2021302181156 赵伯促 第5章作业

5-01 试说明运输层在协议栈中的地位和作用。运输层的通信和网络层的通信有什么重要的区别？为什么运输层是必不可少的？

地位：运输层处于面向通信部分的最高层，同时也是用户功能中的最低层，

作用：向它上面的应用层提供服务向下兼容网络层，起到承上启下的中间作用

区别：运输层为应用进程之间提供端到端的逻辑通信，但网络层是为主机之间提供逻辑通信（面向主机，承载路由功能，即主机寻址及有效的组交换）

原因：各种应用进程之间的通信需要"可靠或尽力而为"的两类服务质量，处领由运输层以复用和分用的形式加载到网络层

5-06 接收方收到有差错的 UDP 用户数据报时应如何处理？

直接丢弃这个用户数据报，但也可以上交给应用层，但是要附上出现差错的警告

5-07 如果应用程序愿意使用 UDP 完成可靠传输，这可能吗？请说明理由。

可能，但这要由应用层自行完成可靠传输，例如应用层使用自己的可靠传输协议
因为底层不保证可靠传输，又想实现可靠传输，只能够将可靠传输放在上层实现

5-12 一个应用程序用 UDP，到了 IP 层把数据报再划分为 4 个数据报片发送出去。结果前两个数据报片丢失，后两个到达目的站。过了一段时间应用程序重传 UDP，而 IP 层仍然划分为 4 个数据报片来传送。结果这次前两个到达目的站而后两个丢失。试问：在目的站能否将这两次传输的 4 个数据报片组装成为完整的数据报？假定目的站第一次收到的后两个数据报片仍然保存在目的站的缓存中。

不行，因为重传时，IP数据报的标识字段会有另一个标识符，只有标识符相同的IP数据报片才能组装成一个IP数据报

5-13 一个 UDP 用户数据报的数据字段为 8192 字节。在链路层要使用以太网来传送。试问应当划分为几个 IP 数据报片？说明每一个 IP 数据报片的数据字段长度和片偏移字段的值。

数据字段为 8192 B

UDP 首部为 8 B

∴ 共有 8192 + 8 = 8200 B

∵ 每个 MAC 帧数据部分最大长度为 1500 B

且每个 IP 数据报首部 20 B

可以得出 IP 数据报的数据部分长度为 1500 − 20 = 1480 B

∴ $\frac{8200}{1480} = 5 \cdots 800$ B

∴ 应当划分为 6 个 IP 数据报片，

每个 IP 数据报片的数据字段长度和片偏移为：

	数据字段长度	片偏移	
①	1500 B	0	包含有 20 B 的首部
②	1500 B	1480 / 8 = 185	
③	1500 B	2×1480/8 = 370	
④	1500 B	3×1480/8 = 555	
⑤	1500 B	4×1480/8 = 740	
⑥	800 B	5×1480/8 = 925	

5-14 一个 UDP 用户数据报的首部的十六进制表示是：06 32 00 45 00 1C E2 17。试求源端口、目的端口、用户数据报的总长度、数据部分长度。这个用户数据报是从客户发送给服务器还是从服务器发送给客户？使用 UDP 的这个服务器程序是什么？

源端口 $=(0632)_{16}$ $=1586$

目的端口 $=(0045)_{16}$ $=69$

长度 $=(001C)_{16}$ $=28$

检验和 $=(E217)_{16}$

∴源端口为1586

目的端口为69

用户数据报的总长度 $=28B$

数据部分长度 $=28-8=20B$

∵目的端口 $=69<1023$ 为熟知端口

∴数据报是从客户端发送给服务器的，服务器程序是 TFTP 简单文件传输协议

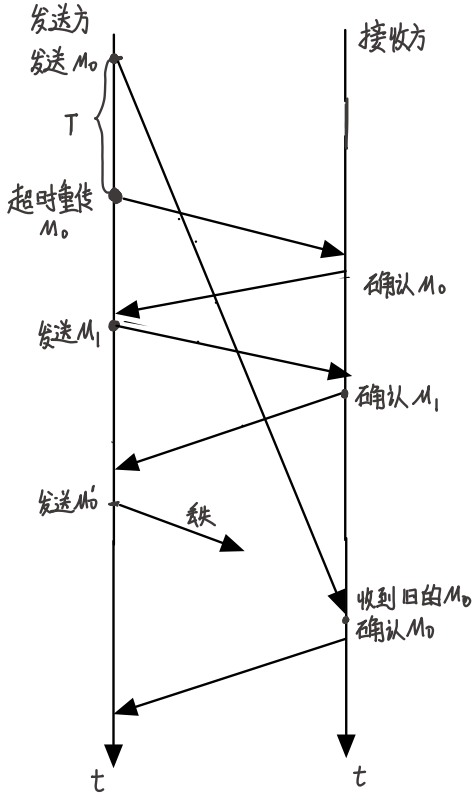
5-17 在停止等待协议中，如果收到重复的报文段时不予理睬（即悄悄地丢弃它而其他什么也不做）是否可行？试举出具体例子说明理由。

不可行，因为如果接收方在收到重复报段时，不给发送方发送确认分组，
发送方会认为接收方没有正确接收这个报段，会一直重传，
达到一定的重传次数后会认为是网络出现故障

例如发送方发送 M_1 后，接收方不予理睬，则发送方会因为没有收到 M_1 的确认信号而不停重传 M_1

5-18 假定在运输层使用停止等待协议。发送方发送报文段 M_0 后在设定的时间内未收到确认，于是重传 M_0，但 M_0 又迟迟不能到达接收方。不久，发送方收到了迟到的对 M_0 的确认，于是发送下一个报文段 M_1，不久就收到了对 M_1 的确认。接着发送方发送新的报文段 M_0，但这个新的 M_0 在传送过程中丢失了。正巧，一开始就滞留在网络中的 M_0 现在到达接收方。接收方无法分辨 M_0 是旧的。于是收下 M_0，并发送确认。显然，接收方后来收到的 M_0 是重复的，协议失败了。

试画出类似于图 5-9 所示的双方交换报文段的过程。

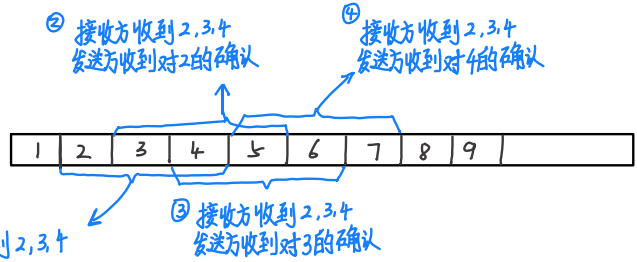


5-21 假定使用连续 ARQ 协议，发送窗口大小是 3，而序号范围是[0, 15]，而传输媒体保证在接收方能够按序收到分组。在某一时刻，在接收方，下一个期望收到的序号是 5。试问：

(1) 在发送方的发送窗口中可能出现的序号组合有哪些？

(2) 接收方已经发送出的、但在网络中（即还未到达发送方）的确认分组可能有哪些？说明这些确认分组是用来确认哪些序号的分组。

(1) 因为接收方能够按序收到分组则发送窗口可能组合为



∴ 可能的组合有 234，345，456，567

(2) ① 窗口中为 2 3 4，接收方期望收到5说明已经对 2,3,4发送确认但未收到

② 窗口中为 3.4.5，接收方期望收到5说明已经对 3,4 发送确认但未收到

③ 窗口中为 4,5,6，接收方期望收到5说明已经对 4 发送确认但未收到

④ 窗口中为 5,6,7，接收方期望收到5说明已经对 4 发送确认且已收到

∴ 2,3,4分组可能已发送但未确认

5-22 主机 A 向主机 B 发送一个很长的文件，其长度为 L 字节。假定 TCP 使用的 MSS 为 1460 字节。

(1) 在 TCP 的序号不重复使用的条件下，L 的最大值是多少？

(2) 假定使用上面计算出的文件长度，而运输层、网络层和数据链路层所用的首部开销共 66 字节，链路的数据率为 10 Mbit/s，试求这个文件所需的最短发送时间。

(1) TCP的序号长度为4B，共可表示 $2^{4\times8}=2^{32}$ 个序号

∵每个序号可以对应一个字节

∴ L的最大值 $=2^{32}B=4GB$

(2) ∵ TCP使用的MSS为1460B

∴需要的报文段数 $=\frac{4GB}{1460B}=\frac{2^{32}}{1460}$

每个报文段首部有66B

∴总数据量 $=\frac{2^{32}}{1460}\times66B+4GB$

∴最短发送时间 $=\frac{2^{32}\left(\frac{66}{1460}+1\right)\times8\ bit}{10\times10^{6}\ bit/s}$

$=\frac{4294967296\times1526\times8}{1460\times10^{7}}\approx3591.299\ s$

5-23 主机 A 向主机 B 连续发送了两个 TCP 报文段，其序号分别是 70 和 100。试问：

(1) 第一个报文段携带了多少字节的数据？

(2) 主机 B 收到第一个报文段后发回的确认中的确认号应当是多少？

(3) 如果 B 收到第二个报文段后发回的确认中的确认号是 180，试问 A 发送的第二个报文段中的数据有多少字节？

(4) 如果 A 发送的第一个报文段丢失了，但第二个报文段到达了 B。B 在第二个报文段到达后向 A 发送确认。试问这个确认号应为多少？

(1) 数据字节序号为 70 ~ 100-1 = 70~99，携带了 30个字节

(2) 确认号应为 99+1 = 100

(3) 第二个报文段序号为 100 ~ 180-1 = 100~179，有 80个字节

(4) 因为B没有收到第一个报文段，所以B期望收到的报文序号应仍为70

5-25 为什么在 TCP 首部中要把 TCP 的端口号放入最开始的 4 个字节？

因为在ICMP的差错报文中要包含紧随IP头部后面的8个字节的内容，
让在TCP头部最开始的4个字节是TCP的端口号，
就可以在ICMP的差错报文的上述8个字节中包含TCP的源端口和目的端口，
当发送IP分组的源收到ICMP差错报文时需要用这两个端口来确定是
哪个应用进程的网络通信出了差错

5-26 为什么在 TCP 首部中有一个首部长度字段，而 UDP 的首部中就没有这个字段？

因为 UDP 提供的无连接的通信服务，首部也是固定的8个字节，而 TCP 报文段除了固定首部20字节，还有一个可选项，为了使接收方知道哪些字节是数据部份，哪些字节是首部，于是设置了这样一个字段标识首部长度

5-37 在 TCP 的拥塞控制中，什么是慢开始、拥塞避免、快重传和快恢复算法？这里每一种算法各起什么作用？“乘法减小”和“加法增大”各用在什么情况下？

慢开始：
- 算法：在主机刚刚开始发送报文段时可先将拥塞窗口 cwnd 设置为一个最大报文段 SMSS 的值。在每收到一个对新的报文段的确认后，将拥塞窗口增加至多一个 SMSS 的数值，即为
  拥塞窗口每次增加量 = min(N, SMSS)
  其中 N 是原先未被确认、但现在被刚收到的确认报文段所确认的字节数
- 作用：试探性地控制拥塞窗口，避免网络出现阻塞

拥塞避免：
- 算法：每经过一个往返时间 RTT，发送方的拥塞窗口 cwnd 的大小就 +1，拥塞窗口 cwnd 按线性规律缓慢增长
- 作用：减缓慢开始的拥塞窗口增长速率，以避免过快增长带来的拥塞

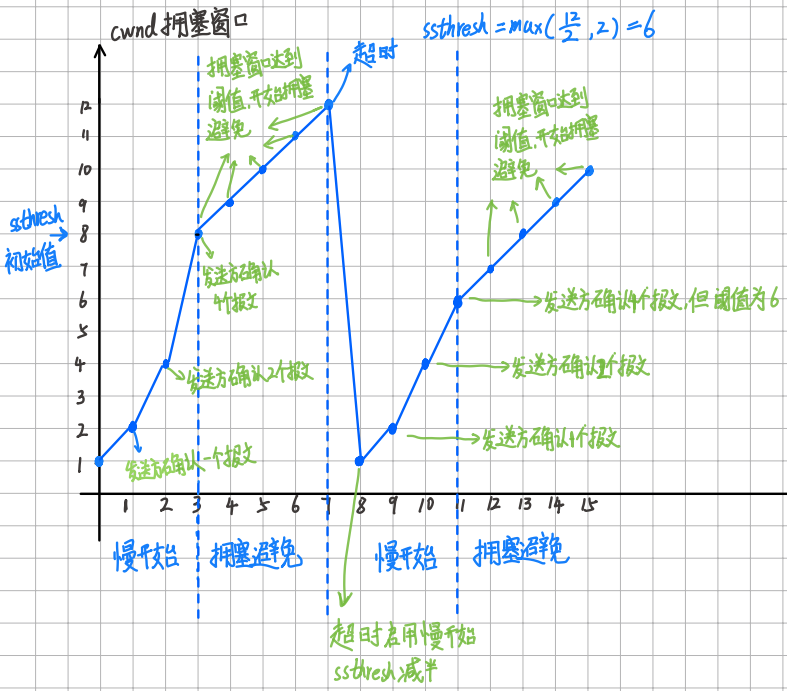
快重传：
- 算法：发送方只要连续收到 3 个重复的确认，就立即进行重传
  将 ssthresh 设置为 max(cwnd/2, 2)
  将 cwnd 设为 1
- 作用：迅速减少主机发送到网络中的分组数，使得发生拥塞的路由器有足够时间把队列中积压的分组处理完毕

快恢复：
- 算法：当发送方连续收到 3 个重复的确认后将慢开始的门限减半，将新拥塞窗口 cwnd 设置为慢开始门限 ssthresh
- 作用：使拥塞窗口不会被过分地减小，提高传输效率

加法增大 用在当慢开始算法使拥塞窗口的大小 ≥ 门限值 时

乘法减小 用在当发送方连续收到 3 个重复确认的时候

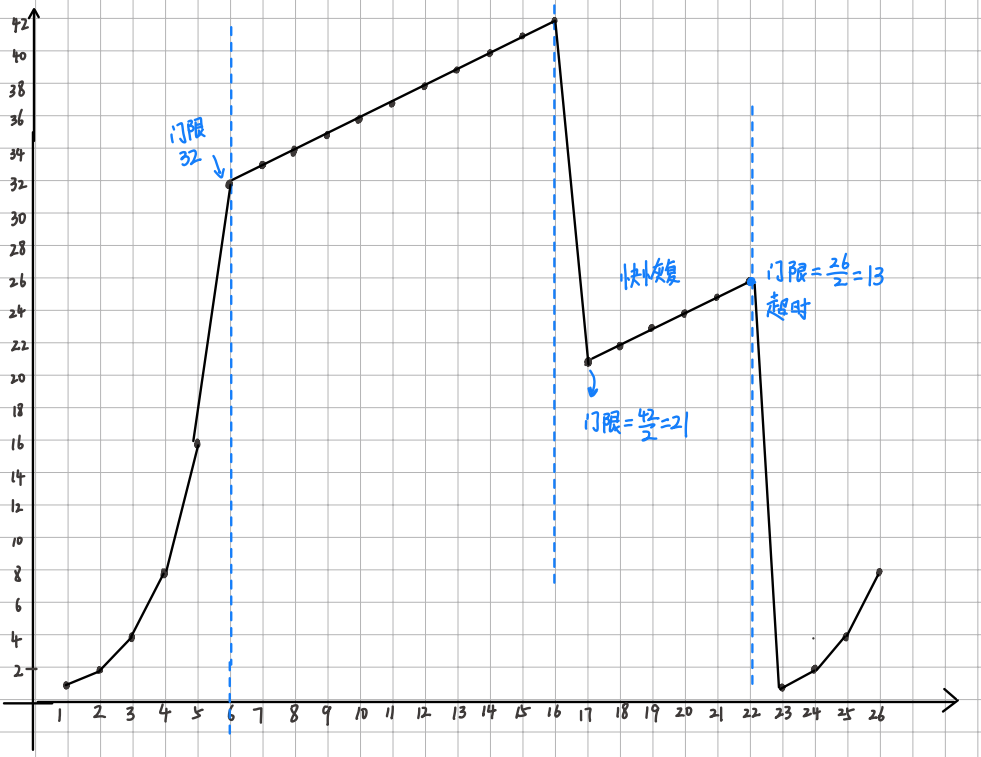
5-38 设 TCP 的 ssthresh 的初始值为 8（单位为报文段）。当拥塞窗口上升到 12 时网络发生了超时，TCP 使用慢开始和拥塞避免。试分别求出 RTT = 1 到 RTT = 15 的各拥塞窗口大小。你能说明拥塞窗口每一次变化的原因吗？



5-39 TCP 的拥塞窗口 cwnd 大小与 RTT 的关系如下所示：

cwnd	1	2	4	8	16	32	33	34	35	36	37	38	39
RTT	1	2	3	4	5	6	7	8	9	10	11	12	13
cwnd	40	41	42	21	22	23	24	25	26	1	2	4	8
RTT	14	15	16	17	18	19	20	21	22	23	24	25	26

(1) 试画出如图 5-25 所示的拥塞窗口与 RTT 的关系曲线。
(2) 指明 TCP 工作在慢开始阶段的时间间隔。
(3) 指明 TCP 工作在拥塞避免阶段的时间间隔。
(4) 在 RTT = 16 和 RTT = 22 之后发送方是通过收到三个重复的确认还是通过超时检测到丢失了报文段？
(5) 在 RTT = 1、RTT = 18 和 RTT = 24 时，门限 ssthresh 分别被设置为多大？
(6) 在 RTT 等于多少时发送出第 70 个报文段？
(7) 假定在 RTT = 26 之后收到了三个重复的确认，因而检测出了报文段的丢失，那么拥塞窗口 cwnd 和门限 ssthresh 应设置为多大？



(2) 慢开始阶段时间间隔为 1~6，23~26
(3) 拥塞避免时间间隔为 6~16，17~22
(4) RTT=16 时是收到3个重复确认，RTT=22时是检测到超时
(5) RTT=1 时门限为32，RTT=18时门限为21，RTT=24时，门限为13
(6) 跌时 cwnd=8，减半后 cwnd=4，门限=cwnd=4

5-40 TCP 在进行流量控制时，以分组的丢失作为产生拥塞的标志。有没有不是因拥塞而引起分组丢失的情况？如有，请举出三种情况。

① 当IP数据报在传输过程中需要分片，但其中一个数据报片未能及时到达终点，而终点组装IP数据报已超时，因而只能丢弃该数据报

② IP数据报已经到达终点，但终点的缓存没有足够的空间存放此数据报

③ 数据报在转发过程中经过一个局域网的网桥，但网桥在转发该数据报的帧时没有足够的空间只好丢弃

5-45 解释为什么突然释放运输连接就可能会丢失用户数据，而使用 TCP 的连接释放方法就可保证不丢失数据。

当主机1和主机2之间连接建立后，主机1发送了一个TCP数据段并正确抵达主机2，接着主机1发送另一个TCP数据报，但是主机2在收到第2个TCP数据段之前发出了释放连接请求，如果就这样突然释放连接，显然主机1发送的第二个TCP报文段会丢失，而使用TCP的连接释放方式，主机2发出了释放连接的请求，那么即使收到主机1的确认后，只会释放主机2到主机1方向的连接，即主机2不再向主机1发送数据，而仍然可接收主机1发来的数据，所以可保证不丢失数据

5-61 在本题中列出的 8 种情况下，画出发送窗口的变化，并标明可用窗口的位置。已知主机 A 要向主机 B 发送 3 KB 的数据。在 TCP 连接建立后，A 的发送窗口大小是 2 KB。A 的初始序号是 0。

(1) 一开始 A 发送 1 KB 的数据。

(2) 接着 A 就一直发送数据，直到把发送窗口用完。

(3) 发送方 A 收到对第 1000 号字节的确认报文段。

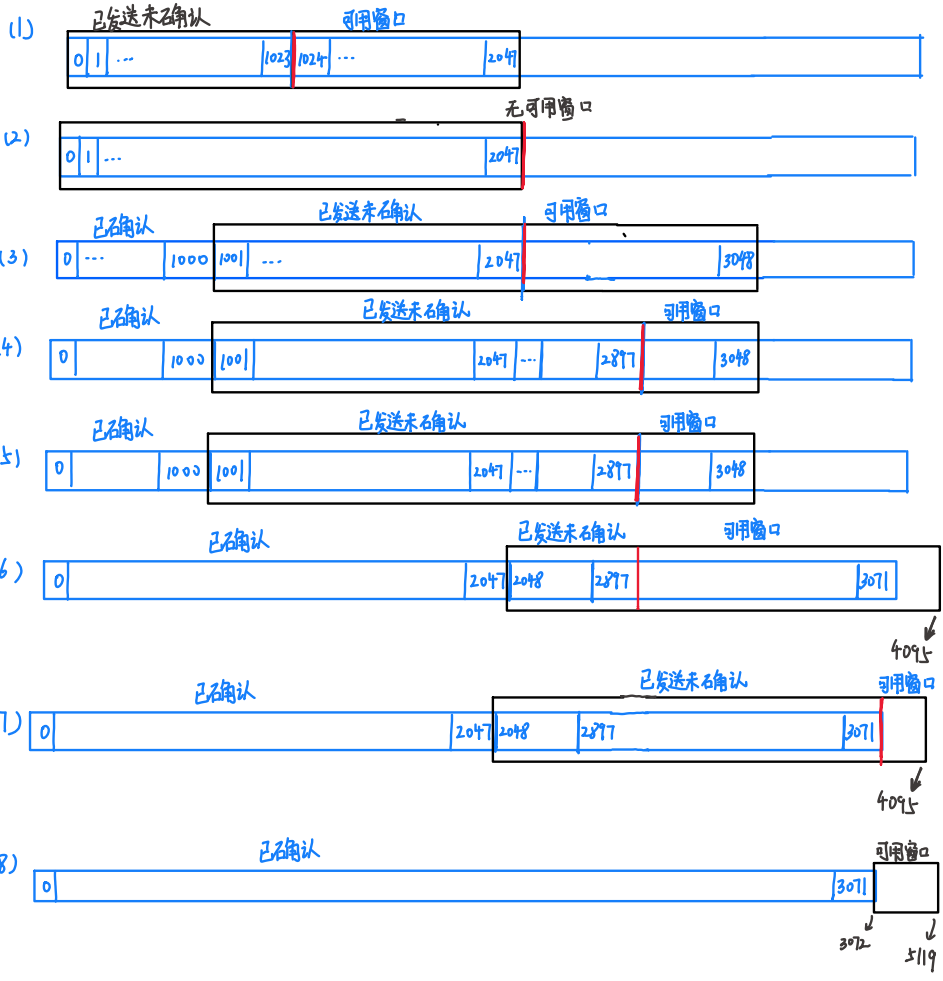
(4) 发送方 A 再发送 850 B 的数据。

(5) 发送方 A 收到 ack = 900 的确认报文段。

(6) 发送方 A 收到对第 2047 号字节的确认报文段。

(7) 发送方 A 把剩下的数据全部都发送完。

(8) 发送方 A 收到 ack = 3072 的确认报文段。



5-74 流量控制和拥塞控制的最主要的区别是什么？发送窗口的大小取决于流量控制还是拥塞控制？

区别：流量控制是在一条TCP连接中的接收端才用的措施，用来限制发送端发送报文的速率，以免在接收端来不及接收，只控制发送端

拥塞控制用来控制TCP连接中发送端发送报文的速率，以免在互联网中的某处发生过载，可能会控制多个发送端限制其发送报文的速率

发送窗口：发送窗口的上限是 min[rwnd, cwnd]，

接收窗口的大小体现了接收端对发送端施加的流量控制

拥塞窗口的大小体现了整个网络的负载情况对发送端施加的拥塞控制

当接收窗口小于拥塞窗口时发送窗口的大小取决于流量控制，即取决于接收端接收能力

当拥塞窗口小于接收窗口发送窗口的大小取决于拥塞控制即取决于整个网络拥塞情况