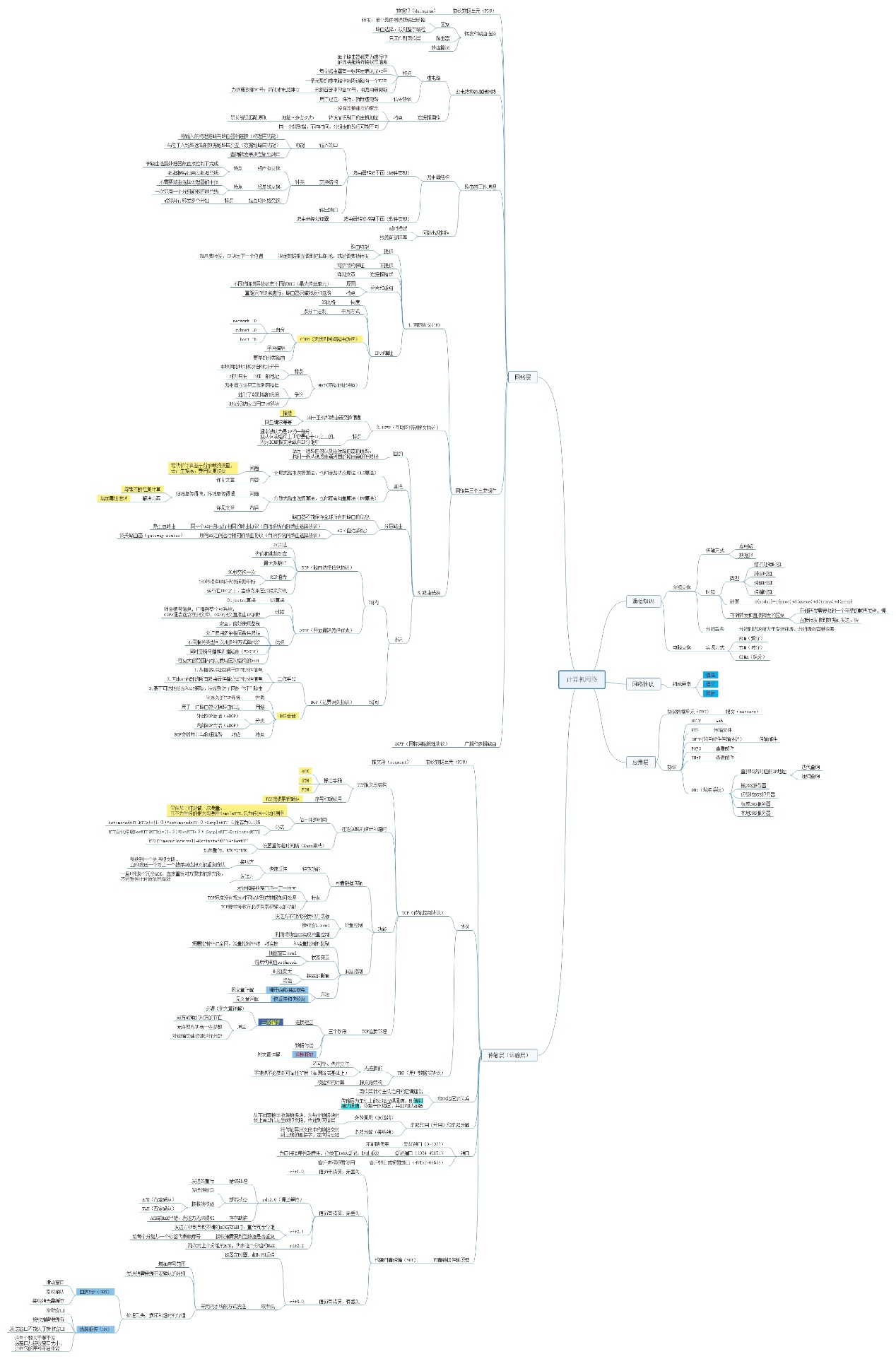
计网复习笔记3-2



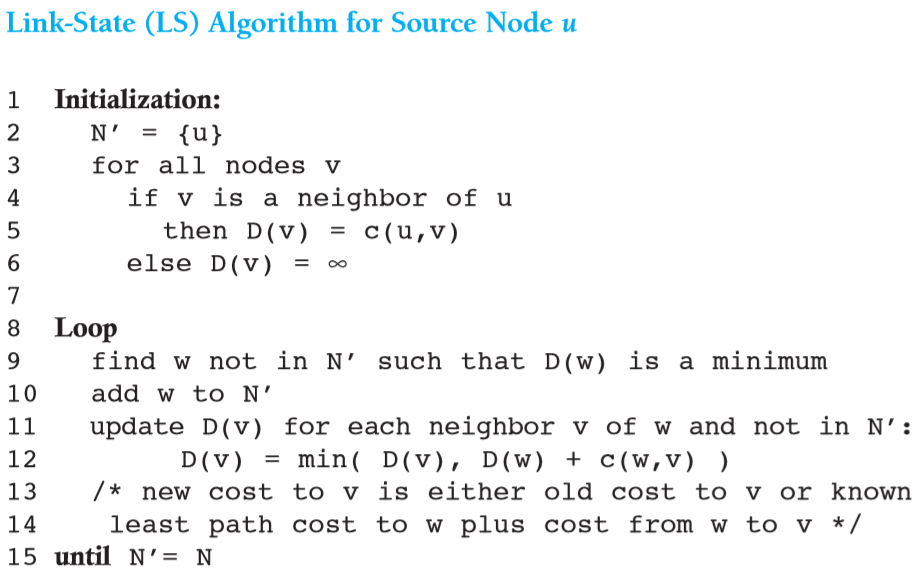
路由选择算法是网络层的重要内容，专门拿出一节讲解路由选择算法。

LS算法中的典型代表即为Dijkstra算法。

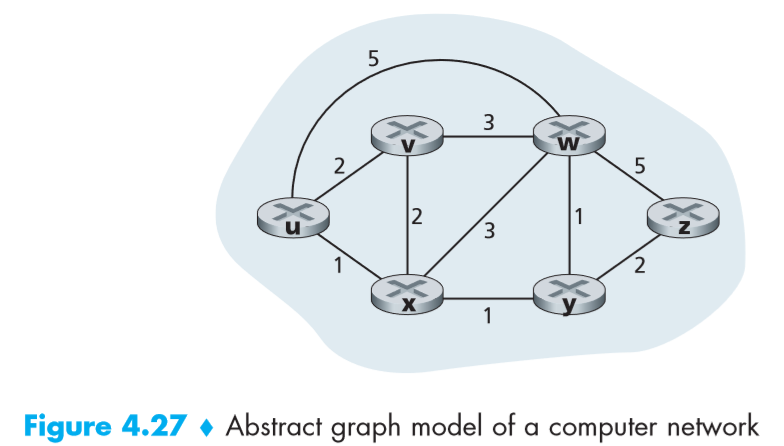
Dijkstra算法是迭代算法，其性质是经算法的第k次迭代后，可知道到k个目的结点的最低费用路径，在到所有目的结点的最低费用路径之中，这k条路径具有k个最低费用。我们定义下列记号。

* D(v)：到算法的本次迭代，从源结点到目的结点v的最低费用路径的费用。
* p(v)：从源到v沿着当前最低费用路径的前一结点（v的邻居）。
* N’：结点子集；如果从源到v的最低费用已确知，v在N’中。

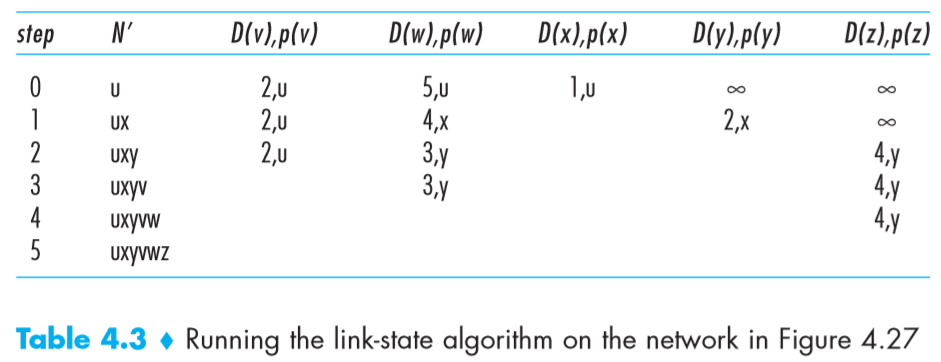
该全局路由选择算法由一个初始化步骤和其后的循环组成。循环执行的次数与网络中结点个数相同，一旦终止，该算法就计算出了从源结点u到网络中每个其他结点的最短路径。



举一个例子，考虑下图中的网络，计算从u到所有可能目的地的最短费用路径。

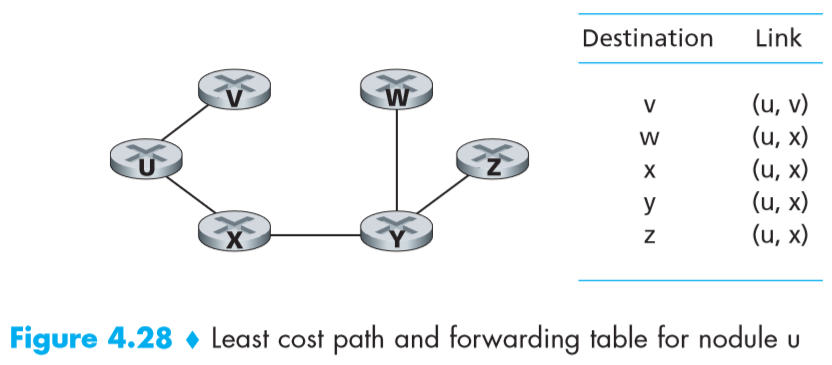


该算法的计算过程以表格的方式总结于下表中，表中的每一行给出了迭代结束时该算法的变量的值。我们详细地考虑前几个步骤。



* 在初始化步骤，从u到与其相连的邻居v、x、w的当前已知最低费用路径分别初始化为2、1和5.特别值得注意的是，到w的费用被设为5（尽管我们很快就会看见一条费用更小的路径确实存在），因为这是从u到w的直接（一跳）链路费用。到y与z的费用被设为无穷大，因为它们不直接与u连接。
* 在第一次迭代中，我们观察那些还未加到集合N’中的结点，并且找出在前一次迭代结束时具有最低费用的结点。那个结点便是x，其费用是1，因此x被加到集合N’中。于是LS算法中的第12行中的程序被执行，以更新所有结点v的D(v)，产生表中第2行（步骤1）所示的结果。到v的路径费用未变。经过结点x到w（在初始化结束时其费用为5）的路径费用被发现为4.因此这条具有更低费用的路径选中，且沿从u开始的最短路径上w’的前一结点被设为x。类似地，到y（经过x）的费用被计算为2，且该表也被相应地更新。
* 在第二次迭代时，结点v与y被发现具有最低费用路径（2），并且我们任意改变次序将y加到集合N’中，使得N’中含有u、x和y。到仍不在N’中的其余结点（即结点v、w和z）的费用通过LS算法中的第12行进行更新，产生如表中第3行所示的结果。
* 如此等等

当LS算法终止时，对于每个结点，我们都得到从源结点沿着它的最低费用路径的前一结点。对于每个前一结点，我们又有它的前一结点，以此方式我们可以构建从源结点到所有目的结点的完整路径。通过对每个目的结点存放从u到目的地的最低费用路径上的下一跳结点，在一个结点（如结点u）中的转发表则能够根据此信息而构建。下图显示了对于图4-27中的网络产生的最低费用路径和u中的转发表。



距离向量（DV）算法是一种迭代的、异步的和分布式的算法，而LS算法是一种使用全局信息的算法。说它是分布式的，是因为每个结点都要从一个或多个直接相连邻居接收某些信息，执行计算，然后将其计算结果分发给邻居。说它是迭代的，是因为此过程一直要持续到邻居之间无更多信息要交换为止。说它是异步的，是因为它不要求所有结点相互之间步伐一致地操作。

在给出DV算法之前，有必要讨论一下存在于最低费用路径的费用之间的一种重要关系。令dx(y)是从结点x到结点y的最低费用路径的费用。则该最低费用与著名的Bellman-Ford方程相关，即

dx(y)=minv{c(x,v)+dv(y)}

方程中的minv是对于x的所有邻居的。

Bellman-Ford方程不止是一种智力上的珍品，它实际上具有重大的实践重要性。特别是对Bellman-Ford方程的解为结点x的转发表提供了表项。为了理解这一点，令v\*是取得方程中最小值的任何相邻结点。接下来，如果结点x要沿着最低费用路径向结点y发送一个分组，它应当首先向结点v\*转发该分组。因此，结点x的转发表将指定结点v\*作为最终目的地y的下一跳路由器。Bellman-Ford方程的另一个重要实际贡献是它提出了将在DV算法中发生的邻居到邻居通信的形式。

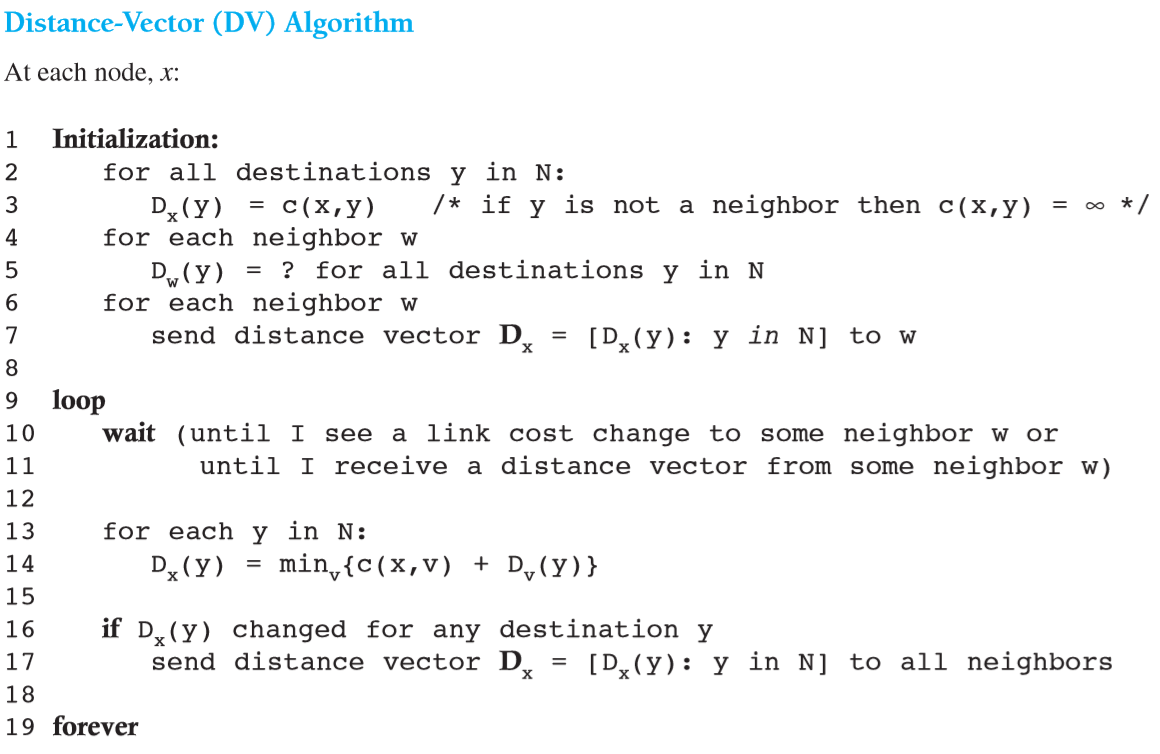
其基本思想如下。每个结点x以Dx(y)开始，对在N中的所有结点，估计从它自己到结点y的最低费用路径的费用。令Dx=[Dx(y):y∈N]是结点 x的距离向量，该向量是从x到在N中的所有其他结点y的费用估计的向量。使用DV算法，每个结点x维护下列路由选择信息：

* 对于每个邻居v，从x到之间相连邻居v的费用为c(x,v)。
* 结点x的距离向量，即Dx=[Dx(y):y∈N]，包含了x到N中所有目的地y的费用的估计值
* 它的每个邻居的距离向量，即对x的每个邻居v，有Dx=[Dx(y):y∈N]。

在该分布式、异步算法中，每个结点不时地向它的每个邻居发送它的距离向量副本。当结点x从它的任何一个邻居v接收到一个新距离向量，它保存v的距离向量，然后使用Bellman-Ford方程更新它自己的距离向量如下：

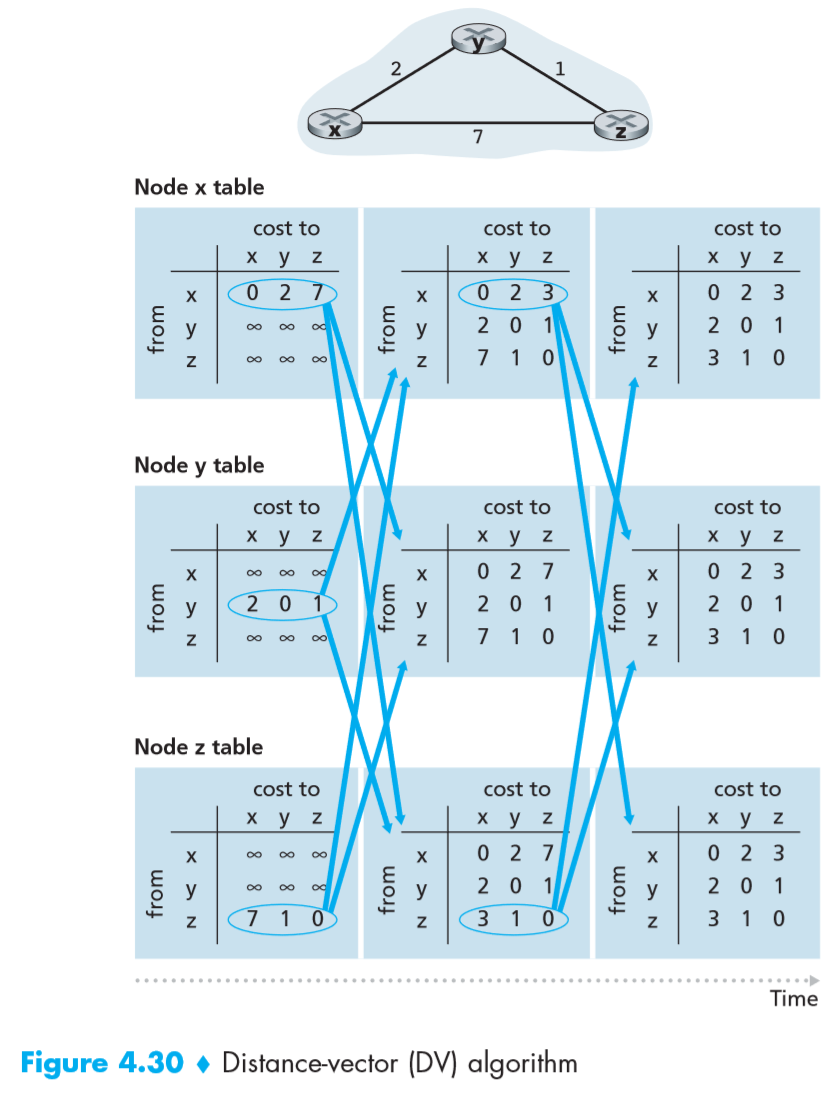
Dx(y)=minc{c(x,v)+Dv(y)} 对N中的每个结点

如果结点x的距离向量因这个更新步骤而改变，结点x接下来将向它的每个邻居发送其更新后的距离向量。令人惊奇的是，只有所有的结点继续以异步方式交换它们的距离向量，每个费用估计Dx(y)收敛到dx(y)，dx(y)为以结点x到结点y的实际最低费用路径的费用！具体的算法过程如下图：



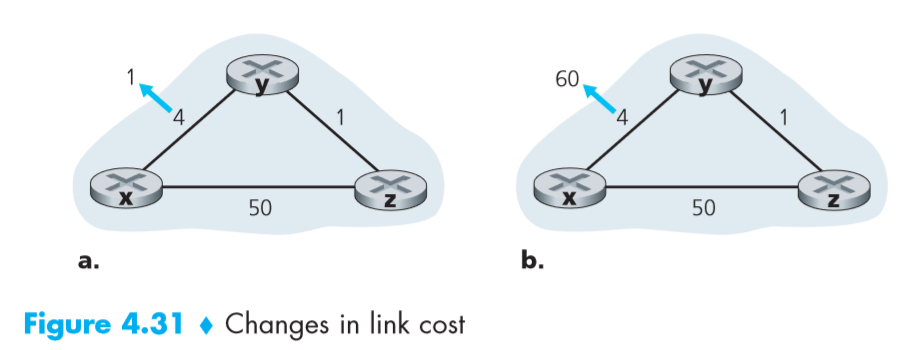
在该DV算法中，当结点x发现它的直接相连的链路费用变化或从某个邻居接收到一个距离向量的更新时，它就更新其距离向量估计。但是为了一个给定目的地而更新它的转发表，结点x真正需要知道的不是到y的最短距离，而是沿着最短路径到y的下一跳路由器邻居结点v\*(y)。

下图举例说明了DV算法的运行，应用场合时该图顶部有三个结点的简单网络。



DV算法的问题：好消息传得快，坏消息传得慢

以下图为例：



图a：从y到x的链路费用从4变为1.我们只关注y与z到目的地x的距离表中的有关表项。该DV算法导致下列实际序列的出现：

* 在t0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t1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t2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。好消息快速得到了传播。

图b：x与y之间的链路费用从4增加到60，导致如下情况：

1. 在链路费用变化之前，Dy(x)=4，Dy(z)=1，Dz(y)=1和Dz(x)=5。在t0时刻，y检测到链路费用变化（费用从4变为60）。y计算到x的新的费用路径的费用值为

Dy(x)=min{c(y,x)+Dx(x),c(y,z)+Dz(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的这个新费用时错误的。但结点y仅有的信息时：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t1时刻，我们遇到路由选择环路，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t1时刻到达y或者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。

1. 因为结点y已算出到x的新的最低费用，它在t1时刻将该新距离向量通知z。
2. 在t1后某个时间，z收到y的新距离向量，它指示了y到x的新最低费用是6.z知道它能以费用1到达y，因此计算出到x的新最低费用Dz(x)=min{50+0,1+6}=7。因为z到x的最低费用已增加了，于是它便在t2时刻通知y其新费用。
3. 以类似方式，在收到z的新距离向量后，y决定Dy(x)=8并向z发送其距离向量。接下来z确定Dz(x)=9并向y发送其距离向量，等等。

循环将持续44次迭代（在y与z之间交换报文），即直到z最终算出它经由y的路径费用大于50位置。

解决方法：增加毒性逆转

刚才描述的特定循环的场景可以通过一种称为毒性逆转的技术而加以避免。其思想较为简单：如果z通过y路由选择到目的地x，则z将通告y，它（z）到x的距离是无穷大，即z将向y通告Dz(x)=∞（即使z实际上直到Dz(x)=5）。只有z经y路由选择到x，z就持续地向y讲述这个善意的谎言。因为y相信z没有到x的路径，故只要z继续经y路由到x（并这样去撒谎），y将永远不会试图经由z路由选择到x。

但对于复杂的网络，毒性逆转依然无法解决问题。