# Chapter 3

R5、由于大多数防火墙都被配置为阻止 UDP 通信，因此使用 TCP 进行视频和语音通信可以让通信通过防火墙。

R8、对于每个持久连接，Web 服务器创建一个单独的“连接套接字”。每个连接套接字被标识为具有四个元组:(源 IP 地址、源端口号、目标 IP 地址、目标端口号)。当主机 C 接收到 IP 数据报，它会检查数据报/段中的这四个字段确定哪个套接字应该通过 TCP 段的有效负载。因此，来自 A 和 B 的请求会具有不同的套接字，这两个参数的标识符用于目标端口的套接字都是 80。但是，这些套接字的标识符源 IP 地址的不同值。与 UDP 不同，传输层通过时 TCP 段对应用程序进程的有效负载，它不指定源 IP 地址，因为这是由套接字标识符隐式指定的。

R14、错、错、对、错、对、错、错

P1、(1) 源端口号：467　　目的端口号：23

(2) 源端口号：513　目的端口号：23

(3) 源端口号：23　　目的端口号：467

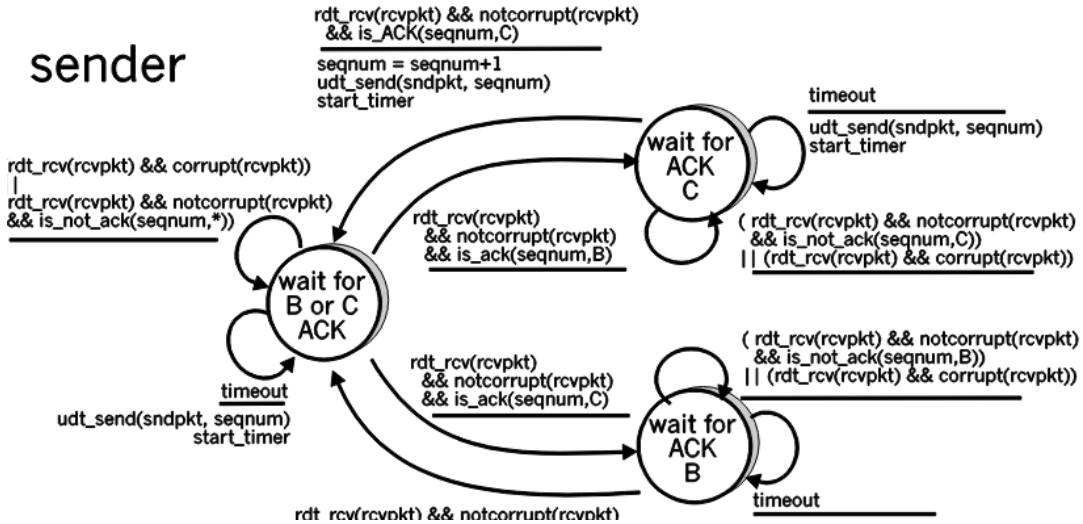
(4) 源端口号：23　　目的端口号：513

(5) 是

(6) 否

P19、本题是在简单停止和等待协议(rdt3.0)上的一个变化。由于信道可能丢失消息，并且由于发送方可能重新发送其中一个接收方已经接收到的消息（要么由于过早超时，要么由于另一个接收方尚未正确接收数据），因此需要序列号。

发送方和接收方 FSM 如下图所示。在此问题中，发送方状态指示发送方是否从 B、从 C 或从非 C 或 B 接收到 ACK。接收方状态指示接收方正在等待哪个序列号。




P27、

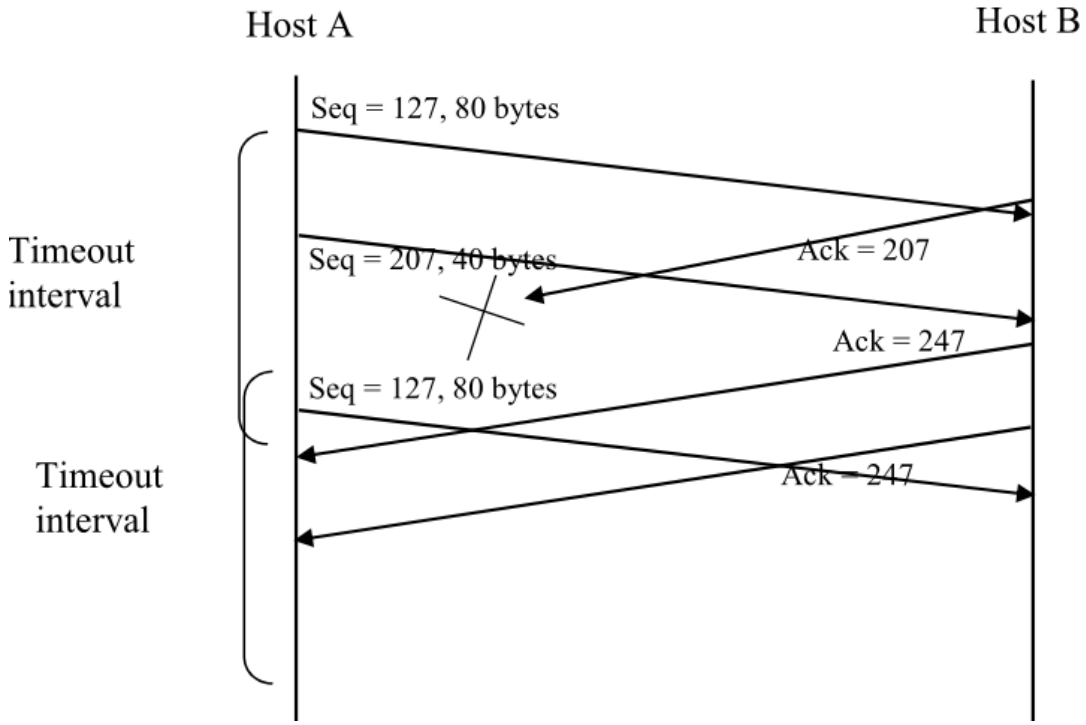
a) 在从主机 A 到 B 的第二段中，序列号为 207，源端口号为 302，目的端口号为 80。

b) 如果所述第一段在所述第二到达段之前到达，则在所述第一到达段的确认中，

所述确认号为 207，所述源端口号为 80，所述目的端口号为 302。

c) 如果第二段到达第一段之前，在第一个到达段的确认中，确认号为 127，表示它仍在等待字节 127 和以后。

d) 如图所示：




P32、

a) 设$\text{EstimatedRTT}^{(n)}$表示第 n 个样本后估计的 RTT 值。

则 $\text{EstimatedRTT}(4)=x\text{SampleRTT1}+(1-x)[x\text{SampleRTT2}+(1-x)[x\text{SampleRTT3}+(1-x)\text{SampleRTT4}]]=x\text{SampleRTT1}+(1-x)x\text{SampleRTT2}+(1-x)2x\text{SampleRTT3}+(1-x)3x\text{SampleRTT4}$

b) $\text{EstimatedRTT}_{(n)} = x\sum_{j=1}^{n-1}(1-x)j\text{SampleRTTj} -1 + (1-x)_{n-1}\text{SampleRTT}_n$

c) $\mathrm{EstimatedRTT}(\infty)=(1-x)/x\sum_{j=1}^{n-1}(1\text{-}x)j\mathrm{SampleRTTj}=1/9\sum_{j=1}^{\infty}9j\cdot\mathrm{SampleRTTj}$

P40、

a) TCP 在[1,6]和[23,26]的间隔缓慢启动；

b) TCP 在间隔[6,16]和[17,22]运行避免拥塞；

c) 第 16 次传输后，数据包丢失由三重重复 ACK 识别。如果超时，则拥塞窗口大小将下降到 1。

d) 第 22 次传输后，由于超时而检测到段丢失，因此，拥塞窗口大小设置为 1。

e ) 阈值最初是 32，因为它处于慢启动停止的窗口大小，并且拥塞避免开始。

f) 当分组丢失时阈值被设置为拥塞窗口的值的一半。当在传输循环 16 期间检测到丢失时，拥塞 Windows 的大小为 42。因此在第 18 传输循环期间阈值为 21。

g) 当分组丢失时阈值被设置为拥塞窗口的值的一半。当在传输循环 22 期间检测到丢失时，拥塞 Windows 的大小为 29。因此，在第 24 轮传输中阈值为 14（取下限为 14.5）。

h) 在第 1 次传输期间，数据包 1 被发送；数据包 2-3 在第 2 次传输中发送；数据包 4-7 在第 3 传输中发送；数据包 8-15 在第 4 个传输中发送；数据包 16-31 在第 5 个传输中发送；数据包 32-63 在第 6 个传输中发送；数据包 64-96 被发送在第 7 次传输循环中。因此，在第 7 发送循环中发送分组 70。

i) 阈值将设置为拥塞窗口(8)的当前值的一半。出现丢失和拥塞窗口将设置为新的阈值 value+3MSS。因此，阈值和窗口的新值分别为 4 和 7。

j) 阈值为 21，拥塞窗口大小为 1。

k) 17 次，1 个分组；18 次，2 个分组；19 次，4 个分组；20 次，8 个分组；第 21 轮，16 个分组；第 22 轮，21 个分组。因此，总数是 52。

P46、

a) 设 w 表示在段中测量的最大窗口大小，则 w * MSS/RTT = 10Mbps，如果最大发送速率超过链路容量，数据包将被丢弃容量。因此，我们有 W * 1500 * 8/0.15 = 10 * 10^6，因此 W 约为 125 个片段。

b)当拥挤窗口大小在 W/2 W 之间变化时，平均窗口大小为 0.75W = 94(上限为 93.75)。平均吞吐量为 94 * 1500 * 8/0.15=7.52 Mbps。

c)当数据包丢失时，W 变为 W/2，即 125/2=62。(125-62) * 0.15=9.45 秒，因为 RTT 的数量（此 TCP 连接将其窗口大小从 62 增加到 125）为 63。

P50、C1 与 C2 的主要区别在于 C1 的 RTT 仅为 C2 的一半。因此，C1 在 50ms 后调整其窗口大小，而 C2 在 100ms 后调整其窗口大小。假设无论何时发生丢失事件，C1 在 50ms 后接收，C2 在 100ms 后接收。我们还得到了以下 TCP 简化模型。在每个 RTT 之后，连接决定是否应该增加窗口大小。对于 C1，我们计算前 50ms 链路中的平均总发送速率。如果该速率超过链路容量，则假设 C1 检测丢失并缩小其窗口大小。但是对于 C2，我们计算了前 100ms 链路中的平均总发送速率。如果该速率超过链路容量，则假设 C2 检测丢失并缩小其窗口大小。请注意，在最后 50ms 内的平均发送速率可能高于链路容量，但最后 100ms 内的平均发送速率小于或等于链路容量，那么在这种情况下，我们假设 C1 将经历丢失事件，而 C2 不会。

基于上述假设的窗口大小和发送速率的演变如下表:

| | C1 | | C2 | |
|---|---|---|---|---|
| 时间（秒） | 窗口大小（在下一个 50 毫秒内发送的段数） | 平均数据发送速率（每个段）第二=窗口/0.05） | 窗口尺寸（编号。第页的分段在下一个发送下 100 毫秒） | 平均数据发送速率(段每秒，=Window/0.1) |
| 0 | 10 | 200（在[0-50]秒内） | 10 | 100（在[0-50]秒内） |
| 50 | 5(减小窗口大小作为 avg。在最后 50 毫秒内向链路发送的总速率为 300=200 100) | 100（在[50-5100]秒内） | | 100（在[50-5100]秒内） |
| 100 | 2（减少了窗口大小作为 AVG。总发送至最后链接 50 毫秒为 200=100100） | 40 | 5(减少了窗口大小作为 AVG。总发送至最后链接 100 毫秒为 250==(200100)/2+(100+100)/2 | 50 |
| 150 | 1（减少了窗口大小作为 AVG。总发送至最后链接 50 毫秒为 90=(4050) | 20 | | 50 |
| 200 | 1(不再减少，因为窗口大小已经是 1) | 20 | 2 （减少窗口大小作为 AVG。发送总数截至 2008 年 12 月 31 日的链接在最后 100 毫秒 80= （4020）/2 （5050）/2） | 20 |

| | C1 | | C2 | |
|---|---|---|---|---|
| 250 | 1(不再减少，因为窗口大小已经是 1) | 20 | | 20 |
| 300 | 1(不再减少，因为窗口大小已经是 1) | 20 | 1（减少了窗口大小作为 AVG。总发送至最后链接 100 毫秒为 40==(2020)/2(2020)/2) | 10 |
| 350 | 2 | 40 | | 10 |
| 400 | 1 | 20 | 1 | 10 |
| 450 | 2 | 40 | | 10 |
| 500 | 1（减少了窗口大小作为 AVG。总发送上次连接到链接的速率 50 毫秒为 50=(4010) | 20 | 1 | 10 |
| 550 | 2 | 40 | | 10 |
| 600 | 1 | 20 | 1 | 10 |
| 650 | 2 | 40 | | 10 |
| 700 | 1 | 20 | 1 | 10 |
| 750 | 2 | 40 | | 10 |
| 800 | 1 | 20 | 1 | 10 |
| 850 | 2 | 40 | | 10 |
| 900 | 1 | 20 | 1 | 10 |
| 950 | 2 | 40 | | 10 |
| 1000 | 1 | 20 | 1 | 10 |

基于上表，我们发现，在 1000ms 后，C1 和 C2 的窗口大小均为 1 段。

b) 在长期运行中，C1 的带宽份额大约是 C2 的两倍，因为 C1 具有较短的 RTT，仅是 C2 的一半，因此 C1 可以将其窗口大小调整为 C2 的两倍。如果我们看上面

的表，我们可以每隔 200ms 看到一个周期，例如，850ms 至 1000ms（含）。在一个周期内，C1 的发送速率(40+20+40+20)=120，它是由发送 C2 给出的三倍大(10+10+10+10)=40。

P52、设 W 表示最大窗口大小。首先，我们可以找到在 TCP 将其窗口大小从 W/2 更改为 W 期间发送的段总数。这通过以下方式给出：

$$S = W/2 + (W/2) * (1+\alpha) + (W/2) * (1+\alpha)2 + (W/2) * (1+\alpha)3 + \ldots + (W/2) * (1+\alpha)k$$

我们发现$k = \log(1+\alpha)2$，然后$S = W * (2\alpha+1)/(2\alpha)$。

损耗率 L 由下式给出：

$$L = 1/S = (2\alpha)/(W * (2\alpha+1))$$

TCP 用于将其窗口大小从 W/2 增加到 W 的时间由下式给出：

$$k * RTT = (\log(1\alpha)2) * RTT,$$

这显然与 TCP 的平均吞吐量无关。

注意，TCP 的平均吞吐量由下式给出：

$$B = MSS * S/((k+1) * RTT) = MSS/(L * (k+1) * RTT)。$$

请注意，这与具有平均吞吐量的 TCP 不同： $B = RTT \cdot L1.22 \cdot MSS$，其中 L 的平方根出现在分母中。