191180048 黄奥成 通信工程

报告看点为1、4、5部分，其他部分仅为展示实验过程以及结果。

1、实现无限拓展节点

事实上，合理的使用C的特性宏以及#include能够大幅度提高代码的复用性以及简洁性。这里我们使用宏的拼接功能并且运用include复用代码。

对于不同node，其功能函数的命名会根据其文件开始的`#define N`与`#define ALTER_SEQNUM(X)`不同而自适应变化，这样就不会出现重复定义。

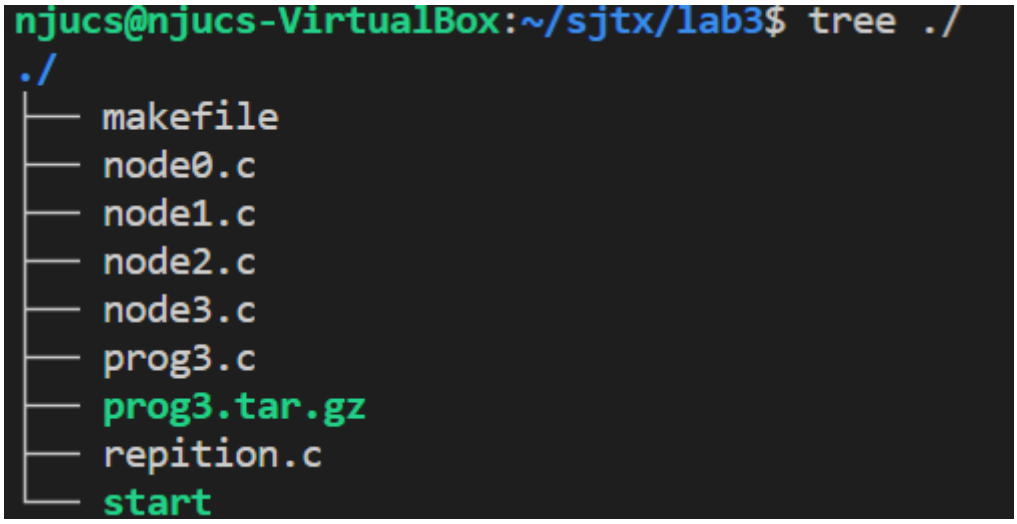
```
//node1.c
#include <stdio.h>

#define N 0
#define ALTER_SEQNUM(X) X##0

int ALTER_SEQNUM(connectcosts)[4] = { 0,  1,  3, 7 };

#include "repition.c"
```

这个宏的具体作用是在X后拼接一个序号例如

```
struct distance_table
{
    int costs[4][4];
} ALTER_SEQNUM(dt);  ======>>>  dt0

int ALTER_SEQNUM(connectcosts)[4];
              ||
              ||
              \/
int connectcosts0[4];

void ALTER_SEQNUM(rtupdate)(rcvdpkt)
```

```
struct rtpkt *rcvdpkt;
              ||
              ||
              \/
void rtupdate0(rcvdpkt)
struct rtpkt *rcvdpkt;


```

具体的文件结构是这样的



而重复的部分，也就是node的核心代码，则在`repition.c`中。这样做的好处是，如果想要添加节点，不必修改每个节点的函数名称而只需要在文件的开头初始化其序号以及拓扑关系（也就是`connectcosts[4]`）。

接下来重点介绍每个函数的实现。

2、具体函数实现

2.1 初始准备

```
void tolayer2(struct rtpkt packet);

void creatertpkt(struct rtpkt *initrtpkt,
            int srcid,
            int destid,
            int mincosts[]);

void printall(struct distance_table *dtptr, int n);
```

由于框架自带的函数`tolayer2`、`creatertpkt`、还有我自己写的`printall`定义在`prog3.c`中。因此在每个节点下(或者说`repition.c`)要先声明才能够使用。

2.2 void ALTER_SEQNUM(rtinit)

```
void ALTER_SEQNUM(rtinit)()
{
    //先将dt表所有边置为无穷
    for (int i = 0; i < 4; ++i)
        for (int j = 0; j < 4; ++j)
            ALTER_SEQNUM(dt).costs[i][j] = 999;
    //然后将自己临近的边置为初始化的值
    for (int i = 0; i < 4; ++i)
        ALTER_SEQNUM(dt).costs[N][i] = ALTER_SEQNUM(connectcosts)[i];
    //接着通知临近节点
    ALTER_SEQNUM(rtremind)();
}
```

2.3 void ALTER_SEQNUM(rtremind)

具体介绍在注释中体现，这里出现了一个小bug恶心了我一小时，在后文DEBUG中阐述

```
void ALTER_SEQNUM(rtremind)()
{
    //便利所有节点
    for (int i = 0; i < 4; ++i)
    {   //当节点为自己或者节点不相邻时continue
        if (i == N || ALTER_SEQNUM(connectcosts)[i] == 999)
            continue;
        //若是相邻节点则创建包并且发送至layer2
        else
        {
            struct rtpkt *newpkt = (struct rtpkt *)malloc(sizeof(struct
rtpkt));
            creatertpkt(newpkt, N, i, ALTER_SEQNUM(dt).costs[N]);
            tolayer2(*newpkt);
        }
    }
    // printall(ALTER_SEQNUM(&dt),N);
}
```

2.4 void ALTER_SEQNUM(rtupdate)

```

void ALTER_SEQNUM(rtupdate)(rcvdpkt) struct rtpkt *rcvdpkt;
{
    //如果收到的不是自己的包则返回，当然翻看源码后这种可能是不存在的
    if (rcvdpkt->destid != N)
        return;

    int modifyid = rcvdpkt->sourceid;
    //首先将自己表中的关于srcid的权值更新
    for (int i = 0; i < 4; ++i)
    {
        if (i == modifyid)
        {
```

```
            for (int j = 0; j < 4; ++j)
            {
                ALTER_SEQNUM(dt).costs[i][j] = rcvdpkt->mincost[j];
            }
        }
    }
    //然后查看是否因为srcid权值更新有新的最短路径
    int flag = 0;
    for (int i = 0; i < 4; ++i)
    {
        int min = ALTER_SEQNUM(connectcosts)[i];
        for(int j=0;j<4;++j)
        {   if(j == N)
                continue;

            if (min > ALTER_SEQNUM(connectcosts)[j] +
ALTER_SEQNUM(dt).costs[j][i])
            {
                min = ALTER_SEQNUM(connectcosts)[j] +
ALTER_SEQNUM(dt).costs[j][i];
            }
        }
        if(min != ALTER_SEQNUM(dt).costs[N][i]){
          flag = 1;
        }

        ALTER_SEQNUM(dt).costs[N][i] = min;
    }
    //如果有新的最短路径或者说有关于自己的权值的改变
    if (flag)
    {
        ALTER_SEQNUM(rtremind)();
    }
    printall(ALTER_SEQNUM(&dt), N);
}

```

2.5 void ALTER_SEQNUM(linkhandler)

事实上，它太过于简单我甚至不相信它这么简单还是附加作业

```
void ALTER_SEQNUM(linkhandler)(int linkid, int newcost)
{
    printf("\nLINK CHANGE!!!\n");
    ALTER_SEQNUM(connectcosts)[linkid] = newcost;
    for (int i = 0; i < 4; ++i)
        ALTER_SEQNUM(dt).costs[N][i] = ALTER_SEQNUM(connectcosts)[i];

    ALTER_SEQNUM(rtremind)();
    printall(ALTER_SEQNUM(&dt), N);
}
```

3、输出剖析

1 初始化收敛

```
njucs@njucs-VirtualBox:~/sjtx/lab3$ make run
gcc -w prog3.c node0.c node1.c node2.c node3.c -o start
./start
Enter TRACE:2
src 0 send cost to dest 1
src 0 send cost to dest 2
src 0 send cost to dest 3
src 1 send cost to dest 0
src 1 send cost to dest 2
src 2 send cost to dest 0
src 2 send cost to dest 1
src 2 send cost to dest 3
src 3 send cost to dest 0
src 3 send cost to dest 2
MAIN: rcv event, t=0.947, at 3 src: 0, dest: 3, contents:   0   1   3   7
src 3 send cost to dest 0
src 3 send cost to dest 2
          via times:1
   D3 |    0    1    2    3
  ----|-----------------
     0|  0    1    3    7
     1|999  999  999  999
     2|999  999  999  999
     3|  7    8    2    0
MAIN: rcv event, t=0.992, at 0 src: 1, dest: 0, contents:   1   0   1 999
src 0 send cost to dest 1
src 0 send cost to dest 2
src 0 send cost to dest 3
          via times:1
   D0 |    0    1    2    3
  ----|-----------------
     0|  0    1    2    7
     1|  1    0    1  999
     2|999  999  999  999
     3|999  999  999  999
MAIN: rcv event, t=1.209, at 3 src: 2, dest: 3, contents:   3   1   0   2
src 3 send cost to dest 0
src 3 send cost to dest 2
          via times:2
   D3 |    0    1    2    3
  ----|-----------------
     0|  0    1    3    7
     1|999  999  999  999
     2|  3    1    0    2
     3|  5    3    2    0
MAIN: rcv event, t=1.276, at 3 src: 0, dest: 3, contents:   0   1   2   7
          via times:3
   D3 |    0    1    2    3
  ----|-----------------
     0|  0    1    2    7
```

```
     1|999  999   999   999
     2|  3    1     0     2
     3|  5    3     2     0
//中间省略200行
          via times:8
   D0 |    0    1     2    3
  ----|-----------------
     0|  0    1     2     4
     1|  1    0     1     3
     2|  2    1     0     2
     3|  5    3     2     0
MAIN: rcv event, t=7.579, at 3 src: 0, dest: 3, contents:   0   1   2   4
          via times:6
   D3 |    0    1     2    3
  ----|-----------------
     0|  0    1     2     4
     1|999  999   999   999
     2|  2    1     0     2
     3|  4    3     2     0
MAIN: rcv event, t=7.941, at 1 src: 0, dest: 1, contents:   0   1   2   4
          via times:6
   D1 |    0    1     2    3
  ----|-----------------
     0|  0    1     2     4
     1|  1    0     1     3
     2|  2    1     0     2
     3|999  999   999   999
MAIN: rcv event, t=8.086, at 0 src: 3, dest: 0, contents:   4   3   2   0
          via times:9
   D0 |    0    1     2    3
  ----|-----------------
     0|  0    1     2     4
     1|  1    0     1     3
     2|  2    1     0     2
     3|  4    3     2     0
MAIN: rcv event, t=8.639, at 2 src: 1, dest: 2, contents:   1   0   1   3
          via times:9
   D2 |    0    1     2    3
  ----|-----------------
     0|  0    1     2     5
     1|  1    0     1     3
     2|  2    1     0     2
     3|  5    3     2     0
MAIN: rcv event, t=8.943, at 2 src: 3, dest: 2, contents:   4   3   2   0
          via times:10
   D2 |    0    1     2    3
  ----|-----------------
     0|  0    1     2     5
     1|  1    0     1     3
     2|  2    1     0     2
     3|  4    3     2     0
MAIN: rcv event, t=9.960, at 2 src: 0, dest: 2, contents:   0   1   2   4
          via times:11
   D2 |    0    1     2    3
  ----|-----------------
```

```
     0|  0    1     2     4
     1|  1    0     1     3
     2|  2    1     0     2
     3|  4    3     2     0
```

可以看到，到程序结束前，四个节点都收敛到相同的矩阵(D1、D3由于互不相连，因此无法获得相互的报文，因而互相不会更新其表项)。

```
D0/D2|  0    1     2    3
 ----|-----------------
    0|  0    1     2     4
    1|  1    0     1     3
    2|  2    1     0     2
    3|  4    3     2     0

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

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

2 Linkchange收敛

```
MAIN: rcv event, t=10000.000, at -1
LINK CHANGE!!!
src 0 send cost to dest 1
src 0 send cost to dest 2
src 0 send cost to dest 3
          via times:19
   D0 |    0    1     2    3
  ----|-----------------
     0|  0   20     3     7
     1|  1    0     1     3
     2|  2    1     0     2
     3|  4    3     2     0

LINK CHANGE!!!
src 1 send cost to dest 0
src 1 send cost to dest 2
          via times:13
   D1 |    0    1     2    3
  ----|-----------------
     0|  0    1     2     4
     1| 20    0     1   999
     2|  2    1     0     2
     3|999  999   999   999
```

```
MAIN: rcv event, t=10000.178, at 1 src: 0, dest: 1, contents:   0  20   3
7
          via times:14
   D1 |    0    1    2    3
  ----|----------------
     0|  0    1    2    4
     1| 20    0    1   999
     2|  2    1    0    2
     3|999  999   999   999
src 1 send cost to dest 0
src 1 send cost to dest 2
          via times:15
   D1 |    0    1    2    3
  ----|----------------
     0|  0   20    3    7
     1|  3    0    1    3
     2|  2    1    0    2
     3|999  999   999   999
MAIN: rcv event, t=10000.702, at 0 src: 1, dest: 0, contents:  20   0   1
999
          via times:20
   D0 |    0    1    2    3
  ----|----------------
     0|  0   20    3    7
     1|  1    0    1    3
     2|  2    1    0    2
     3|  4    3    2    0
src 0 send cost to dest 1
src 0 send cost to dest 2
src 0 send cost to dest 3
          via times:21
   D0 |    0    1    2    3
  ----|----------------
     0|  0    4    3    5
     1| 20    0    1   999
     2|  2    1    0    2
     3|  4    3    2    0
MAIN: rcv event, t=10000.809, at 0 src: 1, dest: 0, contents:   3   0   1
3
          via times:22
   D0 |    0    1    2    3
  ----|----------------
     0|  0    4    3    5
     1| 20    0    1   999
     2|  2    1    0    2
     3|  4    3    2    0
          via times:23
   D0 |    0    1    2    3
  ----|----------------
     0|  0    4    3    5
     1|  3    0    1    3
     2|  2    1    0    2
     3|  4    3    2    0
MAIN: rcv event, t=10001.166, at 3 src: 0, dest: 3, contents:   0  20   3
7
```

```
          via times:13
   D3 |    0    1    2    3
  ----|----------------
     0|  0    1    2    4
     1|999  999  999  999
     2|  2    1    0    2
     3|  4    3    2    0
          via times:14
   D3 |    0    1    2    3
  ----|----------------
     0|  0   20    3    7
     1|999  999  999  999
     2|  2    1    0    2
     3|  4    3    2    0
MAIN: rcv event, t=10001.777, at 1 src: 0, dest: 1, contents:   0   4   3
5
          via times:16
   D1 |    0    1    2    3
  ----|----------------
     0|  0   20    3    7
     1|  3    0    1    3
     2|  2    1    0    2
     3|999  999  999  999
          via times:17
   D1 |    0    1    2    3
  ----|----------------
     0|  0    4    3    5
     1|  3    0    1    3
     2|  2    1    0    2
     3|999  999  999  999
MAIN: rcv event, t=10001.964, at 2 src: 0, dest: 2, contents:   0  20   3
7
          via times:23
   D2 |    0    1    2    3
  ----|----------------
     0|  0    1    2    4
     1|  1    0    1    3
     2|  2    1    0    2
     3|  4    3    2    0
          via times:24
   D2 |    0    1    2    3
  ----|----------------
     0|  0   20    3    7
     1|  1    0    1    3
     2|  2    1    0    2
     3|  4    3    2    0
MAIN: rcv event, t=10002.357, at 3 src: 0, dest: 3, contents:   0   4   3
5
          via times:15
   D3 |    0    1    2    3
  ----|----------------
     0|  0   20    3    7
     1|999  999  999  999
     2|  2    1    0    2
     3|  4    3    2    0
```

```
          via times:16
   D3 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|999  999   999   999
     2|  2    1     0     2
     3|  4    3     2     0
MAIN: rcv event, t=10003.342, at 2 src: 1, dest: 2, contents:  20   0   1
999
          via times:25
   D2 |    0    1     2    3
  ----|-----------------
     0|  0   20     3     7
     1|  1    0     1     3
     2|  2    1     0     2
     3|  4    3     2     0
src 2 send cost to dest 0
src 2 send cost to dest 1
src 2 send cost to dest 3
          via times:26
   D2 |    0    1     2    3
  ----|-----------------
     0|  0   20     3     7
     1| 20    0     1   999
     2|  3    1     0     2
     3|  4    3     2     0
MAIN: rcv event, t=10003.536, at 1 src: 2, dest: 1, contents:   3   1   0
2
          via times:18
   D1 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  3    0     1     3
     2|  2    1     0     2
     3|999  999   999   999
src 1 send cost to dest 0
src 1 send cost to dest 2
          via times:19
   D1 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  4    0     1     3
     2|  3    1     0     2
     3|999  999   999   999
MAIN: rcv event, t=10004.307, at 2 src: 1, dest: 2, contents:   3   0   1
3
          via times:27
   D2 |    0    1     2    3
  ----|-----------------
     0|  0   20     3     7
     1| 20    0     1   999
     2|  3    1     0     2
     3|  4    3     2     0
          via times:28
   D2 |    0    1     2    3
```

```
  ----|-----------------
     0|  0   20     3     7
     1|  3    0     1     3
     2|  3    1     0     2
     3|  4    3     2     0
MAIN: rcv event, t=10004.669, at 2 src: 0, dest: 2, contents:   0   4   3
5
          via times:29
   D2 |    0    1     2    3
  ----|-----------------
     0|  0   20     3     7
     1|  3    0     1     3
     2|  3    1     0     2
     3|  4    3     2     0
          via times:30
   D2 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  3    0     1     3
     2|  3    1     0     2
     3|  4    3     2     0
MAIN: rcv event, t=10005.288, at 0 src: 2, dest: 0, contents:   3   1   0
2
          via times:24
   D0 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  3    0     1     3
     2|  2    1     0     2
     3|  4    3     2     0
          via times:25
   D0 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  3    0     1     3
     2|  3    1     0     2
     3|  4    3     2     0
MAIN: rcv event, t=10005.301, at 0 src: 1, dest: 0, contents:   4   0   1
3
          via times:26
   D0 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  3    0     1     3
     2|  3    1     0     2
     3|  4    3     2     0
          via times:27
   D0 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  4    0     1     3
     2|  3    1     0     2
     3|  4    3     2     0
MAIN: rcv event, t=10005.304, at 3 src: 2, dest: 3, contents:   3   1   0
2
```

```
          via times:17
   D3 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|999  999   999   999
     2|  2    1     0     2
     3|  4    3     2     0
src 3 send cost to dest 0
src 3 send cost to dest 2
          via times:18
   D3 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|999  999   999   999
     2|  3    1     0     2
     3|  5    3     2     0
MAIN: rcv event, t=10005.372, at 0 src: 3, dest: 0, contents:   5   3   2
0
          via times:28
   D0 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  4    0     1     3
     2|  3    1     0     2
     3|  4    3     2     0
          via times:29
   D0 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  4    0     1     3
     2|  3    1     0     2
     3|  5    3     2     0
MAIN: rcv event, t=10005.746, at 2 src: 1, dest: 2, contents:   4   0   1
3
          via times:31
   D2 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  3    0     1     3
     2|  3    1     0     2
     3|  4    3     2     0
          via times:32
   D2 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  4    0     1     3
     2|  3    1     0     2
     3|  4    3     2     0
MAIN: rcv event, t=10006.617, at 2 src: 3, dest: 2, contents:   5   3   2
0
          via times:33
   D2 |    0    1     2    3
  ----|-----------------
     0|  0    4     3     5
     1|  4    0     1     3
```

```
   2|  3    1     0     2
   3|  4    3     2     0
        via times:34
 D2 |    0    1     2    3
----|-----------------
   0|  0    4     3     5
   1|  4    0     1     3
   2|  3    1     0     2
   3|  5    3     2     0

```

同理，最终收敛至

```
D0/D2|  0    1     2    3
 ----|-----------------
   0|  0    4     3     5
   1|  4    0     1     3
   2|  3    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    1     0     2
   3|999  999   999   999

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

```

4、DEBUG

这个bug困扰了我足足有两小时之久。一开始我没搞明白。为什么在`createrpkt`之后，明明是只传入了`dt.cost[0]`就算是误操作了修改，最多也只影响`dt.cost[0][0]`到`dt.cost[0][3]`这个范围，为什么连`dt.cost[1][0]``dt.cost[1][1]`都被修改了。如下图

```
void createrpkt(struct rtpkt *initrtpkt,
                int srcid,
                int destid,
                int mincosts[]);
```

D0	0	1	2	3
0	0	1	0	1
1	0	1	999	999
2	999	999	999	999
3	999	999	999	999

我的第一反应是`createrpkt`出了问题，经过输出地址发现，`initrtpkt`,`mincosts`的地址惊人。

```
initrtpkt:ec1400a0
mincosts:ec1400a0
```

（这里的`mincosts`也即`dt0.costs[0][0]`）。而`initrtpkt`数据结构中储存`initrtpkt.costs[0]`的数组首地址正好也是x+8，因此当`initrtpkt.costs[0]=mincosts[0]`执行后，实际上`dt.costs[0][2]=mincosts[2]`变为了`mincosts[0]`=0，然后顺理成章的`dt.costs[0][4]=1`

然后奇妙的地方来了。由于`&dt.costs[1][0]=&initrtpkt.costs[2]=x+16`也即他们的地址共享，则在运行`initrtpkt.costs[2]=mincosts[2]`时`dt.costs[1][0]=*(x+16)=initrtpkt.costs[2]=mincosts[2]=0`，这也就解释了为什么D0的表项中，本来不可能发生变化的`dt0.costs[1][0]`和`dt0.costs[1][1]`变成了0和1。

我一开始苦思冥想不明白为什么，而且当时`srcid`和`destid`也非常诡异的为x,x+4（当时可能是眼花了，因为后面3位数确实相同，但是地址的高位其实是不一样的）。我第一时间想到的是函数传值压栈的问题，但是当我查看了传入函数的地方也即，真相大白。

```
struct rtpkt *newpkt;

printf("newpkt :%x  dt: %x\n",newpkt ,&ALTER_SEQNUM(dt).costs[0][2]);
creatertpkt1(newpkt,N,i,ALTER_SEQNUM(dt).costs[0]);
```

```
newpkt :97ba9080  dt: 97ba9088
dt :7
mincosts:7
i:0 min[i]:0  1   3   7  97ba9088  97ba9088
```

传入的`newpkt.mincost[0]`的地址正好是`dt.costs[0][2]`。

事实上，我们只需要修改添加一个malloc分配堆地址就能解决这个问题，而且在逻辑上也没有问题，因为在所谓的layer2中发送的数据肯定是有它的实体存在的，而不是一个发送完数据，函数返回后就销毁的栈数据。

```
struct rtpkt *newpkt = (struct rtpkt *)malloc(sizeof(struct rtpkt));
creatertpkt(newpkt, N, i, ALTER_SEQNUM(dt).costs[N]);
```

```
newpkt :a14d5a80  dt: a0ba7088
dt :7
mincosts:7
i:0 min[i]:0  1   3   7  a14d5a88  a0ba7088
```

但是问题来了，why？？？？真的有这么巧合吗。

```
51      else
52      { printf("dt0:%x\n",dt0.costs);
53        struct rtpkt *newpkt;
54        struct rtpkt *oldpkt;
55        struct rtpkt *oldpkt2;
56
57        printf("newpkt:%x\noldpkt:%x\noldpkt2:%x\n",newpkt,oldpkt,oldpkt2);
58        creatertpkt1(newpkt,N,i,ALTER_SEQNUM(dt).costs[N]);
```

```
问题  输出  终端  端口  调试控制台

dt0:f331f0a0
newpkt:f311c857
oldpkt:0
oldpkt2:f331f0a0
```

```
53      struct rtpkt *newpkt;
54      struct rtpkt *oldpkt;
55      struct rtpkt *oldpkt2;
56      struct rtpkt *oldpkt3;
57
58      printf("newpkt:%x\noldpkt:%x\noldpkt2:%x\noldpkt3:%x\n",newpkt,oldpkt,oldpkt2,oldpkt3);
59      creatertpkt1(newpkt,N,i,ALTER_SEQNUM(dt).costs[N]);
60      tolayer2(*newpkt);
```

```
问题  输出  终端  端口  调试控制台

    3|  999   999   999
dt0:ecbfd0a0
newpkt:36695c50
oldpkt:ec9fa85e
oldpkt2:0
oldpkt3:ecbfd0a0
initrtpkt:36695c50
```

从中我们可以发现，无论传入的变量为什么，在`creatertpkt`上方倒数第一个结构体指针的地址与全局变量`dt0`的指针地址相同，而上方倒数第二个指针的值都为0，再往前的值就是随机分配的地址了。这也就导致，如果像我一开始写的那样，使用了`creatertpkt`上方倒数第一个结构体指针的地址作为传入指针，就会使得其地址重叠造成不可描述不可名状的后果。然而，至今我还是没搞明白其原理(在`creatertpkt`上方倒数第一个结构体指针的地址与全局变量`dt0`的指针地址相同，而上方倒数第二个指针的值都为0)，希望助教或者老师熟悉c语言能够给出答案。

5、写在最后

这个lab是真的纯纯的简单，不包括debug的时间，实际上我从上手到完成一个node的框架只用了不到一小时的时间。然后后续的找bug和研究浪费了两小时。然后灵机一动用宏代替复制粘贴4个node(事实上我一开始就是按照可拓展来写的)花了一小会时间，然后就没了🤔。是否有点过于简单了，完全拉不开区分度，然后就是这个框架代码真的是太过于`oldschool`或者说“上古了”，本来就不想写c++和c了，结果用了应该不是c99的标准，看框架代码都快🤮了，本来就基本上用g++编译，现在倒是好了，连gcc都疯狂报错(我当时没看到群里说用cc编译，直接用gcc -w强行编译通过了)。为此提出几点建议：

①这学期的实验都太过简单浮于理论(事实上DV算法过于简单，相信没有人会学不明白，但是写这么简单的lab会使得同学们缺少以后无论是code能力还是工程能力)，而且网上都能查到一手的的“答案”，虽然这个作业是真的简单，但身边不少人还是缝的，估计连查重都过不了，还是希望能找点modern的项目来写。

②就算还要写这个实验，希望下一届学弟学妹能够至少使用改良版本，例如修改原本文件的不符合modern C++标准的一些语句，然后拓展一下将节点的可重复性突出而不是屈于4个节点的小打小闹(在此推销一下我的框架代码，把prog3.c稍作修改就能实现更多节点的拓展)

③mininet好像还是没用上😂，这个项目用自己编的一个模拟的环境好像并不是很需要mininet