计算机网络　第六周作业　10 月 21 日　周三

PB18151866 龚小航

P3.14 考虑一种仅使用否定确认的可靠数据传输协议。假定发送方只是偶尔发送数据。只用 NAK 的协议是否会比使用 ACK 的协议更好？为什么？现在我们假设发送方要发送大量的数据，并且该端到端连接很少丢包。在第二种情况下，只用 NAK 的协议是否会比使用 ACK 的协议更好？为什么？

解：如果只使用否定确认，那么只有在接收方发现确认了未收到某个分组才会向发送方发送否定确认信号。如果发送方只是偶尔发送数据，那么设想这样一种情况：某次发送方发送了一个分组，但该分组在传输过程中丢失。此时接收方没有收到任何消息，会认为发送方没有发送分组；发送方自然也收不到 NAK 信号，也会认为该分组已经传输完毕。这种情况直到下一次传输成功时，接收方才能通过分组编号发现丢失的分组，从而发送 NAK 信号。在偶尔发送的情况下，分组丢失需要很长的时间才能发现。这时若采用 ACK 协议，发送方在一段时间后未收到肯定确认，会自动重传分组，此时 ACK 比 NAK 协议更为合适。

若发送方要发送大量数据且端到端连接很少丢包，此时采用 NAK 协议能很快的发现分组丢失，且由于很低的丢包率 NAK 信号也不会频繁传输。相反此时若采用 ACK 则需要传输大量的肯定确认信号，效果就不如 NAK 协议。

P3.23 .考虑 GBN 协议和 SR 协议。假设序号空间的长度为 k 那么为了避免出现图 3-27 中的问题，对于这两种协议中的每一种，允许的发送方窗口最大为多少？

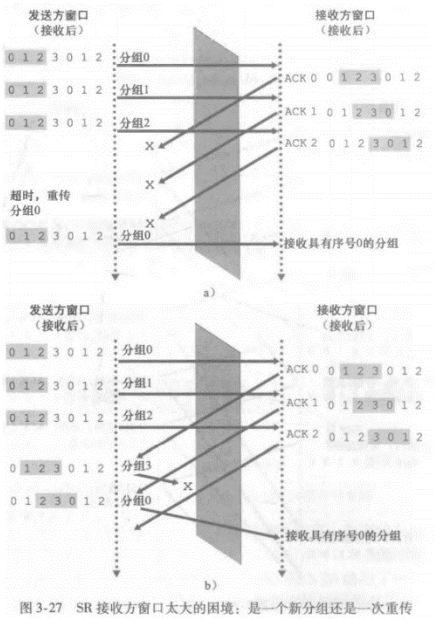

图 3-27　SR 接收方窗口太大的困境：是一个新分组还是一次重传

解：考虑如下图所示的情况。显然窗口空间必须要比序号空间小。假设此时接收方窗口大小为 w，接收端已经接收到序号为 $base-1$ 的包，并等待序号为 $base$ 的包。此时接收方窗口为 $[base, base+w-1]$，对于发送方，此时窗口为 $[base-w, base-1]$。若网络传输时间较长，这 w 个分组都将一起等待来自接收方的 $ACKS$。此时必须令接收器的前缘不与发送者窗口的后缘相重叠，即发送方和接收方两边长度都为 w 的窗口所包含的序号内容不应该出现重复，因此系列号空间应该足够大以至于能容纳 $2w$ 个系列号。因此允许的发送方窗口最大为系列号空间的一半大小。

P3.24 对下面的问题判断是非，并简要地证实你的回答：

a）对于 SR 协议，发送方可能会收到落在其当前窗口之外的分组的 ACK。

b）对于 GBN 协议，发送方可能会收到落在其当前窗口之外的分组的 ACK。

c）当发送方和接收方窗口长度都为 1 时，比特交替协议与 SR 协议相同。

d）当发送方和接收方窗口长度都为 1 时，比特交替协议与 GBN 协议相同。

解： 对其逐条分析：

a）这个说法正确。在发送方的重传超时设定的较小时，假设这样一种情况：发送方当前窗口为 0 1，分组 0 发送，分组 1 发送。接收方正常接收到分组 0，此时返回一个 ACK；而因为发送方重传超时较小，此时 ACK 还未传回发送方就触发了重传机制，导致分组 0 重传。此时假设所有传输都是正常的， $ACK0$ 被发送方接收到以后窗口向前滑动一格，为 1 2，而接收方又会为了重传的分组 0 再次发送一个 $ACK0$，当这个 ACK 到达发送方时就是落在当前窗口之外的分组的 ACK。

b）这个说法正确。和上一问相同，GBN 协议在重传超时设定的较小时也会出现上述情况，只不过会把 0 1两个分组一起全部重传。

c）这个说法正确。当发送方和接收方窗口大小都为 1 时，等同于没有采用流水线技术，而 GBN 和 SR 都是比特交替协议的流水线型效率改进。因此这时它们是完全等同的。

d）这个说法正确。当发送方和接收方窗口大小都为 1 时，等同于没有采用流水线技术，而 GBN 和 SR 都是比特交替协议的流水线型效率改进。因此这时它们是完全等同的。

P3.31 假设测量的 5 个 SampleRTT 值（参见 3.5.3 节）是 106ms、120ms、140ms、90ms 和 115ms. 在获得了每个SampleRTT值后计算EstimatedRTT，使用 $\alpha = 0.125$ 并且假设在刚获得前 5 个样本之后 EstimatedRTT 的值为 100ms . 在获得每个样本之后，也计算 DevRTT，假设 $\beta = 0.25$,并且假设在刚获得前 5 个样本之后 DevRTT 的值为 5ms。最后，在获得这些样本之后计算 TCP Timeoutinterval.

解： 【教材 158 页】先写出 EstimatedRTT 和 DevRTT 以及 Timeoutinterval 的计算式：

$$\text{EstimatedRTT} = (1-\alpha)\cdot\text{EstimatedRTT} + \alpha\cdot\text{SampleRTT}$$

$$\text{DevRTT} = (1-\beta)\cdot\text{DevRTT} + \beta\cdot|\text{EstimatedRTT} - \text{SampleRTT}|$$

$$\text{Timeoutinterval} = \text{EstimatedRTT} + 4 \cdot \text{DevRTT}$$

再分别计算出每次获得 SampleRTT 值后的情况：初始时 $\text{EstimatedRTT} = 100\ \text{ms}$，$\text{DevRTT} = 5\ \text{ms}$

① $\text{SampleRTT} = 106\text{ms}$

$\text{EstimatedRTT} = 0.875 \cdot 100\ \text{ms} + 0.125 \cdot 106\ \text{ms} = 100.75\ \text{ms}$

$\text{DevRTT} = 0.75 \cdot 5\ \text{ms} + 0.25 \cdot |100.75 - 106| = 5.0625\ \text{ms}$

$\text{Timeoutinterval} = 100.75 + 4 \cdot 5.0625 = 121\ \text{ms}$

② $\text{SampleRTT} = 120\text{ms}$

$\text{EstimatedRTT} = 0.875 \cdot 100.75\ \text{ms} + 0.125 \cdot 120\ \text{ms} = 103.15625\ \text{ms}$

$\text{DevRTT} = 0.75 \cdot 5.0625\ \text{ms} + 0.25 \cdot |103.15625 - 120| = 8.0078125\ \text{ms}$

$\text{Timeoutinterval} = 103.15625 + 4 \cdot 8.0078125 = 135.1875\ \text{ms}$

③ $\text{SampleRTT} = 140\text{ms}$

$\text{EstimatedRTT} = 0.875 \cdot 103.15625\ \text{ms} + 0.125 \cdot 140\ \text{ms} \approx 107.762\ \text{ms}$

$\text{DevRTT} = 0.75 \cdot 8.0078125\ \text{ms} + 0.25 \cdot |107.76 - 140| \approx 14.065\ \text{ms}$

$\text{Timeoutinterval} = 107.762 + 4 \cdot 14.065 = 164.022\ \text{ms}$

④ $\text{SampleRTT} = 90\text{ms}$

$\text{EstimatedRTT} = 0.875 \cdot 107.762\ \text{ms} + 0.125 \cdot 90\ \text{ms} \approx 105.542\ \text{ms}$

$\text{DevRTT} = 0.75 \cdot 14.065\ \text{ms} + 0.25 \cdot |105.542 - 90| \approx 14.434\ \text{ms}$

$\text{Timeoutinterval} = 105.542 + 4 \cdot 14.434 = 163.278\ \text{ms}$

⑤ $\text{SampleRTT} = 115\text{ms}$

$\text{EstimatedRTT} = 0.875 \cdot 105.542\ \text{ms} + 0.125 \cdot 115\ \text{ms} \approx 106.724\ \text{ms}$

$\text{DevRTT} = 0.75 \cdot 14.434\ \text{ms} + 0.25 \cdot |106.724 - 115| \approx 12.895\ \text{ms}$

$\text{Timeoutinterval} = 106.724 + 4 \cdot 12.895 = 158.304\ \text{ms}$

P3.36 在 3.5.4 节中，我们看到 TCP 直到收到 3 个冗余 ACK 才执行快速重传。你对 TCP 设计者没有选择在收到对报文段的第一个冗余 ACK 后就快速重传有何看法?

解： 这是一种取舍与平衡，是一种丢包概率与所需时间共同计算得出的近似最优解。

假设现在有若干个分组需要传输，记这些分组为序号为 $n, n+1, n+2$ ……为说明三次收到同一个分组的ACK才触发重传是合理的，以四个分组为例，发送方 A 发送四个分组给接收方 B，N-1 分组已经成功传输。穷举到达分组的可能组合，计算其概率如下:

A 方发送顺序 N-1，N，N+1，N+2，统计接收方收到的分组组合和发出的$ACK(N)$个数:

N-1，N，N+1，N+2	A 收到 1 个 ACK (N)
N-1，N，N+2，N+1	A 收到 1 个 ACK (N)
N-1，N+1，N，N+2	A 收到 2 个 ACK (N)
N-1，N+1，N+2，N	A 收到 3 个 ACK (N)
N-1，N+2，N，N+1	A 收到 2 个 ACK (N)
N-1，N+2，N+1，N	A 收到 3 个 ACK (N)

如果 N 丢了，没有到达 B

N-1，N+1，N+2	A 收到 3 个 ACK (N)
N-1，N+2，N+1	A 收到 3 个 ACK (N)

分析以上数据，可以知道:

如果分组乱序，有 $2/5 = 40\%$ 的概率会造成 A 收到三次 ACK(N);

而如果 N 丢了，则会 100% 概率 A 会收到三次 ACK(N);

而如果 A 接收到二次 ACK(N)，则一定是乱序造成的即然是乱序，说明数据都到达了 B，B 的 TCP 负责重新排序而已，不需要再启动快速重传。

因此在这种权衡下，选择接收到 3 次冗余ACK才触发重传机制是一个较佳的选择。若接收到一个或两个冗余ACK可能会过早的触发重传机制。

P3.40 考虑图 3-58。假设 TCP Reno 是一个经历如上所示行为的协议，回答下列问题。在各种情况中，简要地论证你的回答。

a) 指出 TCP 慢启动运行时的时间间隔。

b) 指出 TCP 拥塞避免运行时的时间间隔。

c) 在第 16 个传输轮回之后，报文段的丢失是根据 3 个冗余 ACK 还是根据超时检测出来的?

d) 在第 22 个传输轮回之后，报文段的丢失是根据 3 个冗余 ACK 还是根据超时检测出来的?

e) 在第 1 个传输轮回里，ssthresh 的初始值设置为多少?

f) 在第 18 个传输轮回里，ssthresh 的值设置为多少?

g) 在第 24 个传输轮回里，ssthresh 的值设置为多少?

h) 在哪个传输轮回内发送第 70 个报文段?

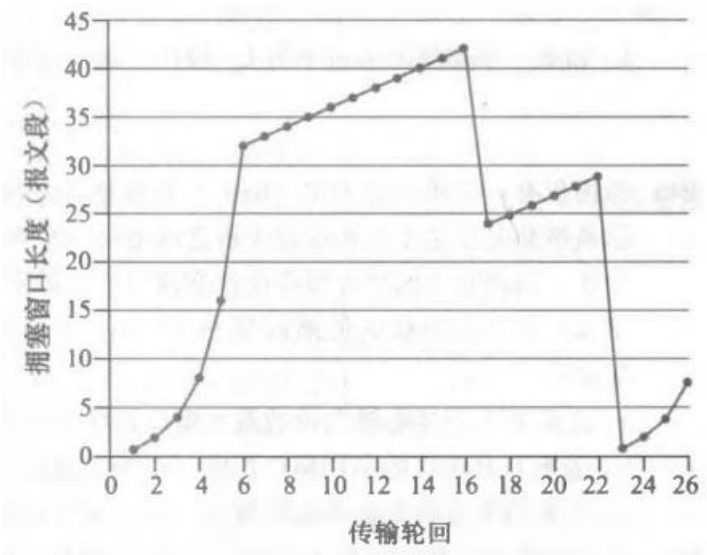

图 3-58 TCP 窗口长度作为时间的函数

i) 假定在第 26 个传输轮回后，通过收到 3 个冗余 ACK 检测出有分组丢失，拥塞的窗口长度和 ssthresh的值应当是多少?

j) 假定使用TCP Tahoe（而不是 TCP Reno）,并假定在第 16 个传输轮回收到 3 个冗余 ACK。在第 19 个传输轮回，ssthresh 和拥塞窗口长度是什么?

k) 再次假设使用TCP Tahoe, 在第 22 个传输轮回有一个超时事件。从第 17 个传输轮回到第 22 个传输轮回（包括这两个传输轮回），一共发送了多少分组?

解：对每问分别分析:

(a) 慢启动时拥塞窗口以指数形式增长。因此图中对应 [1,6], [23,26]

(b) TCP 在执行拥塞避免时，拥塞窗口不再以指数增长，而是线性增长。图中对应 [6,16]. [17,22]

(c) 在第 16 个传输轮回之后，拥塞窗口没有变为 1MSS，因此这是接收到 3 个冗余ACK导致的。

(d) 在第 22 个传输轮回之后，拥塞窗口变为 1MSS，因此这是传输超时导致的。

(e) ssthresh 为慢启动阈值，在第一个轮回周期时，慢启动结束时，它的值为 32

(f) 丢包时阈值将设为拥塞窗口的一半。丢包前拥塞窗口为 42，因此此时阈值为 21

(g) 丢包时阈值将设为拥塞窗口的一半。丢包前拥塞窗口为 29，因此此时阈值为 14

(h) 求散列和即可。$1+2+4+8+16+32=63<70<63+33$，因此在第 7 个传输轮回被发送。

(i) 在第 26 个传输轮回时，拥塞窗口值为 8 个 MSS，因此通过收到 3 个冗余 ACK 检测出有分组丢失时 ssthresh 的值被设置成 0.5 × cwnd = 4 MSS。拥塞窗口长度应当为 4 + 3 = 7 MSS

(*j*) TCP Tahoe 不管是发生超时指示的丢包事件，还是发生 3 个冗余 ACK 指示的丢包事件，都无条件地将拥塞窗口减至 1 个 MSS，并进入慢启动阶段。在第 19 个传输轮回，ssthresh 的值被设置成 21 个 MSS，因为当第 16 个周期丢包事件发生时，拥塞窗口值为 42 个 MSS，所以 ssthresh 的值被设置成 0.5 × cwnd = 21 MSS。但是此时拥塞窗口值为 1MSS

(*k*) 第 17 个传输轮回到第 21 个轮回分别传送了 1 + 2 + 4 + 8 + 16 = 31 个分组，第 22 个轮回传送分组为阈值 21 个，所以一共传送了 52 个分组。

P3.44 考虑从一台主机经一条没有丢包的 TCP 连接向另一台主机发送一个大文件。

a)假定 TCP 使用不具有慢启动的 AIMD 进行拥塞控制。假设每当收到一批 ACK 时，cwnd 增加 1 个 MSS，并假设往返时间大约恒定，cwnd 从 6MSS 增加到 12MSS要花费多长时间(假设无丢包事件)？ .

b)对于该连接，到时间= 6RTT，其平均吞吐量是多少（根据 MSS 和 RTT）？

解：对两问分别分析：

(a) MSS 增长的时间是线性的，因此，从 6MSS 增长到 12MSS 需要 6RTT

(b) 在 $t = 6\text{MSS}$，发送的报文数为5MSS.依次类推：平均吞吐量为：

$$\frac{(6+7+8+9+10+11)\text{MSS}}{6\text{RTT}} = 8.5\text{MSS/RTT}$$