第三章 运输层

1. **假设客户 A 通过端口467 向服务器 S 发起一个 Telnet 会话（Telnet 会话的端口号为 23）。以此同时，客户 B 通过端口号 513 也向服务器 S 发起一个 Telnet 会话。给出下面报文段的源、目的端口号。**

（a）从A向S发送报文段。

（b）从B向S发送报文段。

（c）从S向A发送报文段。

（d）从S向B发送报文段。

（e）如果A和B是不同的主机，那么从A向S发送的报文段的源端口号是否可能与从B向S发送的报文段的源端口号相同？

（f）如果它们是同一台主机，情况会怎么样？

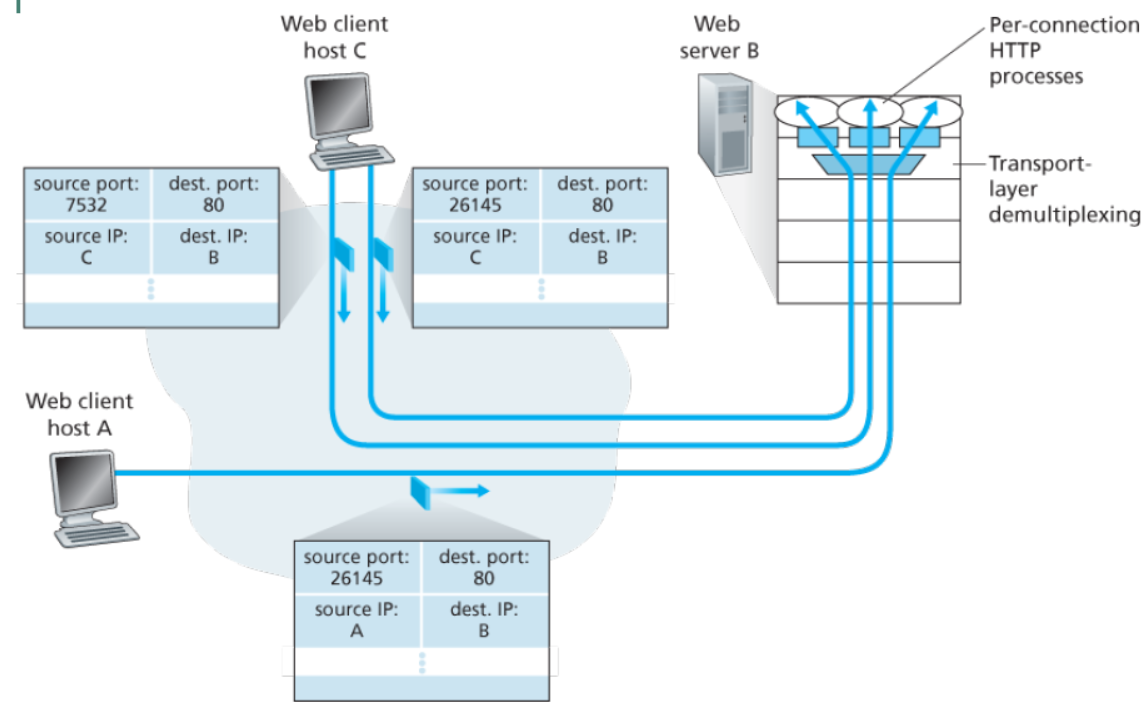
解答：

|  |  |  |
| --- | --- | --- |
|  | 源端口号 | 目的端口号 |
| a) AS | 467 | 23 |
| b) BS | 513 | 23 |
| c) SA | 23 | 467 |
| d) SB | 23 | 513 |

e) Yes.

f) No.

1. **考虑下图。从服务器返回客户进程的报文流中的源端口号和目的端口号是多少？在承载运输层报文段的网络层数据报中，IP地址是多少？**



解答：

假设主机A、B、C的IP地址分别为A、B、C。

主机A：源端口=80，源IP地址= A，目的端口= 26145，目的IP地址= A；

主机C，左边进程：源端口=80，源IP地址= B，目的端口= 7532，目的IP地址= C；

主机C，右边进程：源端口=80，源IP地址= B，目的端口= 26145，目的IP地址= C。

1. **UDP和TCP使用反码来计算它们的校验和。**

**假设有下面3个8比特字节：01010011, 01100110, 01110100。这些8比特字节的反码是多少？（注意：尽管UDP和TCP使用16比特的字来计算校验和，但对于这个问题，考虑的是8比特和。）写出计算过程。UDP为什么要用该和的反码，即为什么不直接使用该和？使用该反码方案，接收方如何检测出差错？1比特的差错能够被检测出来吗？2比特的差错呢？**

解答：

注意，如果溢出，则回卷。





二进制反码为：1 1 0 1 0 0 0 1。

为了检测错误，接收方累加四个字节（三个原始字节和校验和）。如果总和包含0，则接收方认为有错误。

所有的1位错误都将被检测到，但是2位错误可以不被检测到（例如，如果第一个字节的最后一位变成了0，而第二个字节的最后一位变成1）。

1. **一UDP用户数据报的首部十六进制表示是：06 32 00 45 00 1C E2 17。**

**（1）试求源端口、目的端口、用户数据报的总长度、数据部分长度。**

**（2）这个用户数据报是从客户发送给服务器发送给客户？**

**解答：**

该UDP报文的首部为：

|  |  |
| --- | --- |
| 源端口号（0632） | 目的端口号（0045） |
| UDP报文段长度（001C） | 校验和（E217） |
| 数据 | |

源端口是0x0632 = 1586

目的端口是0x0045 = 69

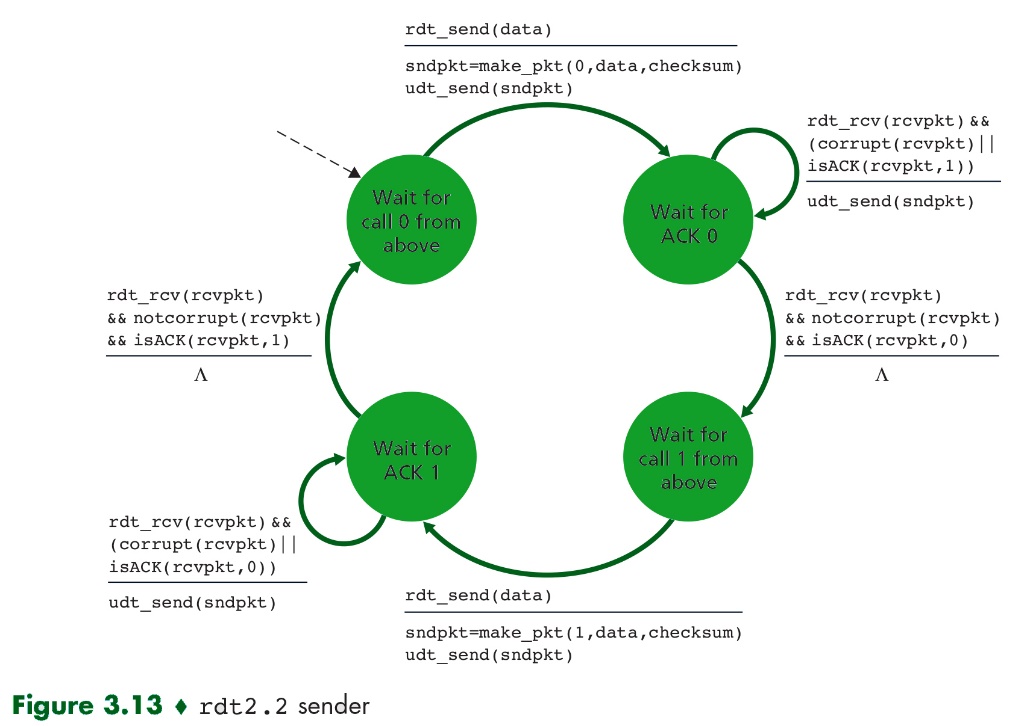
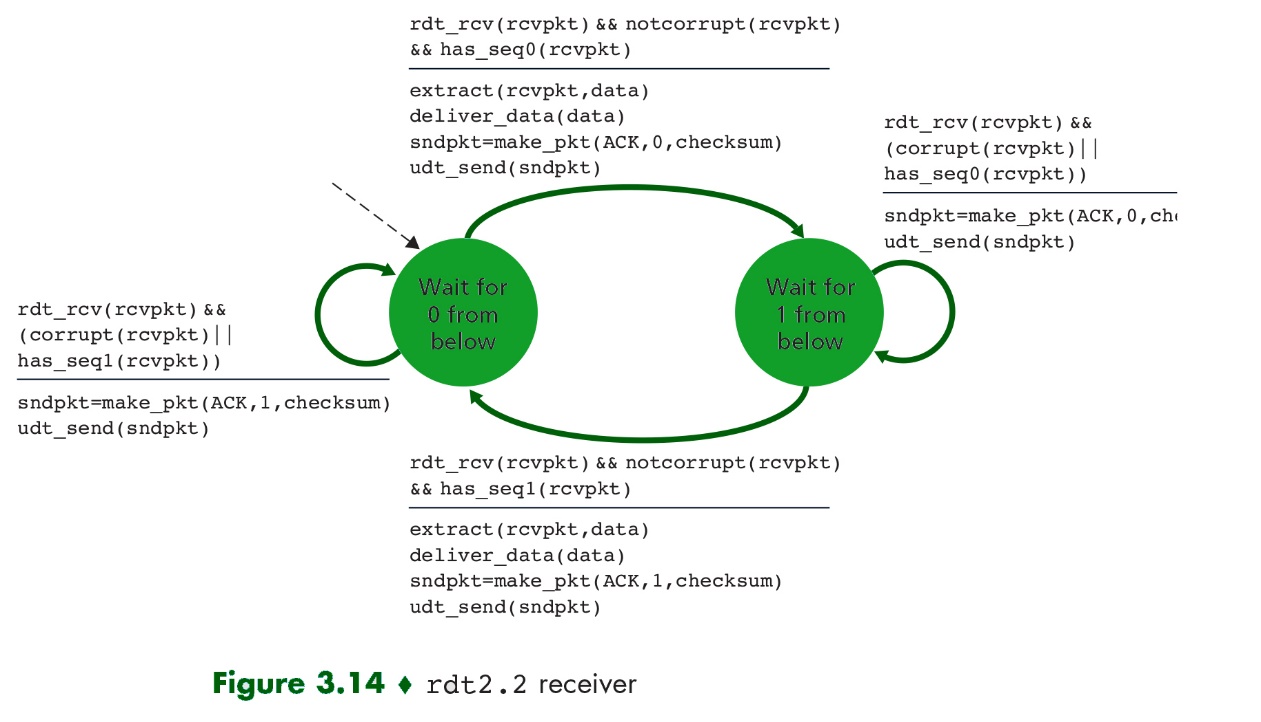
UDP 用户数据报总长度是0x001C = 28

数据部分长度是28-8 = 20。

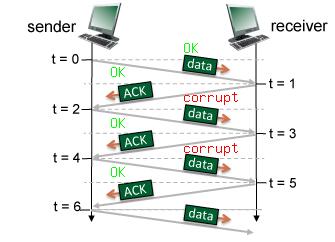
因为目的端口号69 < 1023，是熟知端口，所以此UDP用户数据报是从客户发给服务器。服务器程序实际就是TFTP。

1. **可靠的数据传输：rdt2.2（发送方和接收方动作）**

参考教材P140中的rdt2.2协议。发送方和接收方的FSMs如下：

假设连接发送方和接收方的通道可能损坏，但不会丢失或重新排序分组。现在考虑下图，它显示了rdt 2.2发送方和接收方之间交换的四个数据分组和三个对应的ACK。分组的实际损坏或成功传输/接收分别由corrupt 和OK 表示，如下图所示。



1. 填写下表表明 (i) 在时间t响应收到的分组发送新分组之后发送者和接收者的状态，（ii）与分组相关的序号或与在时间t发送的ACK分组相关联的ACK号。

**解答：**

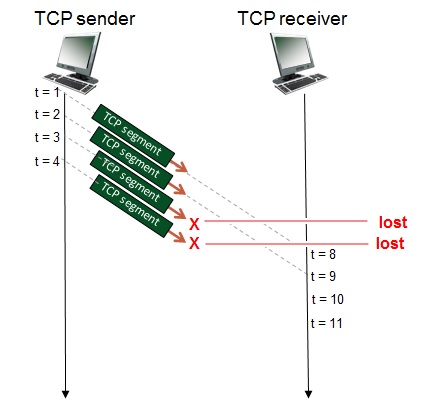
1. 完整表格如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| t | sender state | receiver state | packet type sent | seq. # or ACK # sent |
| 0 | Wait ACK0 | Wait0 from below | data | 0 |
| 1 | Wait ACK0 | Wait1 from below | ACK | 0 |
| 2 | Wait ACK1 | Wait1 from below | data | 1 |
| 3 | Wait ACK1 | Wait1 from below | ACK | 0 |
| 4 | Wait ACK1 | Wait1 from below | data | 1 |
| 5 | Wait ACK1 | Wait1 from below | ACK | 0 |
| 6 | Wait ACK1 | Wait1 from below | data | 1 |

（2）t = 1时，一个数据包被传递到接收端的更高层。

1. **TCP序列和数据段丢失的ACK编号。**

考虑下图，TCP发送方和接收方通过连接进行通信，其中发送方和接收方报文段可能丢失。TCP发送方分别在t=1,2,3,4处发送四个报文段的初始窗口。假设发送端到接收端序列号的初始值是122，前四个段每个包含537个字节。发送方和接收方之间的延迟为7个时间单位，因此第一个报文段到达接收方的时间为t=8。如图所示，四个报文段中的两个在发送方和接收方之间丢失。



回答以下问题：

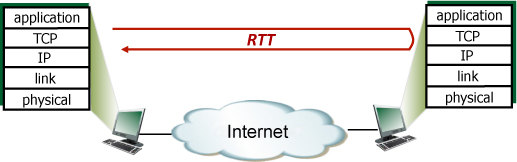
* 1. 给出与发送方发送的四个报文段中的每个报文段相关联的序号；
  2. 列出TCP接收方在接收到实际接收到的报文段时发出的确认序号。特别是，在每个接收方到发送方确认的确认字段中给出值。

**解答：**

发送方发送的四个段中每个段的序号为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **发送方到接收方** | **报文段发送时间** | **发送方到接收方报文段序号** | **报文段接收，发送ACK的时间** | **接收方到发送方ACK序号** |
| 报文段1 | 1 | 122 | 8 | 659 |
| 报文段2 | 2 | 659 | 9 | 1196 |
| 报文段3 | 3 | 1196 |  | 无ACK发送，因为该报文段丢失 |
| 报文段4 | 4 | 1733 |  | 无ACK发送，因为该报文段丢失 |

1. **计算TCP的RTT和超时值**



假设TCP当前对往返时间的估计值（estimatedRTT）和RTT中的偏差（DevRTT）分别为340 msec和19 msec（有关这些变量的讨论，请参见第3.5.3节）。假设RTT接下来的三个测量值分别为260、330和270。

在得到这三个测量的RTT值之后，计算TCP的新值estimatedRTT、DevRTT和TCP超时值（α = 0.125 ；β = 0.25.）

**解答：**

**在作出第一次RTT估计后：**

 estimatedRTT = 0.875\*340 + 0.125\*260 = 330 msecs

DevRTT = 0.75\*19 + 0.25\*(abs(260 - 330)) = 31.75 msecs

Timeout Interval = 330 + 4\*31.75 = 457 msecs

**在作出第二次RTT估计后：**

estimatedRTT = 0.875\*330 + 0.125\*330 = 330 msecs

DevRTT = 0.75\*31.75 + 0.25\*(abs(330 - 330)) = 23.8125 msecs

Timeout Interval = 330 + 4\*23.8125 = 425.25 msecs

**在作出第三次RTT估计后：**

estimatedRTT = 0.875\*330 + 0.125\*330 = 322.5 msecs

DevRTT = 0.75\*23.8125 + 0.25\*(abs(270 - 322.5)) = 23.8125 msecs

Timeout Interval = 322.5 + 4\*30.984375 = 446.4375 msecs

1. **主机A和B通过一个TCP连接通信，且主机B已经收到了来自A的直到字节500的所有字节。假定主机A随后向B发送两个报文段。第一个报文段和第二个报文段分别包含了50和60字节的数据。**

**第一个报文段和第二个报文段的序号分别是 (1) 和 (2)。无论何时B收到A的报文段，它都会发送确认，确认号是 (3) 和 (4)。假定两个报文段按序到达B，第一个确认丢失，第二个确认在第一个超时间隔后到达A。故主机A需重新发送未收到确认的报文段，重新发送报文段的序号是 (5)，字节数是 (6)，待主机B收到该重新发送的报文段后，回复确认报文段，其确认号是 (7)。分别给出 (1) ~ (7) 中的数值。**

Host B

Host A

Timeout interval

Timeout interval

Ack = (7)

Ack = (4)

Seq = (5), (6) bytes

Seq = (1), 50 bytes

Seq = (2), 60 bytes

Ack = (3)

解答：(1) = 501

(2) = 551

(3) = 551

(4) = 611

(5) = 501

(6) = 50

(7) = 611

1. **TCP慢启动、拥塞避免和快速重传。**

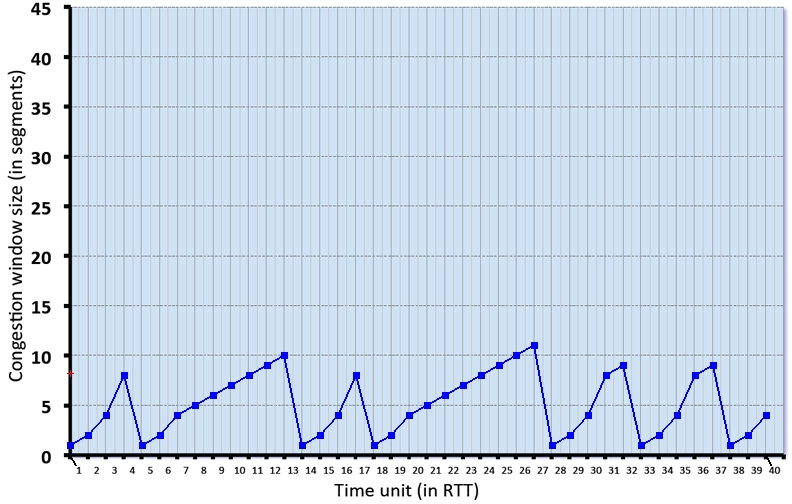
**考虑下图，它绘制了TCP拥塞窗口在每个时间单元（其中时间单元等于RTT）开始时的演变，参考教材图3.53。在这个抽象模型问题中，TCP在每个时间单元的开始发送一个size为cwnd的分组。发送该分组的结果是：**

(i) 所有分组在时间单元结束后被确认，

(ii) 第一个分组有超时，或者

(iii) 第一个分组有一个三个重复（冗余）ACK。

请重现TCP 拥塞窗口事件序列（ACK、丢失）。



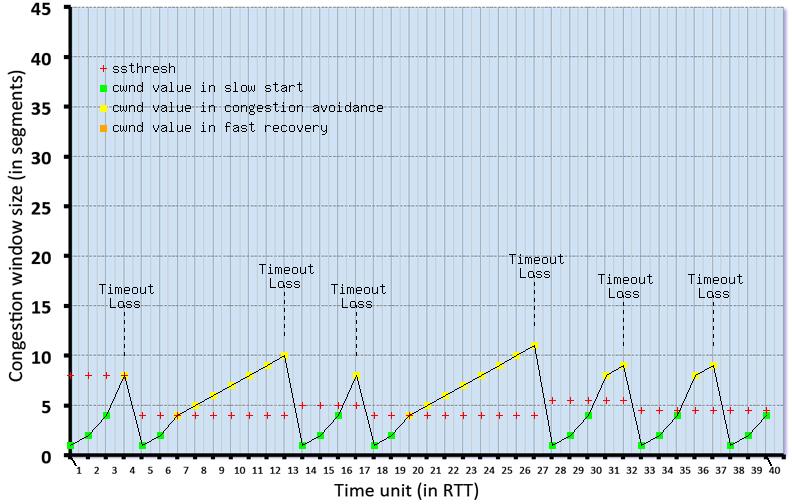
在上面的例子中，考虑TCP拥塞窗口的演变，回答以下问题。其中cwnd的初值为1，ssthresh的初值（请用红色+符号表示）为8。

* 在开始时间，也就是分组刚发送的时候，给出TCP慢启动、拥塞避免和快速恢复的时间。
* 给出分组第一个数据包丢失的时间，并指出该数据包丢失是否通过超时检测到，而不是通过三个重复ACK检测到。
* 给出ssthresh值变化的时间，并给出ssthresh的新值。

**解答：**

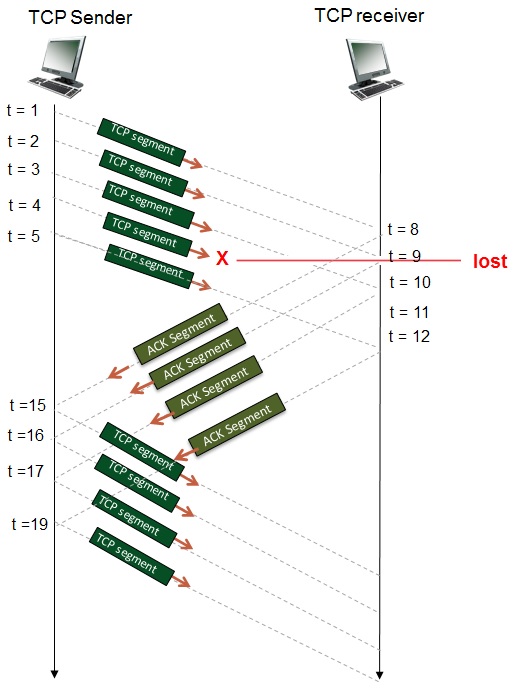
解决方案如下图所示。

对于TCP启动较慢的时间间隔，cwnd的标绘值显示为绿色方块。对于TCP处于拥塞避免状态的时间间隔，cwnd的标绘值显示为黄色方块。当TCP处于快速重合时，cwnd的标绘值显示为橙色的正方形。ssthreshare的值在更改后显示为红色加号。经历丢失的数据包的飞行具有丢失类型(确定上面标记的cwnd的下一个值)。



1. **TCP重传（可靠的数据传输和拥塞控制）**

**考虑下图，TCP发送方和接收方通过连接进行通信，其中发送方到接收方的段可能丢失。TCP发送方分别在t=1,2,3,4,5处发送五个段的初始窗口。假设发送端到接收端序列号的初始值为91，并且所有发送端到接收端段都包含571字节。发送方和接收方之间的延迟为7个时间单位，因此第一个段到达接收方的时间为t=8，此段的接收方到发送方ACK到达TCP发送方的时间为t=15。如图所示，在发送方和接收方之间丢失了五个段中的一个，但是没有一个接收方到发送方的ack丢失。回答以下问题:**



* 给出从t=1开始发送的第一组段中发送方到接收方段中的序列号字段值，以及从t=8开始从接收方发送回发送方的接收方到发送方ACK段中的确认号字段值。
* 接下来考虑发送方为响应从t=15开始接收的ack而生成的发送方到接收方的段。给出所传输段的序号，并简要说明在接收ACK时为什么要传输或不传输给定段。

**解答：**

* 发送方到接收方ACK段中的序列号和接收方到发送方ACK段中的确认号如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **发送方到接收方** | **发送时间** | **发送报文段序号** | **报文段接收时间，且ACK发送** | **接收方回应ACK序号** |
| Segment 1 | t=1 | 91 | 8 | 662 |
| Segment 2 | t=2 | 662 | 9 | 1233 |
| Segment 3 | t=3 | 1233 | 10 | 1804 |
| Segment 4 | t=4 | 1804 |  | No ACK sent since the segment was lost. |
| Segment 5 | t=5 | 2375 | 12 | 1804 Note that ACK this re-acknowledges the last correctly received, in-order byte(1804) |

* 响应每个ACK传输的段序号为：

|  |  |  |
| --- | --- | --- |
| **发送方到接收方** | **发送时间** | **发送方回应收到报文段序号** |
| Segment 1 | t=15 | 1804 |
| Segment 2 | t=16 | 2946 |
| Segment 3 | t=17 | 3517 |
| Segment 4 | t=19 | 4088 |

1. **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有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文件长度，而运输层、网络层和数据链路层所使用的首部开销共66字节，链路的数据率为10Mbps/s，试求这个文件所需的最短发送时间。**

**解答：**

（1）因为TCP协议的序号为32位，所以L的最大值是232 = 4GB。

（2）需要分成的TCP报文段数目为

X = LMAX / 1460 = 2941758

则发送的总字节数为原始数据，加上所传输报文段的首部（不考虑连接建立阶段，仅考虑数据的传输所花费的时间）

N = L + 66 \* X = 232 + 66 \* 232/1460

发送N字节需时间为：N×8/(10×106)。

1. **主机A向主机B连续发送了两个TCP报文段，其序号分别为70和100。试问：**

**（1）第一个报文段携带了多少个字节的数据？**

**（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）如果主机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？**

**解答：**

（1）第一个报文段的数据序号是70到99，共30字节的数据。

（2）确认号应为100。

（3）80字节。

（4）70 。

1. **一个TCP报文段的数据部分最多有多少个字节？为什么？如果用户要传送的数据的字节长度超过TCP报文字段中的序号字段可能编出的最大序号，问还能否用TCP来传送？**

**解答：**

因为TCP报文段的数据部分加上TCP首部的20字节，再加上IP首部的20字节，正好是IP数据报的最大长度65535。所以TCP报文段的数据部分最多有65535-40=65495个字节。

可以使用TCP协议来传送，因为序号字段编号到最大值，又重新开始从0开始编号。

1. **一个TCP报文段的数据部分最多有多少个字节？为什么？如果用户要传送的数据的字节长度超过TCP报文字段中的序号字段可能编出的最大序号，问还能否用TCP来传送？**

**解答：**

因为TCP报文段数据部分加上TCP首部的20字节，再加上IP首部的20字节，正好是IP数据报的最大长度65535，所以TCP报文段数据部分最多有65535-40=65495个字节。

可以使用TCP协议来传送，因为序号字段编号到最大值，又重新开始从0开始编号。

1. **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？**

答：TCP首部除固定长度部分外，还有选项，因此TCP首部长度是可变的。UDP首部长度是固定的。

1. **设TCP使用的最大窗口为65535字节，而传输信道不产生差错，带宽也不受限制。若报文段的平均往返时延为20ms，问所能得到的最大吞吐量是多少？**

答：在发送时延可忽略的情况下，最大数据率=最大窗口\*8/平均往返时间=26.2Mb/s。

1. **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。**

**解答：**

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数据段，这次很不幸，主机2在收到第二个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的连接释放方法，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，而仍然可接受主机1发来的数据，所以可保证不丢失数据。

1. **简述TCP可靠数据传输的过程（即可靠传输机制或原理）。**

**解答：**

TCP 采用了序列号、确认、滑动窗口协议等机制来保证可靠的数据传输。

首先，TCP 要为所发送的每一个分段加上序列号，保证每一个分段能被接收方接收，并只被正确地接收一次。

其次，TCP 采用具有重传功能的积极确认技术作为可靠数据流传输服务的基础。这里，"确认"是指接收端在正确收到分段之后向发送端回送一个确认（ACK）信息。发送方将每个已发送的分段备份在自己的发送缓冲区里，而且在收到相应的确认之前是不会丢弃所保存的分段的。"积极"是指发送方在每一个分段发送完毕的同时启动一个定时器，假如定时器的定时期满而关于分段的确认信息尚未到达，则发送方认为该分段已丢失并主动重发。为了避免由于网络延迟引起迟到的确认和重复的确认，TCP 规定在确认信息中捎带一个分段的序号，使接收方能正确地将分段与确认联系起来。

第三，采用可变长的滑动窗口协议进行流量控制，以防止由于发送端与接收端之间的不匹配而引起数据丢失。TCP 采用可变长的滑动窗口，使得发送端与接收端可根据自己的CPU和数据缓存资源对数据发送和接收能力作出动态调整，从而灵活性更强，也更合理。

1. **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。试分别求出第1次到第15次传输的各拥塞窗口大小。你能说明拥塞控制窗口每一次变化的原因吗？**

**解答：**拥塞窗口大小分别为：1，2，4，8，9，10，11，12，1，2，4，6，7，8，9。

1. **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？“加增倍减”的“乘法减少”和“加法增大”各用在什么情况下？**

**解答：**

慢开始：在主机刚刚开始发送报文段时，先将拥塞窗口 cwnd 设置为一个最大报文段 MSS 的数值，同时设置慢开始阈值ssthresh为某个值（比如64KB或16个MSS）。在每收到一个新的报文段的确认后，将拥塞窗口增加至多一个 MSS 的数值。用这样的方法逐步增大发送端的拥塞窗口 cwnd，可以使分组注入到网络的速率更加合理。

拥塞避免：当拥塞窗口值大于ssthresh时，停止使用慢开始算法而改用拥塞避免算法。拥塞避免算法使发送端的拥塞窗口每经过一个往返时延RTT就增加一个MSS的大小。

快重传算法规定，发送端只要一连收到三个重复的 ACK 即可断定有分组丢失了，就应立即重传丢失的报文段而不必继续等待为该报文段设置的重传计时器的超时。

快恢复算法：

（1） 当发送端收到连续三个重复的 ACK 时，就重新设置慢开始门限 ssthresh。

（2） 与慢开始不同之处是拥塞窗口 cwnd 不是设置为 1，而是设置为 ssthresh + 3 \*MSS。

（3） 若收到的重复的 ACK 为 n 个（n > 3），则将 cwnd 设置为 ssthresh + n \* MSS。

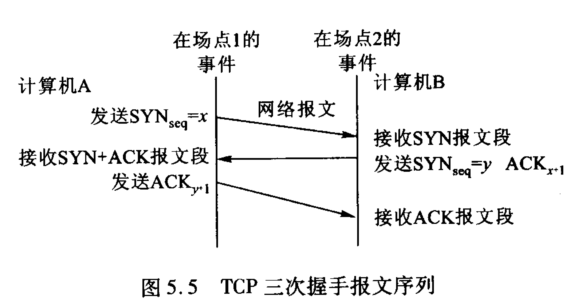
（4） 若发送窗口值还容许发送报文段，就按拥塞避免算法继续发送报文段。

（5） 若收到了确认新的报文段的 ACK，就将 cwnd 缩小到 ssthresh。

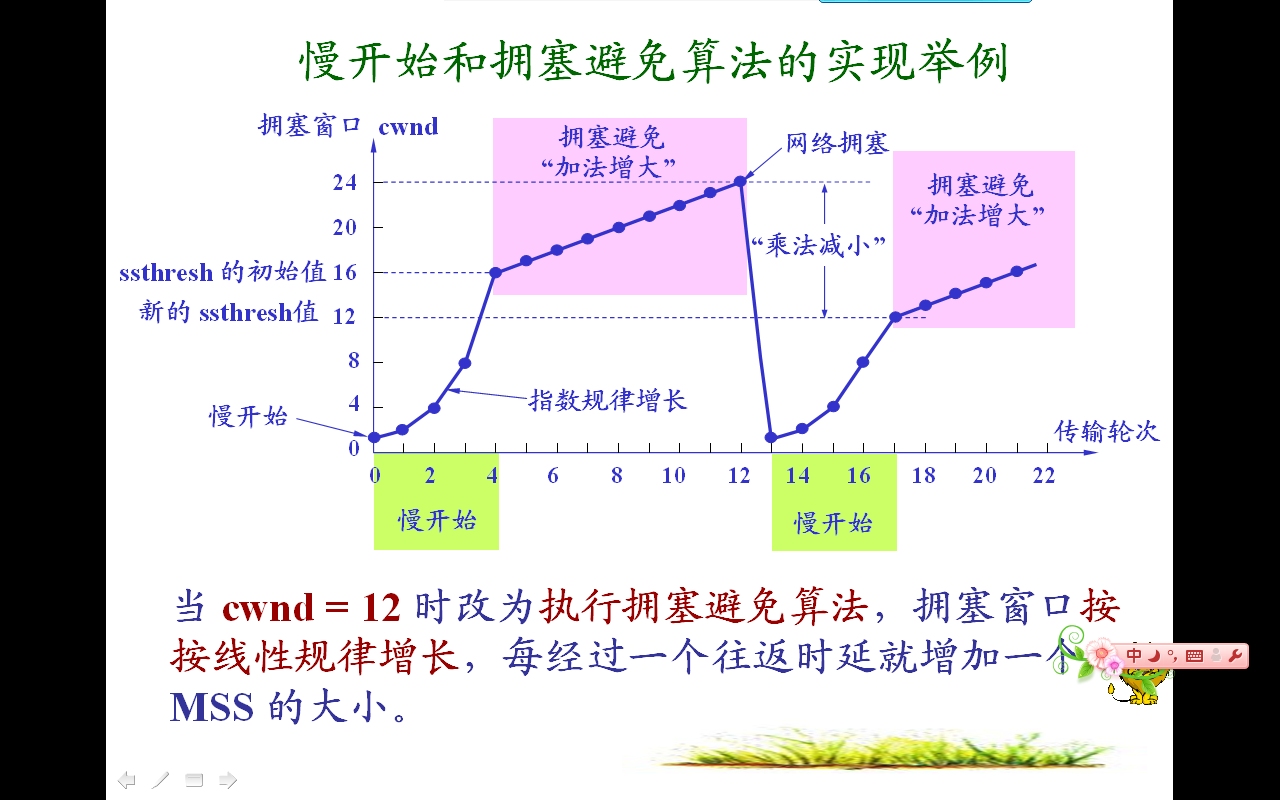
“乘法减小”是指不论在慢开始阶段还是拥塞避免阶段，只要出现一次超时（即出现一次网络拥塞），就把慢开始门限值 ssthresh 设置为当前的拥塞窗口值的一半。当网络频繁出现拥塞时，ssthresh 值就下降得很快，以大大减少注入到网络中的分组数。

“加法增大”是指执行拥塞避免算法后，当收到对所有报文段的确认就将拥塞窗口 cwnd增加一个 MSS 大小，使拥塞窗口缓慢增大，以防止网络过早出现拥塞。

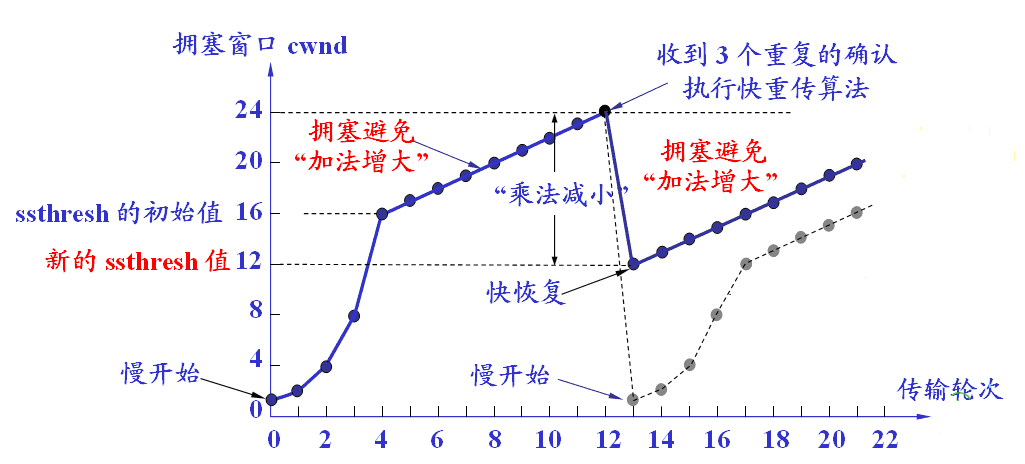
1. **请作图说明TCP三次握手的过程。**



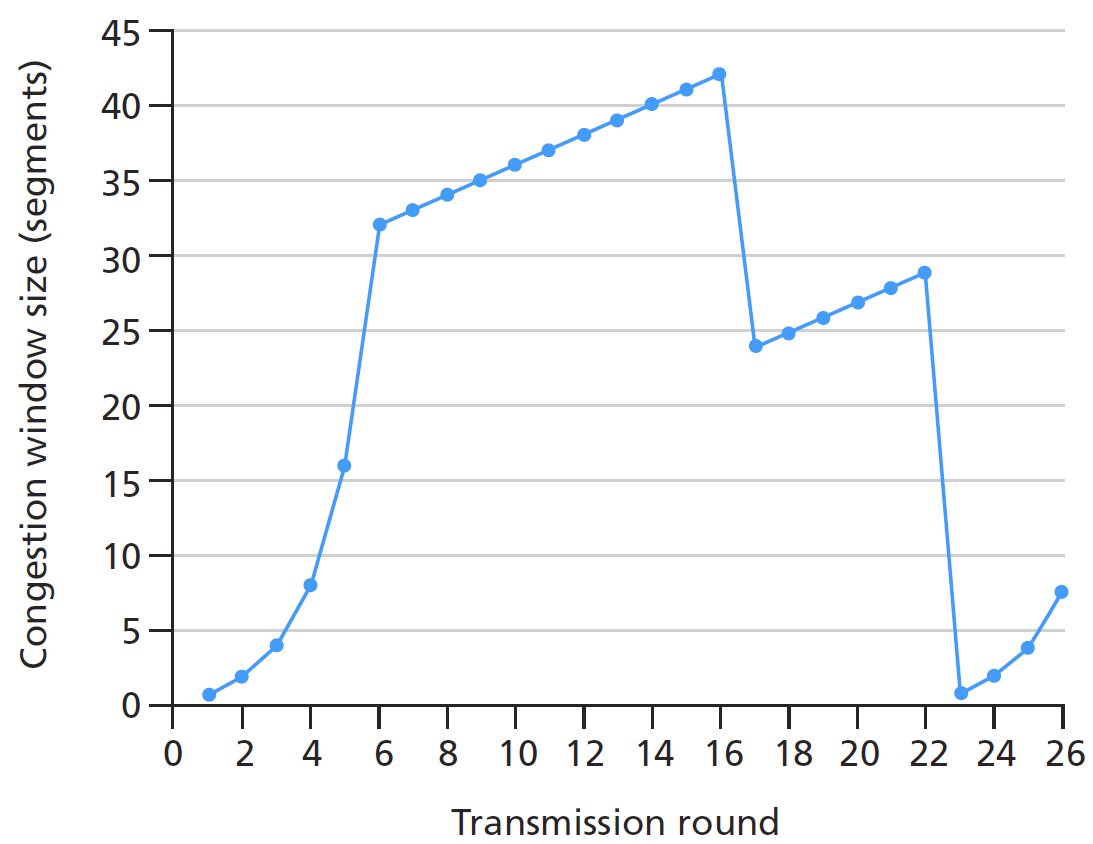
1. **当TCP连接初始化时，把拥塞窗口cwnd置为1，慢开始门限的初始值设置为16。假设当拥塞窗口值为24时，发生拥塞。试运用慢开始和拥塞避免算法画出拥塞窗口值与传输轮次的关系曲线。**



1. **当TCP连接初始化时，把拥塞窗口cwnd置为1，慢开始门限的初始值设置为16。假设当拥塞窗口值为24时，发送方连续收到3个重复的确认报文段。试运用慢开始、拥塞避免、快重传和快恢复算法画出拥塞窗口值与传输轮次的关系曲线。**



1. **考虑如下TCP拥塞窗口演化图，回答下列问题。**



1

17

cwnd = 32

cwnd = 42

cwnd = 29

cwnd = 8

（1）指出TCP慢启动运行时的时间范围。

（2）指出TCP拥塞避免运行时的时间范围。

（3）在第16个传输轮回之后，报文段的丢失是由于检测到三个冗余的ACK还是超时？为什么？此后，在第17个传输轮回，拥塞窗口大小应该为多少？

（4）在第22个传输轮回之后，报文段的丢失是由于检测到三个冗余的ACK还是超时？为什么？

（5）在第1个传输轮回里，ssthreshold的初始值是多少？给出计算理由。

（6）在第18个传输轮回里，ssthreshold的值是多少？给出计算理由。

（7）假定在第26个传输轮回后，通过收到3个冗余的ACK检测到有分组丢失，则此后拥塞窗口cwnd和ssthreshold的值是多少？

**解答：**

（1）TCP慢启动运行的时间范围是[1, 6] 和 [23, 26]。

（2）TCP拥塞避免运行的时间范围是[6, 16] 和 [17, 22]。

（3）在第16个传输轮回之后，报文段的丢失是由于检测到三个冗余的ACK。因为如果是因超时而丢失报文，则拥塞窗口大小 cwnd = 1。此后，在第17个传输轮回，拥塞窗口 cwnd = 42/2 + 3 = 24。

（4）在第22个传输轮回之后，报文段的丢失是由于检测到了超时，因为随后拥塞窗口大小 cwnd = 1。

（5）在第1个传输轮回，ssthreshold的初始值是32。因为从慢启动到拥塞窗口cwnd=32后即开始进入拥塞避免阶段。

（6）在第18个传输轮回，ssthreshold的值是21。因为当检测到包丢失时，TCP将阈值ssthreshold设置为拥塞窗口大小的一半（cwnd/2）。在第16个传输轮回，拥塞窗口大小为42，此后出现丢包，故在第18个传输轮回，ssthreshold的值为 42/2 = 21。

（7）在第26个传输轮回，因为收到3个冗余的ACK检测到有分组丢失，则阈值降低为当前拥塞窗口的一半，即ssthreshold = cwnd/2 = 8/2 = 4；拥塞窗口随后更新为阈值加上3个MSS，即cwnd = ssthreshold + 3\*MSS = 4 + 3 = 7。

1. **在TCP拥塞控制中，假设第0秒时初始拥塞窗口为1，阀值为4，RTT为1秒，在第4秒时出现超时，在第8秒时出现三次重复的确认。则其图形表示可参考下图（因题中参数设置问题，可能有两种不同的画法）。**

