由选择表，这些显示在第三列中。

从邻居接收更新距离向量、重新计算路由选择表项和通知邻居到目的地的最低费用路径的费用已经变化的过程继续下去，直到无更新报文发送为止。在这个时候，因为无更新报文发送，将不会出现进一步的路由选择表计算，该算法将进入静止状态；即所有的结点将执行 DV 算法的第 10 ~ 11 行的等待。该算法停留在静止状态，直到一条链路费用发生改变，如下面所讨论的那样。

1. 距离向量算法：链路费用改变与链路故障

当一个运行 DV 算法的结点检测到从它自己到邻居的链路费用发生变化时（第 10 ~ 11 行），它就更新其距离向量（第 13 ~ 14 行），并且如果最低费用路径的费用发生了变化，向邻居通知其新的距离向量（第 16 ~ 17 行）。图 4-31a 图示了从 y 到 x 的链路费用从 4 变为 1 情况。我们在此只关注 y 与 z 到目的地 x 的距离表中的有关表项。该 DV 算法导致下列事件序列的出现：

- 在 t_0 时刻，y 检测到链路费用变化（费用从 4 变为 1），更新其距离向量，并通知其邻居这个变化，因为最低费用路径的费用已改变。
- 在 t_1 时刻，z 收到来自 y 的更新报文并更新了其距离表。它计算出到 x 的新最低费用（从费用 5 减为费用 2），它向其邻居发送了它的新距离向量。
- 在 t_2 时刻，y 收到来自 z 的更新并更新其距离表。y 的最低费用未变，因此 y 不发送任何报文给 z。该算法进入静止状态。

因此，对于该 DV 算法只需两次迭代就到达了静止状态。在 x 与 y 之间费用减少的好消息通过网络得到了迅速传播。

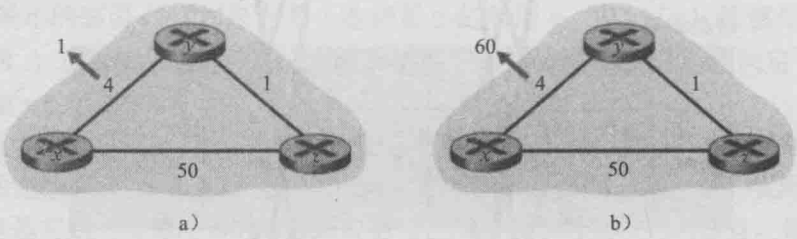

图 4-31 链路费用改变

我们现在考虑一下当某链路费用增加时发生的情况。假设 x 与 y 之间的链路费用从 4 增加到 60，如图 4-31b 所示。

1）在链路费用变化之前，$D_y(x)=4$，$D_y(z)=1$，$D_z(y)=1$ 和 $D_z(x)=5$。在 t_0 时刻，y 检测到链路费用变化（费用从 4 变为 60）。y 计算其到 x 的新的最低费用路径的费用值为

$$D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\} = \min\{60+0, 1+5\} = 6$$

当然，从网络全局的视角来看，我们能够看出经过 z 的这个新费用是错误的。但结点 y 仅有的信息是：它到 x 的直接费用是 60，且 z 上次已告诉 y，z 能以费用 5 到 x。因此，为了到达 x，y 将通过 z 路由，完全期望 z 能以费用 5 到达 x。到了 t_1 时刻，我们遇到**路由选择环路**（routing loop），即为到达 x，y 通过 z 路由，z 又通过 y 路由。路由选择环路就像一个黑洞，即目的地为 x 的分组在 t_1 时刻到达 y 或 z 后，将在这两个结点之间不停地（或直到转发表发生改变为止）来回反复。

2）因为结点 y 已算出到 x 的新的最低费用，它在 t_1 时刻将该新距离向量通知 z。