**目录结构：**

* **lab文件夹中是python绘图程序代码;**
* **Simulator文件夹中是模拟器程序代码，其中Simulator\Simulator\data文件夹中为程序输出的结果;**
* **提供的两篇论文为参考文献。**

# Mesh网络中高效无死锁平面自适应路由算法介绍

参考向东老师发表的论文[1]以及ChienA A等人发表的论文[2]中提到的平面自适应路由算法可知。

在该算法中，每条物理通道只需要三条虚拟通道就可以有效在三维以及更高维的Mesh网络中避免死锁的产生，然而采用该算法网络拓扑一维和三维分别有两条和一条虚拟通道始终处于空闲状态。

平面自适应路由算法给出了一个简单的用于n维Mesh网的死锁避免机制，每条物理通道需要三条虚拟通道。以三维的Mesh网为例来说明平面自适应路由算法的机制。Mesh网首先被划分为一系列的平面，对于三维的Mesh网，分为x-y和y-z平面。路由消息首先在x-y平面内路由，当x方向的偏移量变为0以后，消息跳转至y-z平面继续路由。x-y平面又进一步被分成上升子网与下降子网。

如图1所示，x-y平面中的虚拟通道分配方案如下：上升子网中x方向使用虚拟通道c1,y方向使用c3+;下降子网中x方向使用c2,y方向使用c3-。

如图2所示，y-z平面中的虚拟通道分配方案如下：上升子网中y方向使用虚拟通道c1,z方向使用c3+;下降子网中y方向使用c2,z方向使用c3-。

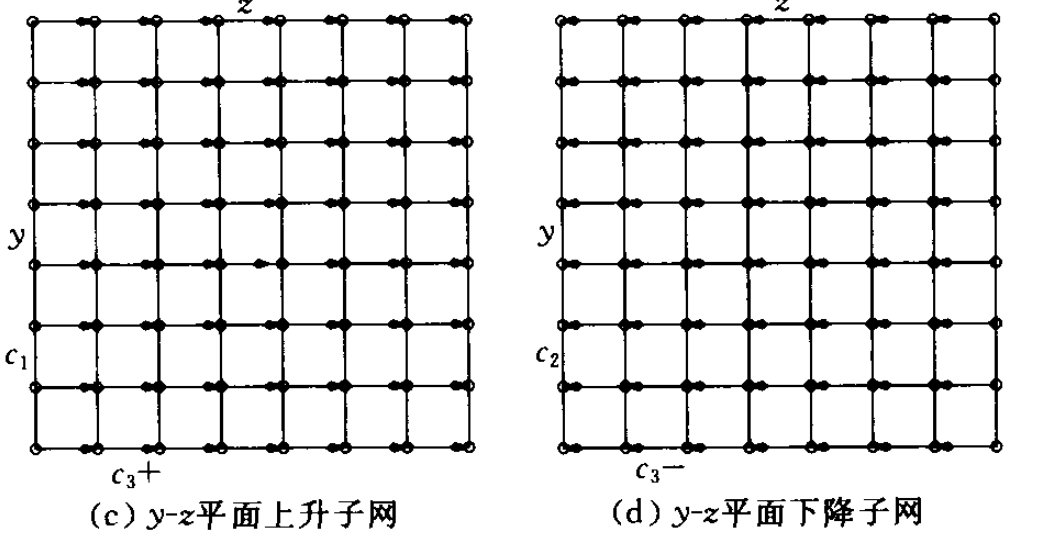
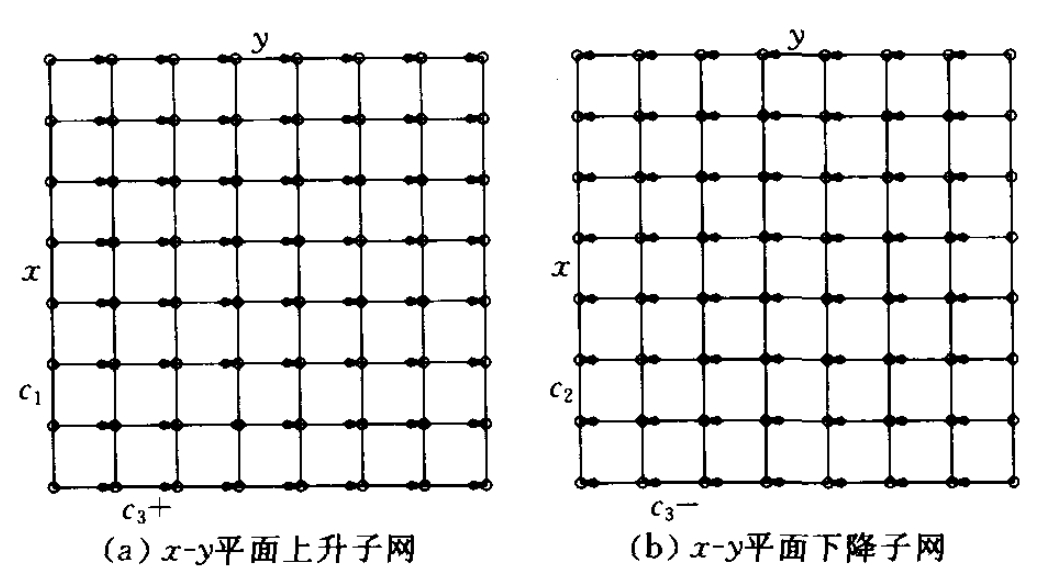
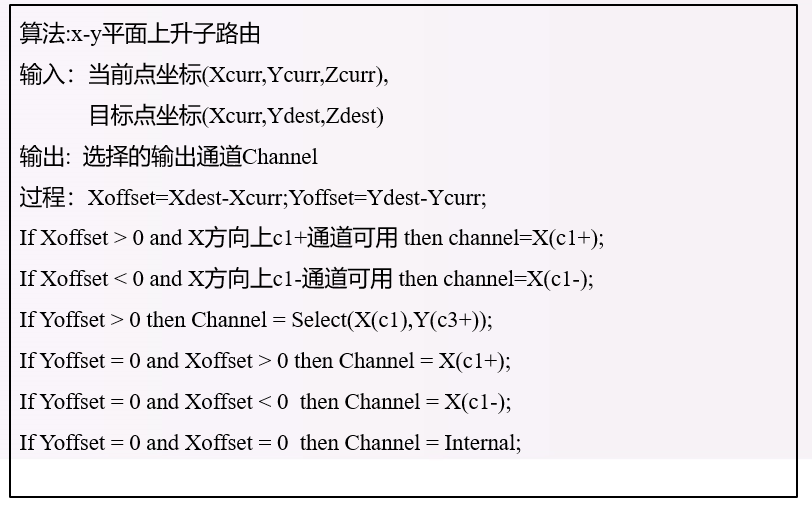
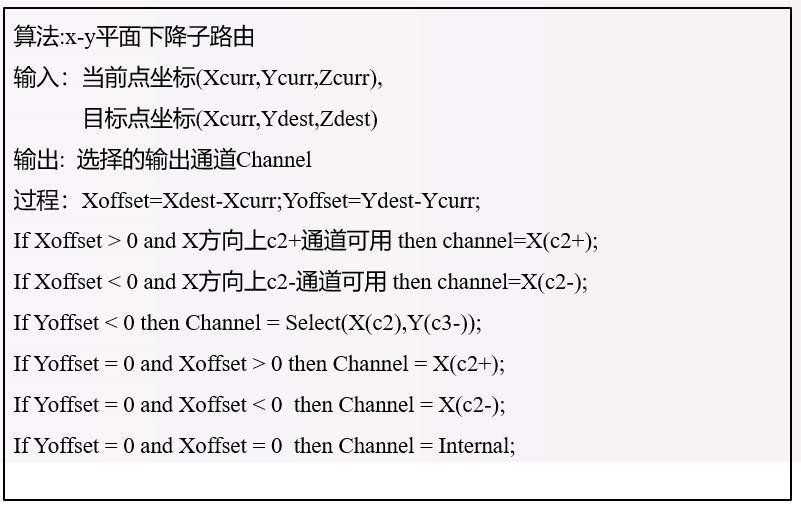


图1 图2

二维平面自适应路由算法我们只考虑x-y平面。根据向东老师论文[1]，对于自适应的容错路由算法，通常需要将每条物理通道划分为一定数量的虚拟通道来实现死锁避免机制，通道上的物理资源（如缓存、带宽）会分配到各个虚拟通道之上。如图3所示(x-y平面上升子网路由)，x-y平面x方向和y方向都只有一条物理通道。当x,y方向都有offset并且该方向虚拟通道都可用的情况下，x,y方向都是可以走的，但是我们优先考虑低维的x，先判断x方向的offset以及x方向的虚拟通道是否可用，假如不可用再判断y方向的offset以及y方向的虚拟通道。假如都不可用，就先排队等着。





# 程序介绍

## 网络基本结构：

1. 网络大小设定为二维8\*8。对应到代码里面是knode=8
2. 流量模式设定为uniform。对应到代码里是GENERATETYPE = allgen(allgen设置为1)
3. 原有代码框架是按照两个buffer也就是两个虚拟通道设计的，因此需要修改为三个虚拟通道，也就是三个buffer。
4. Q2DTorus.h文件中，修改为Q2DTorus(int n, int R1buffer, int R2buffer, int R3buffer);
5. Q2DTorusNode.h文件中，增加宏定义#define R3 3;class Buffer中增加数据成员int r3;
6. Q2DTorus.cpp文件中，构造函数修改为Q2DTorus(int n,int R1buffer,int R2buffer,int R3buffer):{ (head+id)->setbuffer(R1buffer, R2buffer, R3buffer);}
7. Q2DTorusNode.cpp文件中，bufferMin函数新添chn对R3的判断{else{r3-=n;}};bufferPlus函数新添chn对R3的判断{r3+=n;};setbuffer函数中新添{

bufferxneg->r3 = buff3;

bufferxpos->r3 = buff3;

bufferyneg->r3 = buff3;

bufferypos->r3 = buff3;

}

1. Routing2.cpp和Routing.h两个文件没有用到。

## 路由算法实现：

1. Main函数
2. 定义三个虚拟通道,int r1,r2,r3，将三个虚拟通道缓存大小都设置为1。取数组buffer数组中下标为2的元素。

int r1buffer[5] = {1, 2, 1, 1, 2};//虚拟信道1缓存大小，以message个数为基本单位。

int r2buffer[5] = {2, 1, 1, 0, 1};//虚拟信道2缓存大小，若无虚拟通道不使用r2。

int r3buffer[5] = {2, 1, 1, 0, 1 };//虚拟信道3缓存大小。

1. 定义一个输出数据的文件名，路径为data\\PAR.txt,程序会将吞吐量和平均时延随注入率变化的数据输入到该文件中。然后通过python程序绘制曲线。

ofstream out;

out.open(filename[round]);

out << linkrate(链路状态) << " " << (s->totalcir / s->messarrive)(平均时延)

<< " " << linkrate \* ((float)s->messarrive / allmess)(吞吐量) << endl;

在传播完成之后需要out.close()，不然无法写入文件中。

1. Start simulate的时候设置:

r3 = r3buffer[round] \* MESSLENGTH;

tor = new Q2DTorus(knode, r1, r2, r3); //初始化网络结构

1. Routing.cpp文件

核心算法的实现，本程序实现了平面自适应路由算法两种方式，一种是x,y维每一维都只采用一个物理通道无虚拟通道，另一种是无死锁的x,y维都采用三条虚拟通道。

1. 修改checkBuffer函数，该函数是检查传进来的虚拟通道还有没有缓存。原有设计只考虑了两个虚拟通道R1和R2，现在新增虚拟通道R3。

if( chn == R1){

if(buff1->r1 >= MESSLENGTH ){

k = true;

buff1->linkused = true;

}

}else if (chn == R2) {

if (buff1->r2 >= MESSLENGTH) {

k = true;

buff1->linkused = true;

}

}else if (chn == R3) {

if (buff1->r3 >= MESSLENGTH) {

k = true;

buff1->linkused = true;

}

}

1. prefer函数不用修改，该函数是用来判断x,y维的虚拟通道是否有缓存，决定该往哪个方向路由的，根据设计，优先判断x维的虚拟通道是否有缓存，x维假如没有再判断y维，假如x,y维都有缓存那么也优先走x方向，假如x,y维都没有缓存，那便处于等待状态。
2. Mesh网络不需要走环链路，因此我们需要修改noWrapLinkrt函数。在该函数中，根据x方向的偏移和y方向的偏移决定var1和var2两个变量的取值。当xdis<0，=0，>0时var1分别取0,1,2。原有xy路由算法规定当xdis!=0的时候不允许走y方向会将var2设置为1，而平面自适应路由算法并没有此项规定，因此在判断y方向的路由时无需考虑x方向，也就是继续判断ydis<0，=0，>0时var2分别取0,1,2，当x方向偏移不为0我们仍然可以走y方向的路由。对应程序如下：

if( xdis < 0) var1 = 0;

else {

if( xdis == 0) var1 = 1;

else if( xdis > 0) var1 = 2;

}

//planar-adaptive路由算法：x方向的偏移不为0时,仍然可以走y方向

if (ydis < 0) var2 = 0;

else {

if (ydis == 0) var2 = 1;

else if (ydis > 0) var2 = 2;

}

1. 不采用虚拟通道的方案，vchx和vchy都采用R1，然后通过prefer函数判断应该往哪个方向路由即可。对应程序为

If(ALGORITHM == 2){…}

**这种方案需要在main函数中将ALGORITHM设为2，并且将filename[2]输出结果文件名改为data\\PARNoVirChn.txt**。**而采用三条虚拟通道的方案需要将ALGORITHM设为3，输出结果文件名无需修改，采用data\\PAR.txt。**

1. 采用三条虚拟通道的方案：先判断是上升子网还是下降子网。根据之前的解释，需要判断y方向的offset，假如yoffset>0那么需要按照上升子网来路由，假如yoffset<=那么需要按照下降子网来路由。对应到程序里面就是判断var2是否等于2。对应程序如下：

//上升子网

if (var2 == 2) {

vchy = R3;(y方向走R3的虚拟通道)

vchx = R1; (x方向走R1的虚拟通道)

}

//下降子网

else {

vchy = R3; (y方向走R3的虚拟通道)

vchx = R2; (x方向走R2的虚拟通道)

}

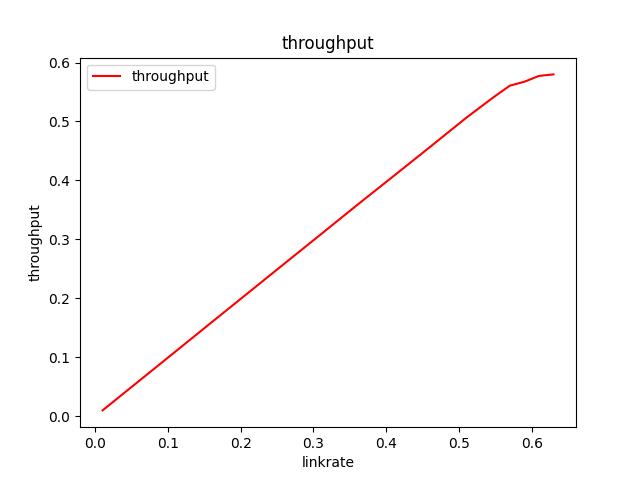
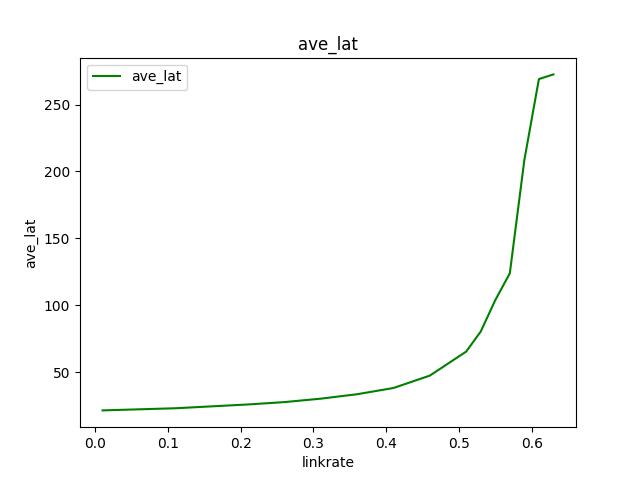
然后通过prefer函数判断应该往哪个方向路由即可。

## 平均延迟和吞吐量的实验结果：

上一章介绍将linkrate,平均时延和吞吐量输入到了PAR.txt和PARNoVirChn.txt文件中。然后采用python绘制曲线图的方式得到实验结果以及python代码如下。

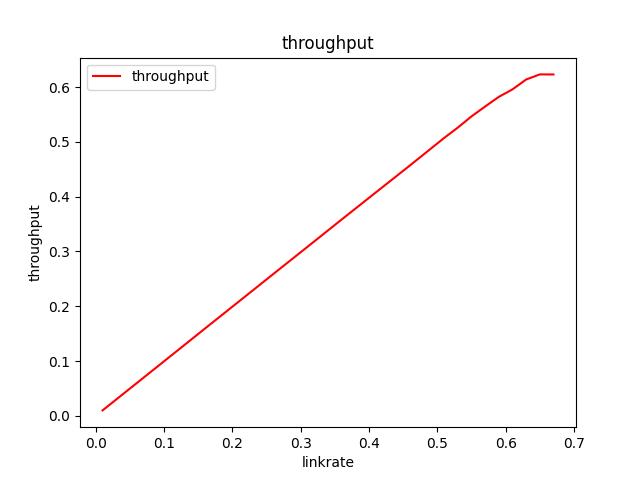
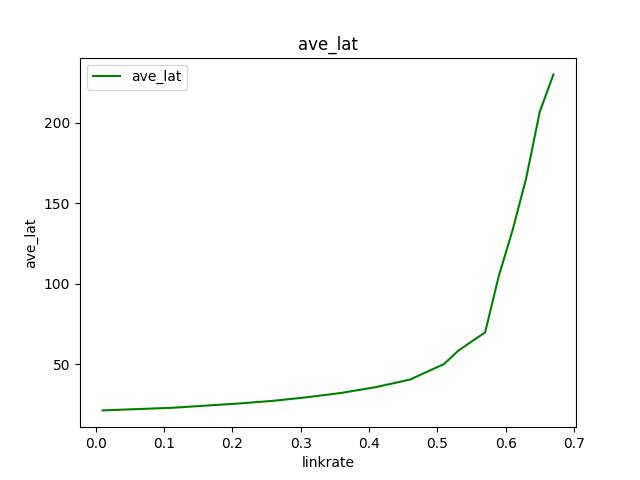
无虚拟通道：





三条虚拟通道：





**通过无虚拟通道和三条虚拟通道的对比，很明显的发现采用虚拟通道的方式确实能在相同linkrate的情况下降低延迟和提高吞吐量。**

参考文献

1. 向东,张跃鲤.Mesh网中高效无死锁自适应路由算法[J].计算机学报,2007(11):1954-1962
2. ChienA A，Kim J H．Planaradaptive routing：Low—cost adaptive networks for muItiprocessors． Journalof ACM ， 1995，42(1)：91—123