姓名 吴忠恒 班级 物联网 19-1 班 学号 2019216864

章节 第六章——传输层

1. **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。**

传输层在 TCP/IP 五层模型处于过渡层，位于网络层之上，应用层之下，是面向应用部分的最低层和面向通信部分的最高层。

传输层（运输层）的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统一的通信服务。

传输层的另外一个主要作用是在应用层和网络层之间充当复用器的作用。应用层的进程请求，通过传输层的端口被提交统一的网络层，并由 IP 协议用统一的方式将它们发送到网络中。

传输层提供以下主要功能：

一是对高层应用屏蔽了通信（低层）的细节，无需过多考虑各种通信因素对网络通信过程本身的影响。

二是提供端到端之间的无差错保证，弥补网络层提供服务的差异和不足。

在网络通信中，传输层有着承上启下的作用，通过提供面向连接和面向无连接的两种服务，实现了一种将面向应用（进程）的通信转变为面向网络设备的通信。

2. **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?**

网络通信中，端口是指在协议栈层间的抽象的协议端口。进一步来讲，传输层中的端口，是指应用层的各种协议进程与传输层协议进行层间交互的一种接口。

传输层地址是一种对有网络通信需求的唯一标记。传输层由主机 IP 地址 + 端口号组成。通过传输层地址，一个用户进程和其他的用户进程区分开，在用户之间提供可靠和有效的端到端服务。

套接字（Socket）是为了使应用程序能够方便地使用协议栈软件进行通信的一种方法。一个完整的套接字由协议，本地地址，本地端口，远程地址，远程端口几个部分组成。在 BSD Linux 系统中，套接字以标准的 UNIX 文件描述符加以标识，应用程序通过对文件描述符的调用，实现与其他程序进行通信。

3. **什么是 UDP 协议? 有什么特点?**

UDP 协议是 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。其基本的目的是提供面向事务的简单、不可靠信息传送服务。

UDP 协议有以下几个主要特点:

(a) UDP 是无连接协议，在发送数据之前不需要建立连接。UDP 使用尽最大努力交付, 不保证可靠交付，同时也不使用拥塞控制。

(b) UDP 是面向报文，没有拥塞控制，很适合多媒体通信的要求。

(c) UDP 支持一对一、一对多、多对一和多对多的交互通信。

(d) UDP 的首部开销小，只有 8 个字节。

4. **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**?

(a) 源端口和目的端口字段

端口是传输层与应用层的服务接口。传输层的复用和分用功能都要通过端口才能实现。

(b) 序号字段

序号字段的值则指的是本报文段所发送的数据的第一个字节的序号。TCP 协议中的流量控制，拥塞控制等机制均依赖则报文首部的序号。

(c) 确认号字段

是期望收到对方的下一个报文段的数据的第一个字节的序号。如果没有确认好，在流量控制环节，发送方就有可能重复发送报文段，导致数据重复。

(d) 窗口字段

该字段用来让对方设置发送窗口的依据，是滑动窗口等基于窗口的流量控制机制的基础。

(e) 检验和

检验和字段检验的范围包括首部和数据这两部分。在计算检验和时，要在 TCP 报文段的前面加上 12 字节的伪首部。通过计算该校验和，TCP 实现了对报文数据的差错控制

5. **简述** TCP **连接建立和释放的基本过程**

TCP 连接的建立过程如下:

第一次: 客户端到服务器。客户端向服务器提出连接建立请求，即发出同步请求报文。

第二次: 服务器到客户端。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。

第三次: 客户端到服务器。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。当服务器收到来自客户端的确认报文后，连接即被建立。

TCP 连接的释放过程如下：

第一次: 客户端到服务器。客户端向服务器发出一个连接释放报文。

第二次: 服务器到客户端。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。

第三次: 服务器到客户端。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。

第四次: 客户端到服务器。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。

6. **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**?

为了保证传输的可靠性和协议栈的稳定，在一次 TCP 连接可以存在多达 9 种不同类型的定时。这些定时器包括：重传计时器，坚持计时器，ER 延迟计时器，PTO 计时器，ACK 延迟计时器，SYNACK 计时器，保活计时器，时间等待计时器，FIN-WAIT2 计时器等。

(a) 重传计时器

当发送方发出数据报文后即启动该计时器 (一般为 60 秒): 在设定时间截至之前收到确认报文，则传输成功, 撤销计时器; 否则，传输失败，重新发送数据报文。

(b) 坚持计时器

当发送方收到一个零窗口确认时，即启动坚持计时器。坚持计时器的值与重传计时器相同。若在截止时间后没有收到接收方的非零窗口确认报文，则发送另一个探测报文，并加倍设置坚持计时器值，如此反复直到收到接收方的非零窗口确认报文为止。

(c) 保活计时器

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。当计时截止后仍未收到客户端的报文，则服务器将向客户端发出探测报文，并每隔 75 秒发送一个探测报文。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。

(d) 时间等待计时器

时间等待计时器，是 TCP 终止连接时启动的计时器。设置这个计时器的主目的是为了能够正常关闭服务端的连接。

7. **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?**

停止等待协议是网络通信中的一种流量控制协议。停止等待协议基于两种假设，一是传输过程中不会出错；二是接收端不保证总能及时接受发送端发出的数据。该协议通过由接收端控制发送端发送数据的速度的方法实现流量控制。

在网络通信中，无论是数据报文还是应答报文，都有可能在传输过程中丢失。不管是数据报文丢失，接收端未收到数据报文的情况，还是应答报文丢失，发送端未收到应答报文的情况，都会导致发送双方都在等待对方的报文，即死锁。

解决死锁的思路是设置重发定时器。发送端在发出报文后即启动重发计时器，若在设置的时间 t_{out} 内未收到接收端的应答报文，则认为报文已丢失，从而重发报文。

8. **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 协议是改进后的停止等待协议。该协议规定，发送方可连续发送多个报文，不必每发完一个报文就停止并等待接收方的确认，即实现了流水线传输。

流水线传输就是发送方可连续发送多个分组，不必每发完一个分组就停顿下来等待对方的确认。这样可使信道上一直有数据不间断地在传送。相比于停止等待协议，信道利用率大大提高。

9. **什么是滑动窗口? 举例说明其基本运行过程。**

滑动窗口是一种流量控制方法，用于约束发送方可发送报文的数量。窗口是指发送方最多可发送未被确认报文的数量，而滑动则是指每收到一个确认报文，窗口可向前滑动一个报文，从而纳入新的待发送的报文。

下面以一个例子说明滑动窗口的基本运行过程

(a) 假设 32 45 这些数据，是上层 Application 发送给 TCP 的，TCP 将其分成四个 Segment 来发往 internet

(b) seg1 32 34,seg3 35 36,seg3 37 41 seg4 42 45 这四个片段，依次发送出去，此时假设接收端之接收到了 seg1 seg2 seg4

(c) 此时接收端的行为是回复一个 ACK 包说明已经接收到了 32 36 的数据，并将 seg4 进行缓存（保证顺序，产生一个保存 seg3 的 hole）

(d) 发送端收到 ACK 之后，就会将 32 36 的数据包从发送并没有确认切到发送已经确认，提出窗口，这个时候窗口向右移动

(e) 假设接收端通告的 Window Size 仍然不变，此时窗口右移，产生一些新的空位，这些是接收端允许发送的范畴

(f) 对于丢失的 seg3，如果超过一定时间，TCP 就会重新传送（重传机制），重传成功会 seg3,seg4 一块被确认，不成功，seg4 也将被丢弃

10. **在确定超时重传时间时，*RTT*、*RTTS* 和 RTT_D, 各自发挥着什么作用**?

测量准确的 RTT 对于确定更加适合的 RTO 具有重要影响。在 TCP 中，如果 RTO 小于 RTT，则会造成很多不必要的重传;RTO 远大于 RTT，则会降低整体网络利用率。

RTTS 的出现，是为了避免 RTT 的偶然波动对 RTO 的计算产生不利影响。通过引入新旧 RTTS，使得计算出来的新 RTTS 能够较为准确地反映整个传输过程地往返时间。

RTTD 同样也是为了优化 RTT 地计算而引入，相对于 RTTS 的平均策略，RTTD 还考虑到了不同过程 RTT 存在的偏差。RTTS 和 RTTD 相结合，就能更好地反映整个传输过程地往返时间。

11. **什么是** RTO **的指数避退**? **有什么作用**?

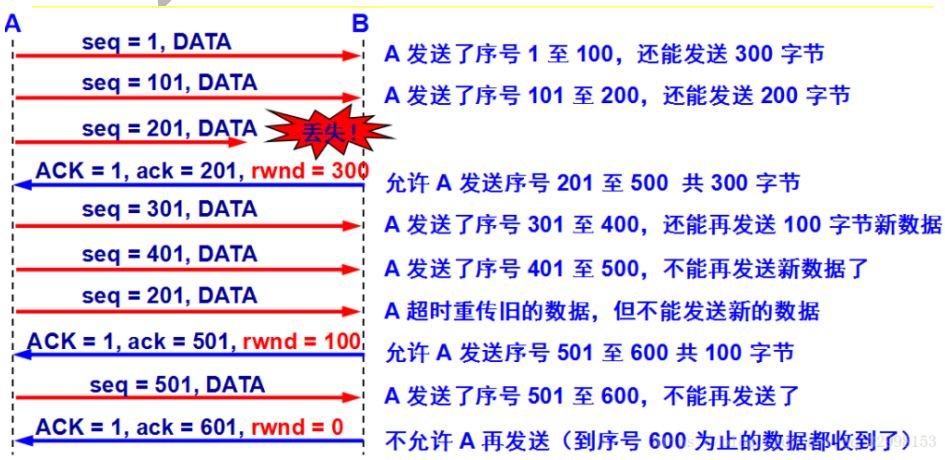
RTO 的指数避是指在 TCP 中，当超时而发生连续多次重传时，在两次重传之间的超时阈值的设置遵循“指数避退”原则。

当超时第一次重传后，第二次重传等待时间是第一次的 2 倍，第三次重传等待时间是第二次的 2 倍,2 为退避因子，直到收到重传数据包的应答，RTO 退避因子回复为 1。

当网络处于无法快速交付数据报文状态时，遵循指数避退原则可以有效减小网络负担。

12. **在** TCP **中，流量控制是怎么实现的**? TCP 协议通过滑动窗口机制实现流量控制。

下面以一个例子说明 TCP 流量控制过程:



(a) 设 A 向 B 发送数据。在连接建立时，B 告诉了 A：“我的接收窗口 rwnd = 400”。

(b) 发送方的发送窗口不能超过接收方给出的接收窗口的数值，请注意，TCP 的窗口单位是字节，不是报文段。

(c) 再设每一个报文段为 100 字节长，而数据报文段序号的初始值设为 1

(d) 图中箭头上面大写 ACK 表示首部中的确认位 ACK，小写 ack 表示确认字段的值。

i. 接收方的主机 N 进行了三次流量控制，第一次把窗口减小到 rwnd =300。

ii. 第二次又减小到 rwnd = 100。

iii. 最后减到 rwnd = 0，即不允许发送方再发生数据了。

iv. 这种使发送方暂停发送的状态将持续到主机 B 重新发出一个新的窗口值为止。

v. B 向 A 发送的三个报文段都设置了 ACK = 1，只有在 ACK = 1 时确认号字段才有意义。

13. **简述慢启动和拥塞避免的基本概念**

慢开始算法：

在刚刚开始发送报文段时，先把拥塞窗口 cwnd 设置为 1 个最大报文段 MSS 的数值，而后每收到一个对新的报文段的确认，就把拥塞窗口增加 1 个 MSS 的数值，这样拥塞窗口 cwnd 的值就随着传输轮次（一个轮次即发送完一个 cwnd 的 MSS）呈指数级增长，事实上，慢启动的速度一点也不慢，只是它的起点比较低一点而已。用这样的方法逐步增大发送方的拥塞窗口 cwnd ，可以使分组注入到网络的速率更加合理。

拥塞避免:

当 cwnd >= ssthresh 时，就会进入 “拥塞避免算法”，让拥塞窗口 cwnd 缓慢地增大，每收到 1 个 ACK 拥塞窗口 cwnd = cwnd + 1/cwnd，即每经过一个传输轮次就把发送方的拥塞窗口 cwnd 加 1。这样拥塞窗口 cwnd 按线性规律缓慢增长，比慢开始算法的拥塞窗口增长速率缓慢得多。