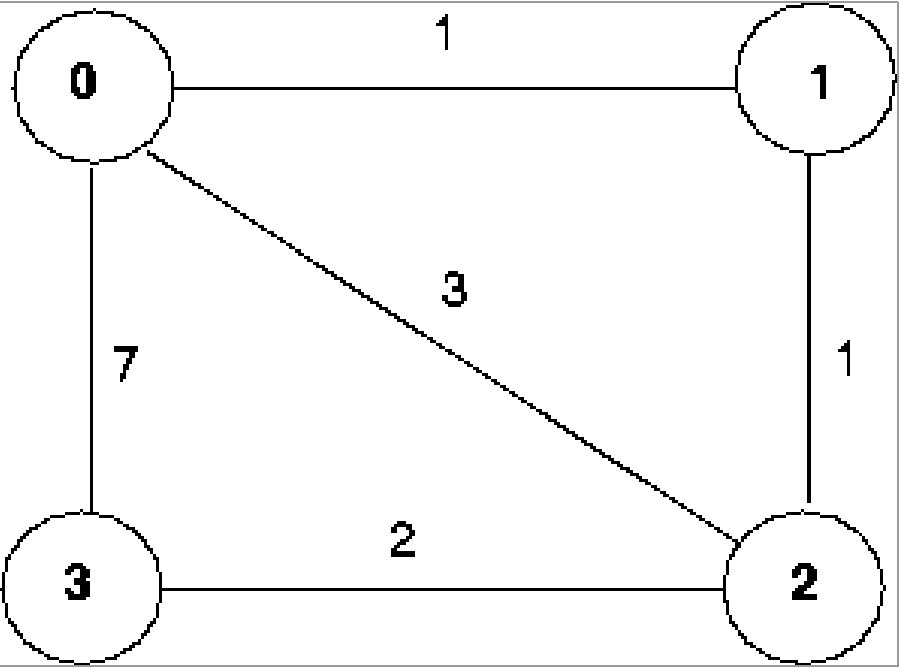
Lab4:RIP路由实验

**Overview**

实现一种分布式、异步的距离向量路由算法， 事先各节点无法

得知整个链路的状态信息，而只能感知到节点本身与相邻节点间的链

路开销，通过路由 packet 的交换，完成各自节点距离向量表的建立与更新。



图一

**The Basic Assignment**

对于节点0，你将编写以下例程：

rtinit0（）此例程将在仿真开始时调用一次。rtinit0（）没有参数。它应该初始化节点0中的距离表，以分别反映到节点1,2,3的成本1,3,7。在图1中，所有链接都是双向的，并且两个方向的成本是相同的。在初始化距离表以及节点0例程所需的任何其他数据结构之后，它应该向其直接连接的邻居（在本例中是1,2以及3）发送所有其他网络节点的最小代价路径的成本。通过调用例程tolayer2（）将该最小成本信息发送到路由分组中的相关节点。如下所述，路由分组的格式也在下面描述。

rtupdate0(struct rtpkt \*rcvdpkt)此例程将在节点0接收到由其直接相连的邻居发送给它的路由数据包时将调用此例程。参数\*rcvdpkt是指向接收到的数据包的指针。

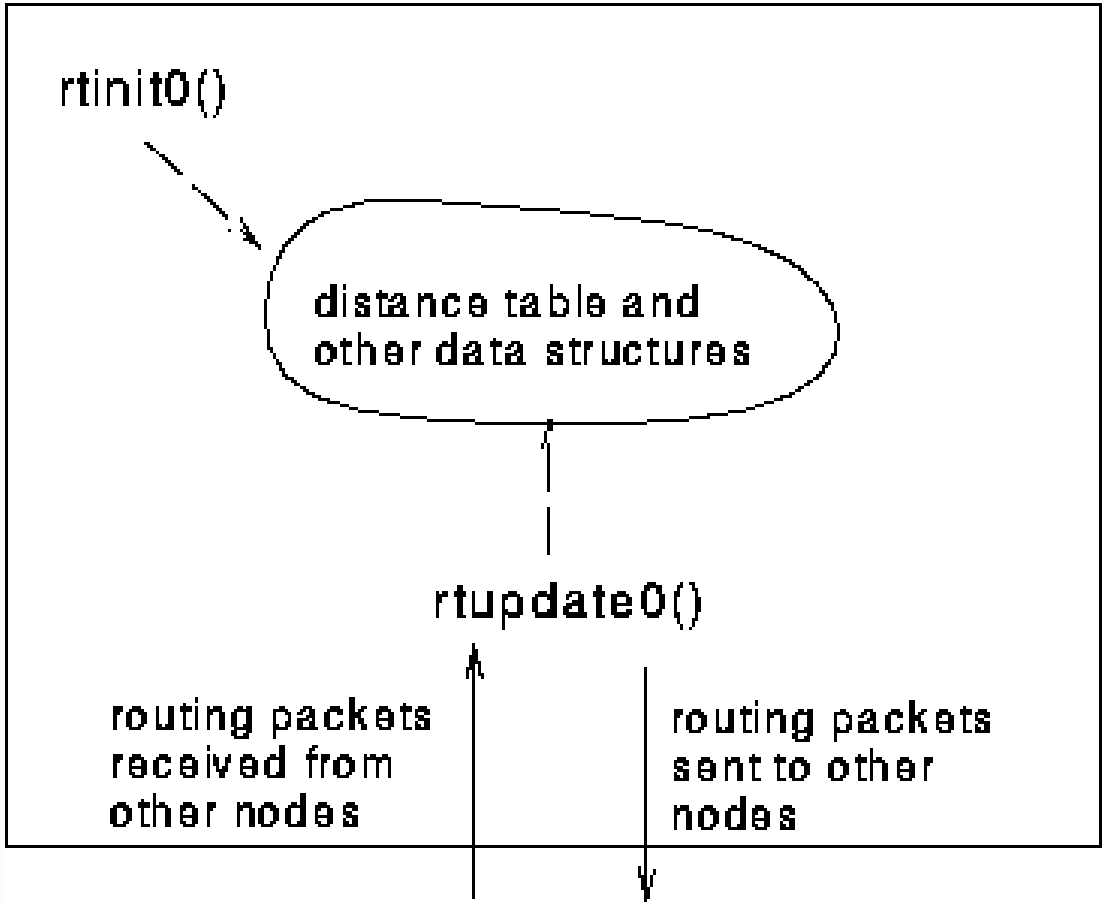
rtupdate0（）是距离向量算法的“心脏”。它在来自其他某个节点i的路由数据包中接收的值包含i到其他网络节点的当前最短路径成本。rtupdate0（）使用这些接收值以更新其自己的距离表（由

距离向量算法）。如果由于更新导致另一节点的最小成本发生变化，节点0通过向其发送一个路由包来向其直接相连的邻居通知该最小成本的变化。

回想一下，在距离向量算法中，只有直接连接的节点将交换路由数据包。因此，节点1和2将与彼此直接通信，但是节点1和节点3将通过一个节点进行通信。每个节点内的距离表是距离向量算法使用的主要数据源。你会发现很方便将距离表声明为一个4乘4的int数组，其中节点0的向量表中[i,j]是节点0通过节点j连接到节点i所需的成本。如果0未直接连接到j，则可以忽略此项。我们将使用整数值999是“无穷大”的约定。

图二提供了节点0内的过程的概念视图。

节点1,2,3定义了相似的例程，因此你总共需要写8个程序：rtinit0(), rtinit1(), rtinit2(), rtinit3(),rtupdate0(), rtupdate1(), rtupdate2(), rtupdate3()



图二

**Software Interfaces**

上面描述的过程就是你要编写的过程，下面为可以调用的例程。

tolayer2(struct rtpkt pkt2send)其中rtpkt结构已经声明了，tolayer2()已经在prog3.c中声明。

extern struct rtpkt {

int sourceid; /\* id of node sending this pkt, 0, 1, 2, or

3 \*/

int destid; /\* id of router to which pkt being sent

int mincost[4]; /\* min cost to node 0 ... 3 \*/

};

注意tolayer2()是一个结构体而不是一个指向结构的指针。

printdt0()将打印节点0的距离向量表。它是一个被传递给一个结构体distance\_table. printdt0()的指针。节点0的距离向量表的结构声明在文件node0.c中。node1.c, node2.c node3.c中也定义了相似的打印例程。

**The simulated network environment**

你的例程rtinit0（）、rtinit1（）、rtinit2（）、rtinit3（）和rtupdate0（），rtupdate1（）、rtupdate2（）、rtupdate3（）发送路由数据包（其格式为如上所述），然后将按顺序传送数据包，并且不会丢失到指定的目的地。只有直接连接的节点才能通信。发送方和接收方之间的延迟是可变的（未知的）。

当你把你的程序和已给的程序一起编译并运行生成的程序将要求您只指定一个有关模拟网络环境的值。

追踪：将tracing值设置为1或2将打印出有用的有关仿真内部正在进行的操作的信息（例如发生在数据包和计时器上）。跟踪值为0将关闭此功能。大于2的跟踪值将显示各种奇怪消息,这是为了我自己的模拟器调试目的。

跟踪值2可能有助于调试代码。你应该记住，真正的实现者没有底层网络提供有关数据包即将发生的事情的信息！

**The Basic Assignment**

你需要编写rtinit0(), rtinit1(), rtinit2(), rtinit3() 和rtupdate0(), rtupdate1(), rtupdate2(), rtupdate3()实现如图1所示的路由表的拓扑和成本计算。

你应该将节点0到3的进程放在名为node0.c...node3.c的文件中。不允许声明任何全局变量在给定的C文件之外可见（例如，你在node0.c中定义的任何全局变量，只能在node0.c中访问）。这是强迫你遵守按照你必须采用的编码惯例，必须要在四个不同的节点中都运行过程。编译进程：cc

prog3.c node0.c node1.c node2.c node3。这些文件的原版是：node 0.c,node1.c,node2.c,node3.c。

你可以在http://gaia.cs.umass.edu/kurose/network/prog3.c中找到找到文件prog3.c的复印版。

这个作业只需要任何支持C语言的环境就可以完成，没有任何UNIX的特性。

和往常一样，需要提交一份代码，一个设计文件实验报告。

对于你的示例输出，你的进程应该打印出一条消息每当rtinit0(), rtinit1(), rtinit2(), rtinit3() 或者 rtupdate0(), rtupdate1(), rtupdate2(), rtupdate3()被调用，给出时间（通过全局变量中的clocktime获得）。对于rtupdate0(), rtupdate1(), rtupdate2(), rtupdate3()时应该打印传递给你的例程的路由数据包的发送者的身份，无论距离表是否更新，距离表的内容以及发送给邻居节点的信息更新了。

示例输出应该是TRACE为2的输出列表。突出每个节点中生成的最终距离表。你的程序将运行，直到网络中不再有路由数据包在传输中，此时模拟器将终止。

**The Advanced Assignment**

**写**rtlinkhandler0(int linkid, int newcost) 和 rtlinkhandler1(int linkid, int newcost)两个例程，如果链路0和1的成本发生了变化这两个例程就会被调用。这两个例程应该在文件node0.c,node1.c中分别定义。程序将通过链路另一端的相邻节点的名称（id），其开销已经改变了，和新的成本相关。注意当一条链路成本更改后，这些例程将必须更新距离表，并且可能（或不可能）必须向相邻节点发送更新的路由数据包。

为了作业的这个部分，你需要将常量链接的值changes（prog3.c中的第3行）更改为1。

仅供参考。FYI，链接的成本将在time10000时从1变为20，然后在time20000时改回1。你的例程将在这些时间被调用。

建议：首先完成任务中的基础部分和框架然后在此基础上进行扩展！

**Q&A**

关于这个实验，学生们提出了很多问题。如果你有兴趣看收到的问题（和答案），请查看

http://gaia.cs.umass.edu/kurose/network/programming\_assignment\_QA.htm