。每种协议都有独立的端口号空间，例如，HTTP服务通常使用TCP端口80，而DNS服务使用UDP端口53。

**传输层地址**:在OSI/RM中，当两个不同主机的两个进程需要通信时，必须指明对方是哪一个进程，这个标记称为传输层地址，也称为传输服务访问点，即：传输层地址（TSAP）=主机IP地址+端口号

**套接字**:套接字是应用程序与传输层之间的接口，通过套接字应用程序可以与网络进行通信。套接字由IP地址和端口号的组合构成，它允许应用程序使用传输层协议（如TCP或UDP）在网络上发送和接收数据。

**端口、传输层地址、套接字相互之间的关系**:

①端口是传输层中用于标识不同应用程序或服务的数字标签。

②传输层地址由IP地址和端口号组成，用于唯一标识网络中的特定应用程序或服务。

③套接字是应用程序通过传输层协议与网络通信的接口，它由IP地址和端口号组成，允许应用程序发送和接收数据。

# 什么是UDP协议？有什么特点？

用户数据报协议（User Datagram Protocol，UDP）是OSI参考模型和TCP/IP模型中都有的一种面向无连接的传输层协议。

**特点**:

①快速性：由于UDP不需要建立连接和维护状态，它的开销较小，传输速度相对较快。这使得UDP适用于那些对速度要求较高、能容忍一定数据丢失的应用场景。

②无连接性：UDP是无连接的协议，不需要在发送数据之前先建立连接，也不需要在传输过程中维护连接状态。每个UDP数据包都是独立的，相互之间没有关联。

③不可靠性：UDP不提供数据可靠性保证。它不负责数据包的重新发送或确认，因此数据包可能会丢失或到达顺序可能会被打乱。这使得UDP不适合要求高可靠性的应用，但适用于实时性要求较高、对延迟更为敏感的应用。

④适用性：UDP常用于音频、视频流媒体、在线游戏等实时性要求高、数据量较小、对可靠性要求不高的应用场景。

⑤支持广播和多播：UDP可以支持向多个目标发送数据，包括广播（发送给网络中所有主机）和多播（发送给特定组的主机）。

# 在TCP报文段的首部中，你认为哪些字段非常重要？原因是什么？

①源端口号和目的端口号：这两个字段用于标识通信中的发送方和接收方，是实现端到端通信的关键。

②序列号：TCP报文段必须携带一个序列号，用于保证数据的顺序和完整性。序列号使得TCP能够将数据分段，并在接收端重新组装。序列号字段用于标识TCP报文段中第一个字节的序列号，用于对数据进行排序和重新组装。它确保数据可以按正确的顺序传输和组装，同时也用于实现流量控制和拥塞控制。

③确认号：确认号字段指示了期望收到的下一个字节的序列号。接收端使用确认号向发送端发送确认信号，表示已经成功接收到特定的数据段。这是实现可靠传输的关键。

④数据偏移量：该字段用于指示数据部分的开始位置，并帮助接收端确定数据段的边界。

⑤标志位（URG、ACK、PSH、RST、SYN、FIN）：这些标志位用于表示特定的TCP状态和行为。例如，URG标志指示数据紧急，PSH标志指示数据需要立即被推进到应用层，SYN标志用于建立连接，FIN标志用于关闭连接。

⑥窗口大小：窗口大小用于流量控制，它告诉接收端当前可接收的最大数据量。这有助于防止接收端因接收数据过快而无法处理的情况。

⑦校验和：这是一个错误检测机制，用于确保TCP报文段在传输过程中没有被篡改。

这些字段都是TCP报文段的重要组成部分，每个字段都有其特定的意义和功能，共同保证了TCP协议的可靠传输和高效率。

# 简述TCP连接建立和释放的基本过程

**TCP连接建立过程（三次握手）**:

第一步，客户端发送SYN：客户端向服务器提出连接建立请求，即发出同步请求报文。客户端向服务器发送一个TCP报文段，其中SYN标志位被置为1，同时选择一个初始序列号（ISN）。

第二步，服务器回应SYN-ACK：服务器收到客户端的连接请求后，向客户端发出同意建立连接的同步确认报文。服务器收到客户端的SYN后，回复一个带有SYN和ACK标志位的报文段，确认收到了客户端的SYN，并选择自己的初始序列号。

第三步，客户端发送ACK：客户端在收到服务器的同步确认报文后，向服务器发出确认报文。客户端收到服务器的SYN-ACK后，发送一个带有ACK标志位的报文段作为确认，表示连接建立成功。此时，TCP连接已经建立，可以开始数据传输。

**TCP释放过程（四次挥手）**:

第一步，客户端发送FIN：客户端向服务器发出一个连接释放报文。客户端希望关闭连接时，发送一个带有FIN标志位的报文段给服务器，表示客户端不再发送数据。

第二步，服务器回应ACK：服务器收到客户端的释放连接请求后，向客户端发出确认报文。服务器收到客户端的FIN后，发送一个带有ACK标志位的报文段作为确认，但此时服务器仍可向客户端发送数据。

第三步，服务器发送FIN：服务器在发送完最后的数据后，向客户端发出连接释放确认报文。当服务器也准备关闭连接时，发送一个带有FIN标志位的报文段给客户端，表示服务器不再发送数据。

第四步，客户端回应ACK：客户端在收到服务器连接释放报文后，向服务器发出确认报文。客户端收到服务器的FIN后，发送一个带有ACK标志位的报文段作为确认，连接经过这个ACK之后，进入TIME\_WAIT状态一段时间（等待可能延迟的报文段到达）后，关闭连接。

# 在TCP中有哪些基本的计时器？这些计时器在TCP协议中各自发挥什么样的作用？

**1重传计时器**：

作用：用于处理丢失的数据包。当发送端发送数据后，等待对应的确认ACK，如果在一定时间内没有收到确认，则启动重传计时器，重新发送数据。

**2坚持计时器**：

作用：用于实现持续性超时重传。TCP中的流量控制使用了滑动窗口机制，持续计时器用于控制当接收端的窗口为0（接收端不再接受数据）时，发送端周期性发送一个探测报文段来判断接收端是否已经可以接收数据。

**3保活计时器**:

作用：用于判断两个TCP端点之间长久的连接是否正常。当客户端与服务器建立了TCP连接后，保活计时器即被激活，并设置计时值（通常为2小时）。每当服务器收到来自客户端的报文，即重置计时器。如果发出10个探测报文后依然没有得到客户端的确认报文，则服务器假定客户端遇到了故障，，于是强制关闭这条连接。

**4时间等待计时器**：

作用：在TCP连接释放过程中使用。当连接处于TIME\_WAIT状态时，用于等待延迟的报文段到达。这个状态下的连接保持关闭状态，直到时间等待计时器超时后才关闭连接。

# 什么是停止等待协议？为什么说在有流控的停止等待协议中可能会出现死锁？如何破除死锁？

**S-W协议:**停止等待协议是一种简单的协议，用于实现可靠的数据传输。在停止等待协议中，发送方发送数据后会等待接收方的确认，一旦收到确认，发送方才会发送下一个数据包。接收方收到数据后，发送确认并等待下一个数据包。

**死锁原因**：

①发送方等待确认：发送方发送数据后等待确认，但由于接收方窗口已满，无法及时收到确认。

②接收方等待数据：接收方无法接收更多数据，因为它的窗口已满，所以也不会发送确认。

③这种互相等待的状态导致了死锁，因为两端都在等待对方采取行动而无法继续进行通信。

**如何破除死锁:**

①超时重传：发送方设置重传计时器，如果在一定时间内未收到确认，就认为数据包丢失，重新发送数据。这种机制能够避免长时间等待确认而导致的死锁。

②选择重传：接收方在收到重复的数据包时，可以根据序列号信息判断是否已经收到过该数据，避免重复处理已经接收过的数据。这有助于发送方避免不必要的重传，提高效率。

③流量控制协商：发送方和接收方可以协商更合理的窗口大小或使用其他流量控制方法，以确保发送方不会因为接收方窗口已满而陷入死锁状态。

# 什么是连续ARQ协议？为什么说连续ARQ协议可以大幅度的提高信道利用率？

**连续ARQ协议：**连续ARQ是一种基于反馈的可靠数据传输协议，它允许发送方连续发送多个数据包而无需等待单个数据包的确认。它通过引入滑动窗口机制来实现多个数据包的连续传输和管理。

**原因:**

①流水线化传输：发送方不必等待每个数据包的确认，可以连续发送多个数据包，提高了传输效率。

②更好的网络利用率：连续ARQ允许在一个较大的窗口内发送多个数据包，有效地利用了网络带宽和资源，提高了网络利用率。

③减少等待时间：连续ARQ协议通过允许连续发送数据包，减少了发送方因等待确认而处于空闲状态的时间。

④利用了接收方的处理能力：接收方在收到数据包后立即发送确认，因此在接收方具备足够处理能力的情况下，可以快速地确认多个数据包，使得发送方能够持续发送数据。

# 什么是滑动窗口？举例说明其基本运行过程。

**滑动窗口**:滑动窗口是数据链路层或传输层中一种流量控制和可靠数据传输的技术，用于管理发送端和接收端之间的数据传输。它允许发送方连续发送多个数据包，而无需等待每个数据包的确认，同时可以根据接收端的反馈动态调整发送的数据量。

**运行过程**:

①确定窗口大小：发送方和接收方首先需要确定窗口的大小。窗口大小通常由接收方根据自身的处理能力和网络状况来设定，并通知给发送方。

②发送数据：发送方在每个数据包中都包含一个序列号，以便接收方可以正确地排序数据包。发送方将数据包发送到接收方。

③接收数据：接收方接收到数据包后，将其存储在一个缓冲区中，并等待一段时间以便接收更多的数据包。

④发送确认：当接收方接收到一个完整的窗口大小的数据包后（或者达到一定的时间限制），接收方向发送方发送一个确认信号（ACK），通知发送方这些数据包已经正确接收。

⑤滑动窗口：发送方接收到确认信号后，将窗口向前滑动，移除已经发送并确认的数据包，同时将新的未发送数据包添加到窗口的末尾。

⑥调整窗口大小：如果接收方处理能力有限，接收方可以通过反馈信息通知发送方减小窗口大小，从而降低接收数据的速率。同时，发送方也可以根据接收方的反馈信息来调整发送数据的速率。

例如，假设窗口大小为3，发送方首先发送了序列号为1、2、3的三个数据包，然后等待接收方的确认。接收方接收到这三个数据包后，向发送方发送一个确认信号，包含序列号3的确认。发送方接收到确认信号后，将序列号为3的数据包从窗口中移除，然后将窗口向前滑动一位，添加新的未发送数据包（4、5、6）。然后发送方继续按照这个过程发送数据包。

# 在确定超时重传时间时，RTT、RTTs和RTTD各自发挥着什么作用?

RTT（往返时间）：RTT是指从发送数据包到接收到对应确认的时间。它是网络中数据传输的往返延迟时间。

发挥作用：在确定超时重传时间时，RTT被用来估计数据包从发送到接收并返回的时间。通常，选择一个合适的RTT作为超时时间，以确保在网络正常情况下大多数数据包都能在超时时间内收到确认。

RTTs（往返时间的平均值）：RTTs是指多个RTT的平均值。它用于平滑RTT的变化，减少对单个RTT极端值的影响。

发挥作用：在确定超时重传时间时，RTTs通常作为基准值，考虑了一系列RTT的变化，更能反映网络的实际延迟情况。

RTTD（往返时间的标准差）：RTTD是指RTTs的标准差，表示RTT变化的稳定性或波动性。

发挥作用：在确定超时重传时间时，RTTD可以用来评估网络的不稳定性。如果RTTD较大，表示网络延迟的波动性高，可能会考虑增加超时时间，以应对网络延迟的不确定性。

# 什么是RTO的指数避退？有什么作用？

**RTO的指数退避**：TCP协议中的一种动态调整重传超时时间的机制。这个机制是为了应对网络延迟的变化，通过根据网络状况动态调整超时重传时间，提高数据传输的可靠性。

**作用**:

①适应网络环境：网络的延迟和拥塞可能会导致数据包的丢失或延迟，因此RTO的指数避退机制允许根据网络情况调整重传超时时间，使其更加适应当前网络环境。

②动态调整：根据收到的ACK确认或超时重传，TCP协议使用指数避退算法来调整RTO的时间。当数据包被确认接收时，表示网络状况良好，可以适当减小RTO；而当发生超时重传时，表示可能出现网络拥塞或延迟，此时增加RTO的时间来避免频繁的重传。

③平滑调整：指数避退机制使得RTO的调整更为平滑，而不是在每次超时时都进行剧烈的调整。通过指数避退算法，RTO时间可以缓慢增加，如果连续发生超时重传，则会相对较快地增加。

# 在TCP中，流量控制是怎么实现的？

**在TCP中，利用滑动窗口实现流量控制**

①确定窗口大小：在滑动窗口协议中，首先要确定窗口的大小。窗口大小可以根据网络状况、接收方处理能力等因素进行调整。

②发送数据：发送方在每个数据包中都包含一个序列号，以便接收方可以正确地排序数据包。发送方将数据包发送到接收方，并开始等待接收方的确认。

③接收确认：接收方接收到数据包后，会向发送方发送一个确认信号（ACK），确认信号中包含接收到的数据包的序列号。发送方接收到确认信号后，将该数据包从窗口中移除。

④滑动窗口：当发送方发送完窗口内的所有数据包后，窗口会向前滑动，窗口内的数据包会从窗口中移除，同时窗口的末尾会添加新的未发送数据包。

➄调整窗口大小：如果接收方处理能力有限，发送方可以通过调整窗口大小来控制发送数据的速率。如果接收方无法及时处理接收到的数据包，发送方可以减小窗口大小，从而降低发送数据的速率。

# 简述慢启动和拥塞避免的基本概念

**慢启动**:慢启动是TCP在连接刚开始时的一种启动阶段。其基本概念是在开始时以指数级增加发送窗口的大小，以达到快速适应网络的目的。

工作原理：开始时，发送方以一个较小的发送窗口开始发送数据，然后每收到一个确认ACK，发送窗口大小便会成倍增加。这种指数级增长的方式能够快速适应网络的状况，提高发送速率。当TCP连接建立后，发送方会首先将拥塞窗口大小设为1，然后每次收到确认应答后，将拥塞窗口大小加倍。这种策略可以防止网络拥塞，因为发送方会先试探性地发送少量数据，等待接收方的确认应答，以了解网络的拥塞状况。

**拥塞避免**:拥塞避免是为了避免网络发生拥塞而设计的机制。在慢启动阶段之后，TCP进入拥塞避免阶段，通过线性增加发送窗口大小来逐步增加数据包发送的速率。

工作原理：一旦发送方感知到网络开始拥塞，即出现丢包等拥塞信号，TCP会将发送窗口逐渐增加，但不再采用指数级增长，而是线性增加。这种方式能够缓慢增加数据包发送的速率，更为谨慎地避免网络拥塞。当拥塞窗口大小达到一定阈值后，发送方不再将窗口大小加倍，而是每次收到确认应答时将窗口大小加1。这种策略可以避免因数据包丢失而导致的网络拥塞。当网络出现拥塞时，发送方会立即感知到，并将窗口大小减半，然后重新进入慢启动过程。