**《计算机网络》课程设计报告**





**路由协议算法实验**

**学 号 3020001267**

**姓 名 王旭**

**学 院 智能与计算学部**

**专 业 软件工程**

**年 级 2020**

**任课教师 张亚平**

**2022年 6 月 10 日**

# 一、报告摘要

1. 加深对距离矢量路由算法的理解。

2. 掌握距离矢量路由算法的具体实现方式和细节。

为图1所示网络结构设计并实现一种分布式的距离矢量路由算法。图中所有链路都是双向的，且两个方向上的距离相同。

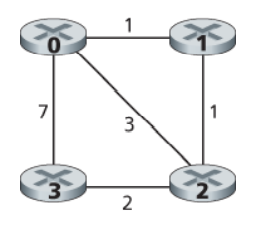


图1 网络结构

本次实验在所提供的网络仿真环境(emulated environment)中实现。网络仿真环境包含了丰富的基础代码：网络仿真的模拟、数据包发送和接收、除路由算法以外的功能代码等。

# 二、任务需求分析

每个节点的路由算法部分留空，需要自行填补，具体包括节点的初始化操作、节点路由表的维护方法和节点接收到路由更新信息的响应等内容。以节点0为例，其它节点类似。对于节点0，需要填补并实现以下函数：

**rtinit0()**：该函数没有参数，在仿真开始时被调用，用于节点0的初始化操作。在该函数中，节点0将根据网络拓扑结构初始化自身的距离表，然后通过调用tolayer2()函数（已定义在prog3.c中，不需要修改），向邻居节点（节点1,2和3）发送自身到网络其他节点的最近距离。

**rtupdate0(struct rtpkt \*rcvdpkt)**：该函数在节点0接收到路由消息包（routing packet）时调用。函数的参数是指向该消息包对应数据结构的指针。在该函数中，节点0根据接收到路由消息包更新自己的路由表。路由表更新后，如果节点0到网络其它节点的最近距离发生了改变，则节点0向邻居节点发送新的自身到网络其它节点的最近距离。

# 三、协议设计

本实验实现的是DV算法。

距离向量（Distance-Vector, DV）算法是一种迭代的、异步的和分布式的算法，说它是分布式的，是因为每个节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。(有趣的是，此算法是自我终 止的，即没有计算应该停止的信号，它就停止了。)说它是异步的，是因为它不要求所有节点相互之间步伐一致地操作。

令是从节点兀到节点的最低开销路径的开销。则有如下公式

其基本思想如下。每个节点x以D\_x(y)开始，对在N中的所有节点y,估计从x到y 的最低开销路径的开销. 令

是节点x的距离向量，该向量是从x到在N中的所有其他节点y的开销估计向量。使用DV算法，每个节点x维护下列路由选择信息

1. 对于每个邻居v，从x到直接相连邻居v的开销为c（x，v）

2. 节点x的距离向量，即

包含了到N中所有目的地y的开销估计值

3. 它的每个邻居的距离向量，即对x的每个邻居v有

在该分布式、异步算法中，每个节点不时地向它的每个邻居发送它的距离向量副本。 当节点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：

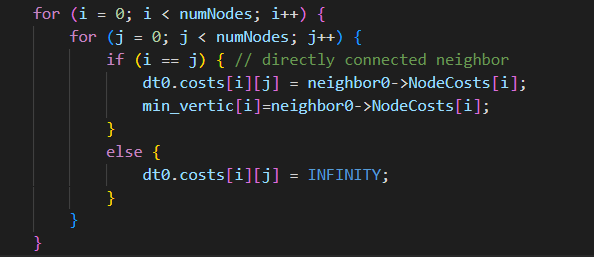
如果节点x的距离向量因这个更新步骤而改变，节点x接下来将向它的每个邻居发送其更新后的距离向量，这继而让所有邻居更新它们自己的距离向量。每个开销估计收敛到, 为从节点x到节点y的实际最低开销路径的开销.

# 四、协议实现

在具体实现中，还是与上面的设计有区别，在本实验中，无需考虑路由转换表的设计，因为本实验只考虑向邻居发送报文这个步骤，而不考虑其他过程。本实验也不保存邻居的距离向量，而是保存一个距离表。

每个节点内的距离表是距离向量算法所使用的主要数据结构，基础代码将其定义为一个4\*4的整型数组（costs）。以节点0为例，距离表的元素costs[i][j]表示当前状态下，节点0经过直接邻居j到节点i的距离。如果0不能直接到j或者目前j无法到i，则用整数值9999（表示“无穷大”）填充该位置。

所以初始化操作就是要初始化距离表和距离向量，以0节点为例，因为初始时只知道当前的到邻居的开销，所以距离表只需要对cost[i][i]填充到邻居i的开销即可，因为cost[i][i]就表示经过i到i的开销，也就是到邻居i的开销，这个信息由getNeighborCosts获取，其它的全部填为极大值也就是9999。再初始化距离向量，直接等于到邻居开销即可。

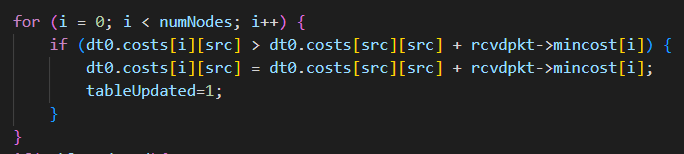


在更新操作中主要就是根据报文是从哪来的进行相应的更新，比如是从src来的报文，那么就更新src这一列的信息。

For each node i:

Cost[i][src]=min{ Cost[i][src],cost[src][src]+srcpkt\_cost[i]}

也就是



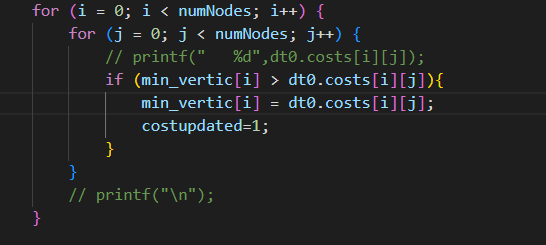
而后在根据更新后的距离表修改距离向量，再将这个距离向量发出去即可。

距离向量就是

For each src node i:

For each 要经过的 node j:

Mincost[i]=min{ Mincost[i],cost[i][j]}

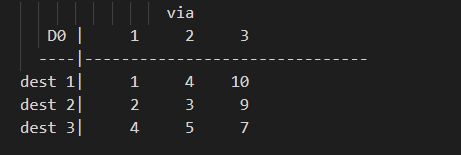


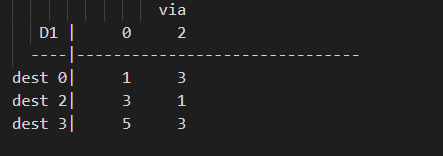
注意还要根据距离向量的更新情况决定是否向邻居发送报文，如果没变化就不发。

# 四、实验结果及分析

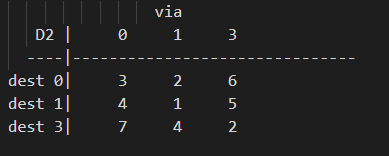
测试所实现协议的功能和性能，并对性能结果进行分析。需要针对考察点逐一展开。

最后D0的距离表，到节点1、2、3的距离分别是1、2、4，分别从1、1、1走。

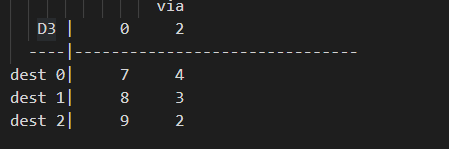




D1的距离表，到0、2、3的距离分别是1、1、3，分别从0、2、2走

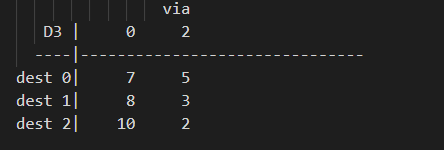


D2的距离表，到0、1、3的距离分别是2、1、2，分别从1、1、3走

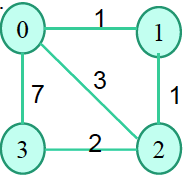


D3的距离表，到0、1、2的距离分别是4、3、2，分别从2、2、2走

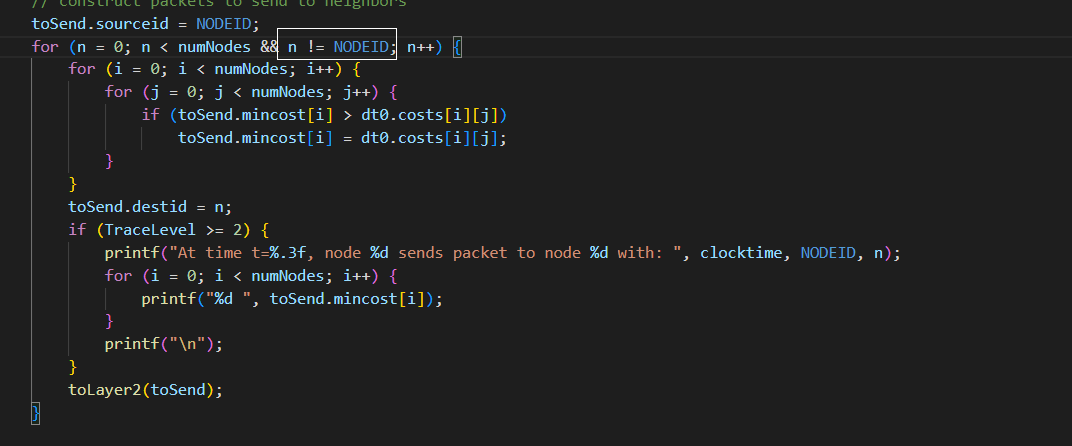
这里值得注意的是，实验一开始提供的代码是有一些问题的，是收敛不到正确结果的，如果用老师提供的程序来跑这个样例，对于做最后的D3



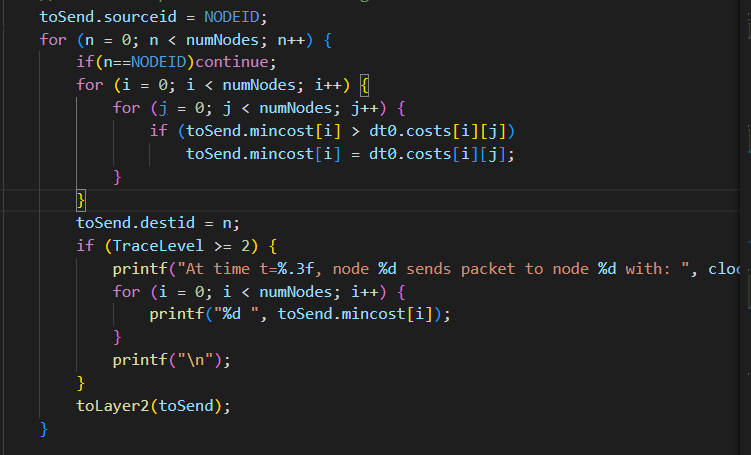
到0、1、2的距离分别是5、3、2

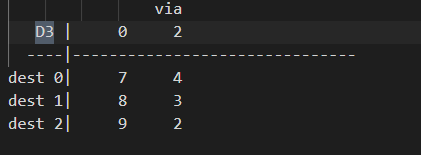


然而从3到0可以走3-2-1-0,开销分别是2、1、1加起来就是4，所以我那个是对的，实验提供的是错的。错误的地方就在于更新距离表后发送距离向量的步骤。



这是一个低级错误，本意应该是对于n如果等于自己，就不发送，但是原图这么写的结果就是当轮到自己时for循环就结束了，导致没有发给所有邻居

应该改为：



改正后结果就正确了。

# 五、总结

这次实验使我收获很多，这是第一次接触这种分布式，异步而非同步的算法，也为这种算法感到惊叹，以往学习的算法都是同步式的，集中式的，现在大趋势都是分布式，所以有必要多学一些这样的知识以满足需求。