计算机网络网络层实验3报告

马成 20307130112

代码逻辑

1. init()的时候先将距离表全部写为999，然后根据实际边将邻居节点附上对应的值，初始化best和connection。最后向所有邻居节点发包

```
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
        dt0.costs[i][j] = 999;
    }
}
struct rtpkt tt;
tt.sourceid = 0;
for (int j = 0; j < 4; j++) {
    dt0.costs[0][j] = connectcosts0[j];
    tt.mincost[j] = connectcosts0[j];
}
for (int j = 0; j < 4; j++) {
    if (j == 0||connectcosts0[j]==999) continue;
    tt.destid = j;
    tolayer2(tt);
}
```

2. rtupdate

    1. 更新自己距离表中有关这个包source节点的信息

```
int sid = rcvdpkt->sourceid;
for (int i = 0; i < 4; i++) {
    dt0.costs[sid][i] = rcvdpkt->mincost[i];
}
```

    2. 保存原有距离表后将距离表重置再次计算，在计算的时候还要注意对best数组的更新。最后检查是否更改

```
int change = 0;
int temp[4];
for (int j = 0; j < 4; j++) {
    temp[j] = dt0.costs[0][j];
    dt0.costs[0][j] = 999;
    best0[j] = -1;
}
for (int j = 0; j < 4; j++) {
    if (j == 0) {
        dt0.costs[j][j] = 0;
        best0[0] = 0;
    }
    for (int k = 0; k < 4; k++) {
        if (connectcosts0[k] + dt0.costs[k][j] < dt0.costs[0][j]) {
```

```
                dt0.costs[0][j] = connectcosts0[k] + dt0.costs[k][j];
                best0[j] = k;
            }
        }
    }
    for (int j = 0; j < 4; j++) {
        if (dt0.costs[0][j] != temp[j]) change = 1;
    }
```

3. 如果有修改向邻居发包，这里会使用毒性逆转的操作

```
struct rtpkt tt;
tt.sourceid = 0;
if (change) {
    for (int j = 0; j < 4; j++) {
        if (j == 0 || connectcosts0[j] == 999) continue;
        tt.destid = j;
        for (int k = 0; k < 4; k++) {
            if (best0[k] == j) tt.mincost[k] = 999;
            else tt.mincost[k] = dt0.costs[0][k];
        }
        tolayer2(tt);
    }
}
```

3. linkhandler

1. 更新距离表

```
connectcosts0[linkid] = newcost;
```

2. 其余和rtupdate函数的2、3步骤几乎完全一致

关键结果

1. 我在测试程序中修改了一句输出代码

```
if (TRACE > 1) {
    printf("MAIN: rcv event, t=%.3f, at %d",
           eventptr->evtime, eventptr->eventity);
    if (eventptr->evtype == FROM_LAYER2) {
        printf(" src:%2d,", eventptr->rtpktptr->sourceid);
        printf(" dest:%2d,", eventptr->rtpktptr->destid);
        printf(" contents: %3d %3d %3d %3d\n",
               eventptr->rtpktptr->mincost[0], eventptr->rtpktptr-
>mincost[1],
               eventptr->rtpktptr->mincost[2], eventptr->rtpktptr-
>mincost[3]);
    }
    // add by myself
    //without the nextline more than one sentenec will print at one line
    else printf("\n");
}
```

2. 为了方便我在每一次接受到一个包处理过后会输出完整的距离表

3. 没有进行链路修改（仅显示最终每个路由器的距离表，由于有毒性逆转结果可能比较奇怪）

```
 D0 |    0    1    2    3
----|---------------------
   0|    0    1    2    4
   1|  999    0    1    3
   2|    2    1    0    2
   3|    4    3    2    0

 D1 |    0    1    2    3
----|---------------------
   0|    0  999  999  999
   1|    1    0    1    3
   2|  999  999    0    2
   3|  999  999  999  999

 D2 |    0    1    2    3
----|---------------------
   0|    0    1    2    4
   1|    1    0  999  999
   2|    2    1    0    2
   3|  999  999  999    0

 D3 |    0    1    2    3
----|---------------------
   0|    0    1    2    4
   1|  999  999  999  999
   2|    2    1    0  999
   3|    4    3    2    0

完整距离表
final |    0    1    2    3
  ----|---------------------
     0|    0    1    2    4
     1|    1    0    1    3
     2|    2    1    0    2
     3|    4    3    2    0
```

4. 进行第一次修改

```
 D0 |    0    1    2    3
----|---------------------
   0|    0    4    3    5
   1|    4    0    1    3
   2|  999    1    0    2
   3|    5    3    2    0

 D1 |    0    1    2    3
----|---------------------
   0|    0    4    3    5
   1|    4    0    1    3
   2|    3  999    0    2
   3|  999  999  999  999
```

```
 D2 |    0    1    2    3
----|---------------------
   0|    0  999  999  999
   1|  999    0  999  999
   2|    3    1    0    2
   3|  999  999  999    0


 D3 |    0    1    2    3
----|---------------------
   0|    0    4    3    5
   1|  999  999  999  999
   2|    3    1    0  999
   3|    5    3    2    0
```

完整距离表

```
final |    0    1    2    3
  ----|---------------------
     0|    0    4    3    5
     1|    4    0    1    3
     2|    3    1    0    2
     3|    5    3    2    0
```

5. 第二次修改

```
 D0 |    0    1    2    3
----|---------------------
   0|    0    1    2    4
   1|  999    0    1    3
   2|    2    1    0    2
   3|    4    3    2    0

 D1 |    0    1    2    3
----|---------------------
   0|    0  999  999  999
   1|    1    0    1    3
   2|  999  999    0    2
   3|  999  999  999  999


 D2 |    0    1    2    3
----|---------------------
   0|    0    1    2    4
   1|    1    0  999  999
   2|    2    1    0    2
   3|  999  999  999    0


 D3 |    0    1    2    3
----|---------------------
   0|    0    1    2    4
   1|  999  999  999  999
   2|    2    1    0  999
   3|    4    3    2    0
```

完整距离表

```
final |    0    1    2    3
  ----|--------------------
     0|    0    1    2    4
     1|    1    0    1    3
     2|    2    1    0    2
     3|    4    3    2    0
```