计算机网络课程报告

Routing Algorithm

路由算法实验报告

学生姓名 李润泽

学院名称 智能与计算学部

专　　业 计算机科学与技术

班　　级 2019 级计科 1 班

学　　号 3019244266

时　　间 2021 年 5 月 24 日

目录

一、实验内容概述

二、距离矢量路由算法的原理

三、算法实现过程（以节点 0 为例）

3.1 rtinit0()

3.2 rtupdate0(struct rtpkt *rcvdpkt)

四、仿真过程

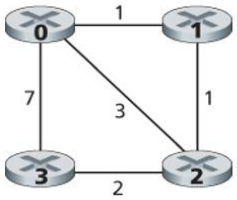
4.1 实验环境

4.2 仿真结果

五、总结

一、实验内容概述

本次实验要求同学编写一组“分布式”过程，并为下图所示的网络实现异步距离矢量路由。



为专注于算法的设计与实现，本次实验已经提供了诸如网络仿真过程的模拟代码、数据报具体收发的代码、节点除路由算法以外的功能代码等大量基础代码。这些基础代码已经构建起一套完善的网络仿真环境。

只有每个节点的路由协议算法部分留空，需要实验人员进行填补，具体包括节点的初始化操作、节点距离路由表的为方法和节点接收到路由更新信息的响应过程等内容。

二、距离矢量路由算法的原理

距离矢量路由算法的基本思想如下：

每个路由器维护一个距离矢量表，通过相邻路由器之间的距离矢量通告进行距离矢量表的更新。每个距离矢量表向包括到达目的节点的最佳输出线路和到达目的节点的所需距离，每隔一段时间，路由器会向所有邻居节点发送它到每一个目的节点的距离表，同时接收从每个邻居节点发来的距离表。这样以此类推，经过一段时间之后，网络中各路由器所获得的的距离矢量信息在各路由器上统一起来。从而每一个路由器只需要查看这个距离矢量表就可以为不同来源的分组找到一条最佳的路由。

路由表更新规则：

1、发现了一条到达某目的地的新路径（此前不存在，即距离为 999），则在路由表中增加该路由；

2、发现了一条到达某目的地且距离更短的新路由，则用该路由替换原有路由；

3、到达某目的地的一条路由，其后继节点到达该目的地的距离发生了变化，则需要更新距离。

三、算法实现过程（以节点 0 为例）

3.1 rtinit0()

该函数在仿真开始前调用，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，然后向直接相邻的节点发送自身到网络其他节点的最近距离。

函数实现如下：

```
void rtinit0() {
    printf("Node0 init:\n");
    int i,j;
    for(i=0;i<4;i++){
        for(j=0;j<4;j++){
            dt0.costs[i][j] = 999;
        }
    }
    for(i=0;i<4;i++){
        dt0.costs[0][i] = cost_n0[i];
    }
    printdt0(&dt0);
    struct rtpkt pkt0;
    for(i=1;i<4;i++){
        createrpkt(&pkt0,0,i,cost_n0);
        tolayer2(pkt0);
        printf("    Node0 is sending packet. Source:%d, Dest:%d.\n",pkt0.sourceid,pkt0.destid);
    }
    printf("Finish init Node0.\n");
    printf("\n");
}
```

3.2 rtupdate0(struct rtpkt *rcvdpkt)

该函数在节点 0 接收到网络消息封包时调用，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 应合理响应消息封包的内容，及时更新自身的各种数据结构，完成大部分路由算法功能。

函数实现如下：

```
void rtupdate0(rcvdpkt)
  struct rtpkt *rcvdpkt;{
    printf("Node0 update:\n");
    if(rcvdpkt->destid!=0){
        printf("    Warning(Node0): Illigal destination ID from received packet.\n");
        return;
    }
    if(rcvdpkt->sourceid>4||rcvdpkt->sourceid<=0){
        printf("    Warning(Node0): Illigal source ID from received packet.\n");
        return;
    }
    int destid = rcvdpkt->destid;
    int sourceid = rcvdpkt->sourceid;
    int tmp[4] = {999,999,999,999};
    int recvcost[4];
    int flag = 0;
    int i;
    for(i=0;i<4;i++){
        recvcost[i] = rcvdpkt->mincost[i];
    }
    for(i=0;i<4;i++){
        dt0.costs[sourceid][i] = recvcost[i];
        buffer0[sourceid][i] = cost_n0[sourceid] + recvcost[i];
    }
```

```
    printf("    Received packet from Node%d, mincost:(%d %d %d %d)\n",sourceid,recvcost[0],
recvcost[1],recvcost[2],recvcost[3]);
    int j;
    for(i=0;i<4;i++){
        for(j=0;j<4;j++){
            if(buffer0[j][i]<tmp[i]) tmp[i] = buffer0[j][i];
        }
        if(tmp[i]!=dt0.costs[0][i]) flag = 1;
    }
    if(flag){
        struct rtpkt sendpkt;
        for(i=0;i<4;i++){
            dt0.costs[0][i] = tmp[i];
        }
        for(i=1;i<4;i++){
            createrpkt(&sendpkt,0,i,dt0.costs[0]);
            tolayer2(sendpkt);
            printf("  Node0 is sending packet. Source:%d, Dest:%d.\n",sendpkt.sourceid,
sendpkt.destid);
        }
        printdt0(&dt0);
        printf("Node0 has already been updated!\n");
        printf("\n");
        return;
    }
    printdt0(&dt0);
    printf("Node0 do not need change.\n");
    printf("\n");
    return;
}
```

四、仿真过程

4.1 实验环境

编码 IDE：Dev-C++ 5.11；

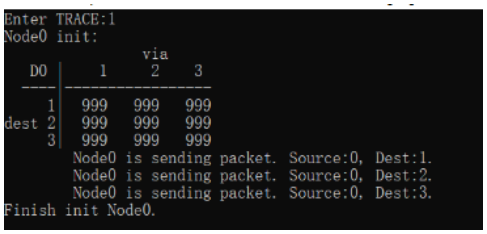
实验基础源码：node0.c，node1.c，node2.c，node3.c；

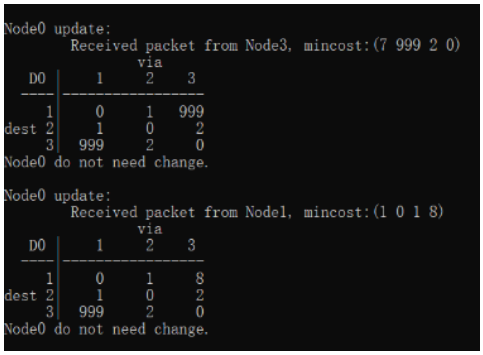
仿真控制器：prog3.c。

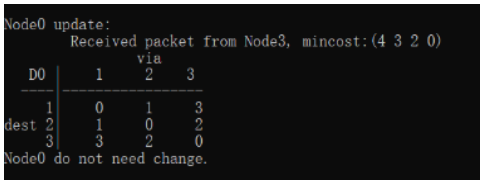
4.2 仿真结果

在 Dev-C++ 5.11 中编译并运行该项目，可以得到以下结果：

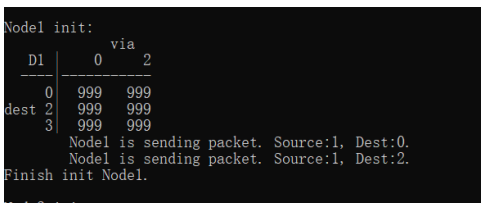
Node0：







Node1:



```
Node1 update:
    Received packet from Node0, mincost:(0 1 3 7)
    Node1 is sending packet. Source:1, Dest:0.
    Node1 is sending packet. Source:1, Dest:2.
              via
   D1 |    0     2
  ----|-----------
     0|    0     3
dest 2|  999   999
     3|  999   999
Node1 has already been updated!

Node1 update:
    Received packet from Node2, mincost:(3 1 0 2)
    Node1 is sending packet. Source:1, Dest:0.
    Node1 is sending packet. Source:1, Dest:2.
              via
   D1 |    0     2
  ----|-----------
     0|    0     3
dest 2|    3     0
     3|  999   999
Node1 has already been updated!
```

```
Node1 update:
    Received packet from Node0, mincost:(0 1 2 4)
              via
   D1 |    0     2
  ----|-----------
     0|    0     2
dest 2|    2     0
     3|  999   999
Node1 do not need change.
```

Node2:

```
Node2 init:
              via
   D2 |    0     1    3
  ----|----------------
     0|  999   999   999
dest 1|  999   999   999
     3|  999   999   999
    Node2 is sending packet. Source:2, Dest:0.
    Node2 is sending packet. Source:2, Dest:1.
    Node2 is sending packet. Source:2, Dest:3.
Finish init Node2.
```

```
Node2 update:
    Received packet from Node0, mincost:(0 1 2 4)
              via
   D2 |    0     1    3
  ----|----------------
     0|    0     1    4
dest 1|    1     0    3
     3|    5     3    0
Node2 do not need change.

Node2 update:
    Received packet from Node3, mincost:(4 3 2 0)
              via
   D2 |    0     1    3
  ----|----------------
     0|    0     1    4
dest 1|    1     0    3
     3|    4     3    0
Node2 do not need change.
```

Node3:

```
Node3 init:
              via
   D3 |    0     2
  ----|-----------
     0|  999   999
dest 1|  999   999
     2|  999   999
    Node3 is sending packet. Source:3, Dest:0.
    Node3 is sending packet. Source:3, Dest:2.
Finish init Node3.
```

```
Node3 update:
    Received packet from Node2, mincost:(2 1 0 2)
    Node3 is sending packet. Source:3, Dest:0.
    Node3 is sending packet. Source:3, Dest:2.
              via
   D3 |    0     2
  ----|-----------
     0|    0     2
dest 1|  999   999
     2|    2     0
Node3 has already been updated!
```

```
Node3 update:
    Received packet from Node0, mincost:(0 1 2 4)
              via
   D3 |    0     2
  ----|-----------
     0|    0     2
dest 1|  999   999
     2|    2     0
Node3 do not need change.
```

五、总结

本次实验相较于前两次实验相对容易，虽然需要编写 4 个 c 文件中的空缺函数，但本质大体相同。通过本次实验，我理解并顺利实现了基于 DV 算法的路由协议算法的相关实验。回顾此次路由协议算法的编写，我从理论到实践，学到了很多，不仅巩固了以前学到的理论知识，而且学以致用，将其运用在实践中。只有从理论中进行实践并获得实质成果，才能真正掌握这门技术，并提高自身独立思考的能力。未来我会不断提高自己的理论和实践能力，并对实验进行改进与完善，提升自己在计算机网络方面的理论知识和实践能力。