分布式系统

分布式系统 基本概念

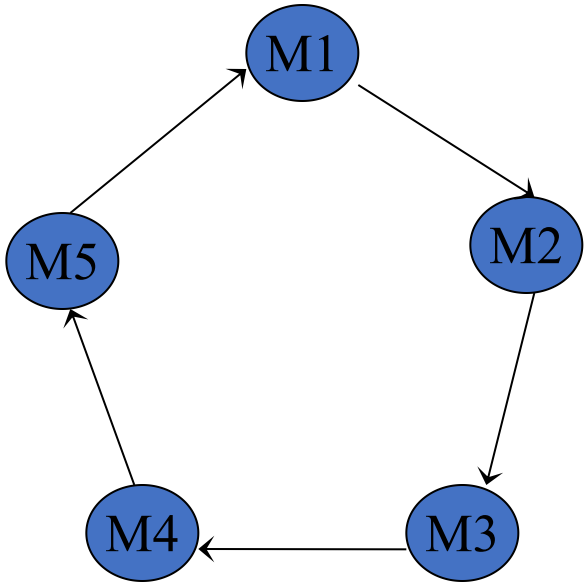
刘二腾

2018.12.24

分布式路由算法

路由算法

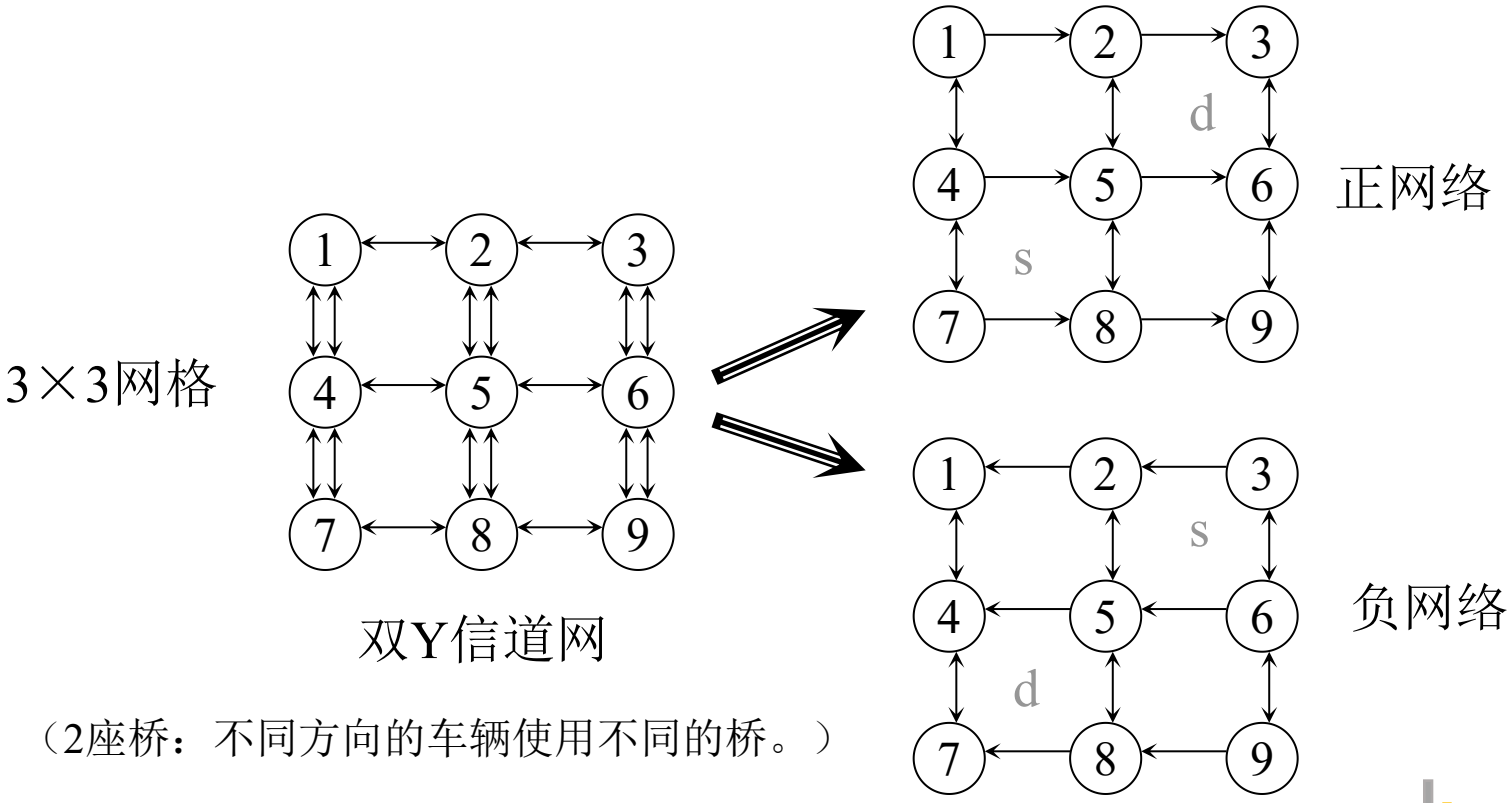
- 自适应、无死锁和容错路由
  - 网络通信中，消息在占有资源（缓冲区、信道）的情况下再申请资源，就可能会发生死锁。在自适应路由环境下，死锁更容易发生
  - 在图中形成一个循环。（自适应路由中，这种循环更有可能会形成！）
  - 表现在路由中，即：路由中形成了一个回路



路由算法

- 虚信道和虚网络
  - 在死锁的4个条件（互斥、占有并等待、不可剥夺、循环等待）中，避免路由死锁主要通过避免循环等待来实现。
  - 网络通信中的链路（信道）是双向的，路由时容易产生回路。如果避免网络通信中产生回路，那么就不会形成循环等待，路由也就不会发生死锁

- 虚信道和虚网络
  - 一种办法是通过划分子网，每个子网中没有回路。不同方向的网络通信在不同子网中进行，从而避免死锁。例



（2座桥：不同方向的车辆使用不同的桥。）

路由算法

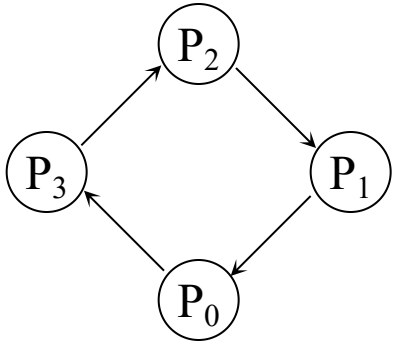
- 虚信道和虚网络
  - 当物理上不存在双（多）信道时，可以使用虚信道来实现
  - 虚信道
    - 多个虚信道对一个物理信道进行复用，每个虚信道都有自己的缓冲器。当物理信道被其它虚信道使用时，就用这个缓冲器保存信息
    - 如果虚信道之间没有循环等待，那么就可以避免死锁
    - 虚信道同时也可以提高对物理信道的利用率

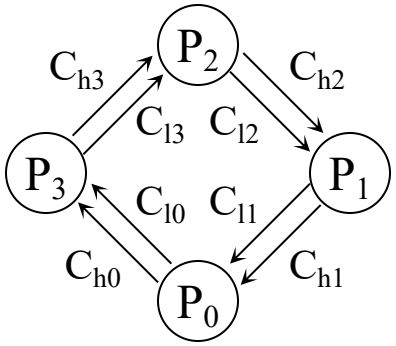
路由算法

- 虚信道和虚网络
  - 虚信道上更高一级的虚拟化是虚网络
  - 虚网络
    - 一个给定的物理网络可被分成几个虚网络
    - 每个虚网络包括一系列的虚信道
    - 在虚网络中相邻的节点被映射到物理网络中的时候也要相邻
    - 虚信道和虚网络一个是信道层次上的虚拟化，一个是网络层次上的虚拟化
    - 必须合理安排虚信道，以避免因为虚信道间的依赖性而产生死锁；虚网络通常设计成没有回路的，其中通信不会产生死锁

路由算法

- 虚信道和虚网络
  - 例：4个节点的单向环
  - 如果同时有几个路由进程启动，就会发生死锁
  - 为每个链路设计2个虚信道来避免死锁

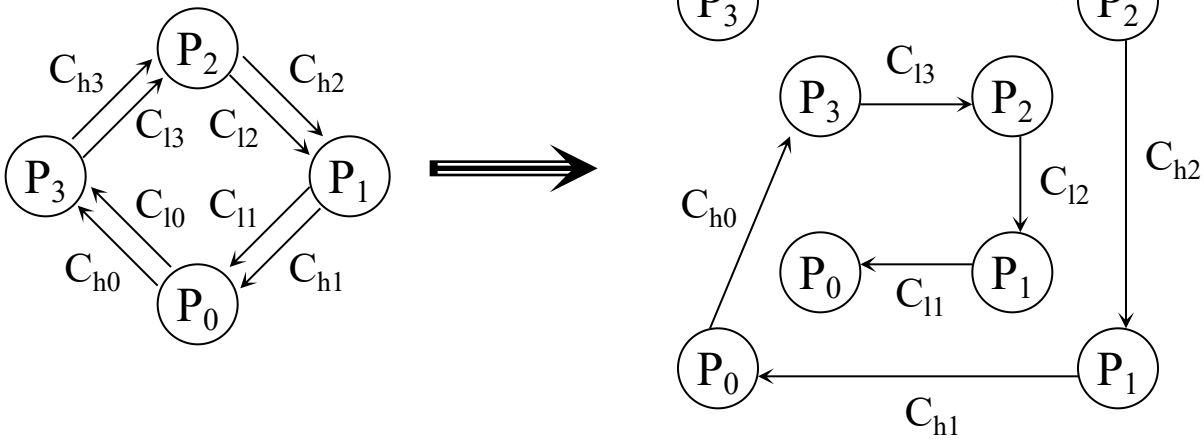




- 高虚信道：C_{h0}、C_{h1}、C_{h2}、C_{h3}
- 低虚信道：C_{l0}、C_{l1}、C_{l2}、C_{l3}

路由算法

- 虚信道和虚网络
  - 虚信道安排：
  - 如果源地址大于目的地址，可以从任何一个信道开始；但一旦使用一个高（低）信道，那么以后也要使用同一信道
  - 如果源地址小于目的地址，首先使用高信道，经过P_3节点后，高虚信道切换为低虚信道
  - 这样，相应的信道依赖图如下



*虚信道C_{l0}不被使用

• 信道依赖图中没有回路，路由不会死锁！

路由算法

- 完全自适应和无死锁路由
  - 适应性和无死锁是两个互相矛盾的要求
  - 一个确定性路由可以确保无死锁，但网络组件（链接或节点）错误时，无法适应，路由只好失效
  - 另一方面，没有限制的适应性路由很可能引发死锁
  - 目标：在不引发死锁的前提下，尽可能增加适应性

路由算法

- 完全自适应和无死锁路由
  - 为了做到完全自适应的无死锁路由，需要引入足够多的虚信道
  - 当路由开始时，使用虚信道1(vc_1)。在第i步使用虚信道i(vc_i)以及相应的链接。这样，如果一个给定网络的最长路径（网络直径）是D_{max}，就要用D_{max}个虚信道。
  - 为了减少虚信道的数量，可以通过将虚信道分类来实现
  - 将给定网络分成k个子集：$S_1,S_2,\ldots,S_k$，每个子集都不会包含相邻的节点。当一个消息从子集S_i中的一个节点移动到子集S_j中的一个节点时（这里$i<j$），称之为正移动；反之为负移动。当发生负移动时，虚信道标记加1（假定信道标记是从1开始的）。这样，所需虚信道的个数就是一个路由路径中负移动的个数
  - 选择一个合适的k及子集划分，可以使路由过程中的负移动个数最小（所需虚信道最少）

路由算法

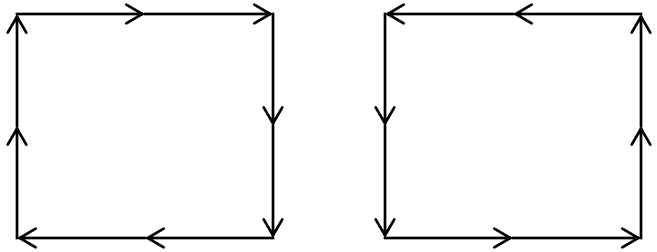
- 逃逸信道（Escape channels）
  - 要实现完全自适应的无死锁路由往往是困难的。通常使用混合路由
  - 系统中包括2个路由，一个是使用标记为非等待的虚信道（不能阻塞）的完全适应性路由；另一个是限制性但无死锁路由（如决定性路由），使用标记为等待的虚信道（称为逃逸信道，可以阻塞）
  - 开始时，使用完全适应路由，直到阻塞；然后切换到限制性路由
  - 这样，当信道不繁忙（不用等待）时，就使用完全适应路由，以满足适应性要求；当信道繁忙或适应性路由出问题（如死锁）时，就使用逃逸信道（尽管此时出现阻塞，需要等待），确保了不会死锁
  - 一个例子：k元n维立方的维度逆转路由。

路由算法

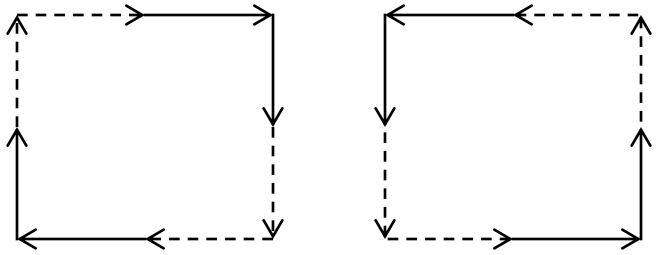
- 部分自适应和无死锁路由
  - 具备部分的适应性，且没有死锁。通常从决定性、无死锁路由着手，进行扩展，在其中加入部分的适应性。
  - 讨论：
    - 2维网格的转弯模型
    - k元n维立方的平面自适应模型
    - 部分自适应的e-立方路由
    - n维立方的转弯模型
    - 部分自适应的XY-路由（基于起源的路由）

路由算法

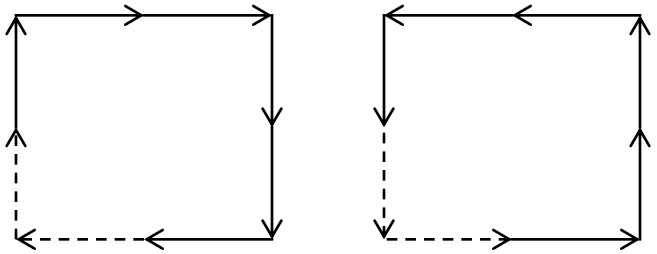
- 部分自适应和无死锁路由
  - 2维网格的转弯模型
  - 2维网格中会形成2种抽象回路



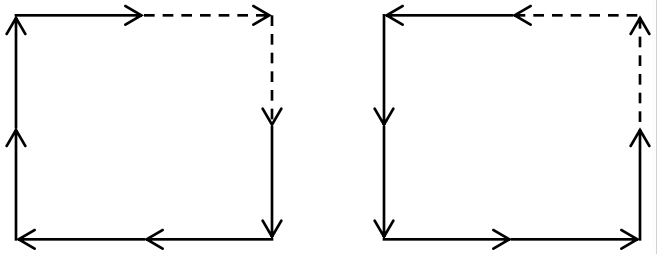
  - 其中包含8种转弯
  - 决定性的路由（如XY-路由）用到了4种转弯（先X方向、再Y方向的转弯）
  - 从而打破回路，不会产生死锁。但其没有适应能力



- 部分自适应和无死锁路由
  - 事实上，可以去掉回路中的一个（转弯）角来打破回路，同时具备部分适应性（有6个转弯）



正向优先路由
（从负到正，即西到北、南到东的转弯被禁止）

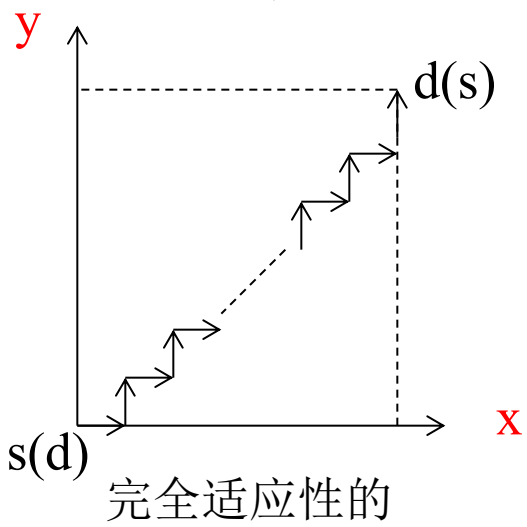


负向优先路由
（从正到负，即东到南、北到西的转弯被禁止）

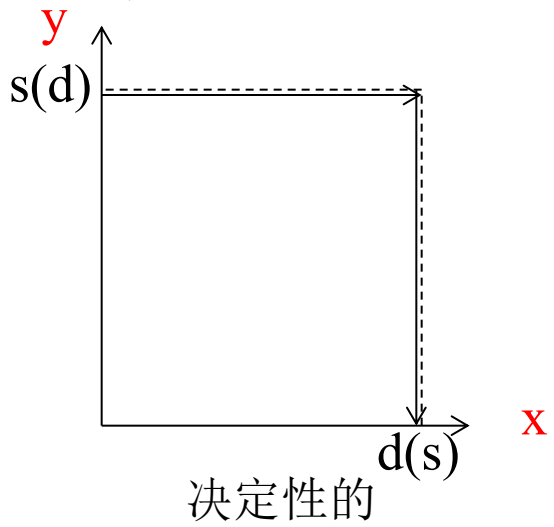
（上北下南、左西右东）

  - 转弯模型的基本思想就是通过禁止最少的转弯来增加适应性，同时避免回路

- 部分自适应和无死锁路由
  - 由于源和目标的位置的不同，转弯模型可能有适应性，也可能没有
  - 如果目标位于源的东北或西南，那么正向优先路由就是完全适应性的
  - 如果目标位于源的东南或西北，那么正向路由就是决定性的

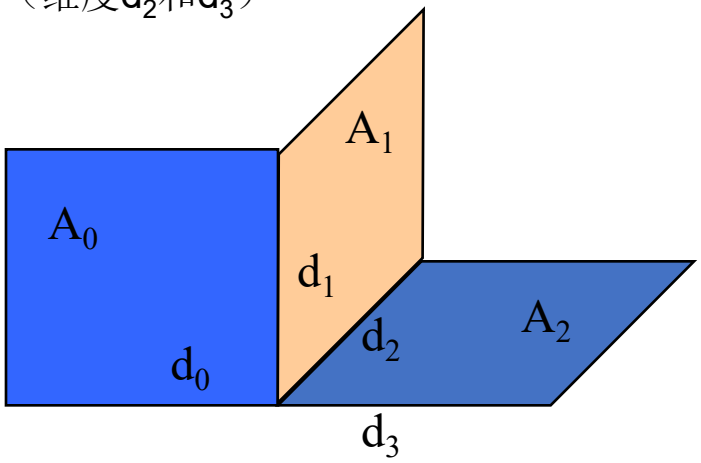


完全适应性的



决定性的

- 平面自适应模型
  - k元n维立方的平面自适应模型
  - 基本思想：k元n维立方中，在某一段时间将路由的自由度限制到几个维度，从而既有一定的适应性，又不产生死锁
  - 如果每次只选2个维度，那么就是平面自适应模型：有$A_0, A_1, \ldots, A_n$这些平面，其中A_i在维度d_i和d_{i+1}方向扩展。如三个平面的例子：A_0（维度d_0和d_1），A_1（维度d_1和d_2），和A_2（维度d_2和d_3）



路由算法

- 平面自适应模型
  - 对每个平面A_i，引入3个虚信道，1个沿第i维，2个沿第(i+1)维。将第i维的虚信道分成2个单向信道，第(i+1)维的2个虚信道分开，从而形成这个平面（相当于2维网格）上的2个子网：一个正网络、一个负网络
  - A_i平面内部的适应性路由可以使用前述2维网格的正负网络路由来实现：依据源和目标的位置不同而选用正网络或者负网络。如果目标的标识大于源标识，则选用正网络；否则，选用负网络
  - 相对于完全适应而言，平面适应方法牺牲了一些路由自由度（适应性），但大大降低了虚信道的数目
  - 相对于决定性路由而言，平面适应方法在一个平面中路由时具有适应性

路由算法

- 扩展e-立方路由
  - 对一个超立方中的任意回路，总能找出沿着某个维度（设为第i维）的2个链接，其中一个链接是从第i位为0的节点到第i位为1的节点（正向链接），另一个是从第i位为1的节点到第i位为0的节点（负向链接）
  - 为了打破一个回路，禁止从较高维度的正向链接向沿着维度i的负向链接转换，除非这个转换符合e-立方路由。从而扩展了e-立方路由的限制，增加了适应性

路由算法

- 扩展e-立方路由
  - 即：如果e-立方路由满足维度递增的顺序，一个沿着维度$dim(c_1)$的信道c_1到一个沿着维度$dim(c_2)$的信道c_2的转换是允许的当且仅当下述条件中的一个为真：
  - $dim(c_1)<dim(c_2)$（满足e-立方要求），或
  - c_2是正向的
  - 假定维度是从右向左编号的（最右边的一位对应于维度1），那么变换：（10010，00010）→（00010，00000）是不允许的；变换（10010，10110）→（10110，11110）（满足条件1和条件2）和变换（10010，10110）→（10110，00110）（满足条件1）是允许的

路由算法

- 扩展e-立方路由
  - 将转弯模型扩展到n维立方中。
  - n维立方中，假定$s=s_n s_{n-1}\ldots s_1$和$d=d_n d_{n-1}\ldots d_1$是源和目标节点。设S是s和d所不同的维度的集合。S被分为S_1（正向转换集合）和S_2（负向转换集合）。即，如果$s_i=0$且$d_i=1$则$i\in S_1$，如果$s_i=1$且$d_i=0$则$i\in S_2$
  - 路由分成两个阶段：在第一阶段，消息按照任意顺序沿着集合S_1中的维度路由；在第二阶段，消息按照任意顺序沿着集合S_2中的维度路由。（两阶段期间都具有适应性，但都没有回路！）

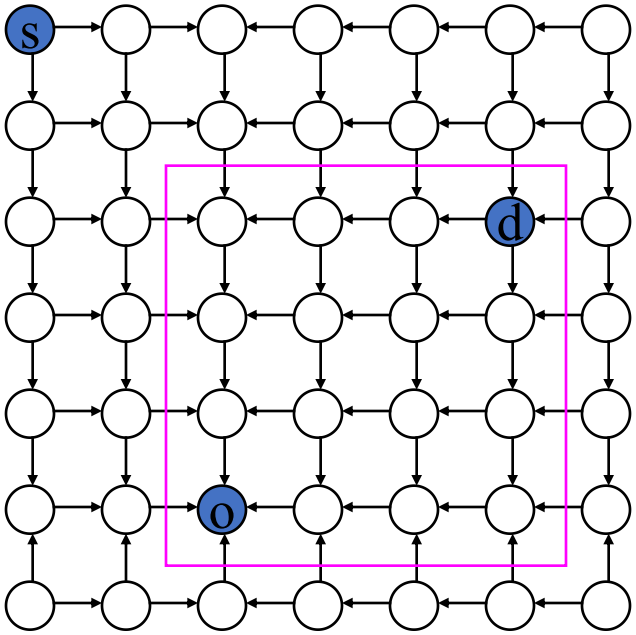
路由算法

- n维立方的转弯模型
- 例：如果s=0101和d=1010，那么S_1={2，4}、S_2={1，3}。下面的路由路径是合法的：
  - 0101→1101→1111→1011→1010
  - 0101→0111→1111→1011→1010
  - 0101→1101→1111→1110→1010
  - 0101→0111→1111→1110→1010
  - 适应性分析：n维立方中，一个完全适应性的路由算法中，有|S|!种不同的路由选择。使用本扩展转弯模型，这个数字是$|S_1|!\times|S_2|!$。如：当s=0101和d=1010时，有|S|=4，$|S_1|$=2，$|S_2|$=2。使用完全适应性路由算法有|S|!=4!=24种路由选择，而使用本模型有$|S_1|!\times|S_2|!$ =2!×2!=4种路由选择

路由算法

- 基于起源的路由
  - 基于起源的路由是2维网格XY-路由的一个扩展
  - 在2维网格中预先选定一个起源节点o，网络被分成2个子网：IN子网（包括所有指向起源节点o的单向信道）和OUT子网（包括所有离开起源节点o的单向信道）。以起源节点o和目标节点d为对角顶点的矩形建立一个outbox

outbox包含了所有位于目标节点d和起源节点o之间的最短路径上的节点。



outbox

图中仅仅显示了IN子网。可以通过将图中的每个链接反向来得到OUT子网。

路由算法

- 基于起源的路由
  - 路由分成两个阶段。阶段1是从源s到起源节点o，阶段2是从起源节点o到目标节点d。路由的第一阶段使用IN子网，第二阶段使用OUT子网。（两阶段中都没有回路！）
  - 确切地说，基于起源的路由使用IN子网将消息从源向目标区域靠近（以得到部分适应性）；一旦消息到达outbox边界，就将切换为OUT子网向目标节点d转发（也有部分适应性）

路由算法

- 基于起源的路由
  - 路由分成两个阶段。阶段1是从源s到起源节点o，阶段2是从起源节点o到目标节点d。路由的第一阶段使用IN子网，第二阶段使用OUT子网。（两阶段中都没有回路！）
  - 确切地说，基于起源的路由使用IN子网将消息从源向目标区域靠近（以得到部分适应性）；一旦消息到达outbox边界，就将切换为OUT子网向目标节点d转发（也有部分适应性）

路由算法

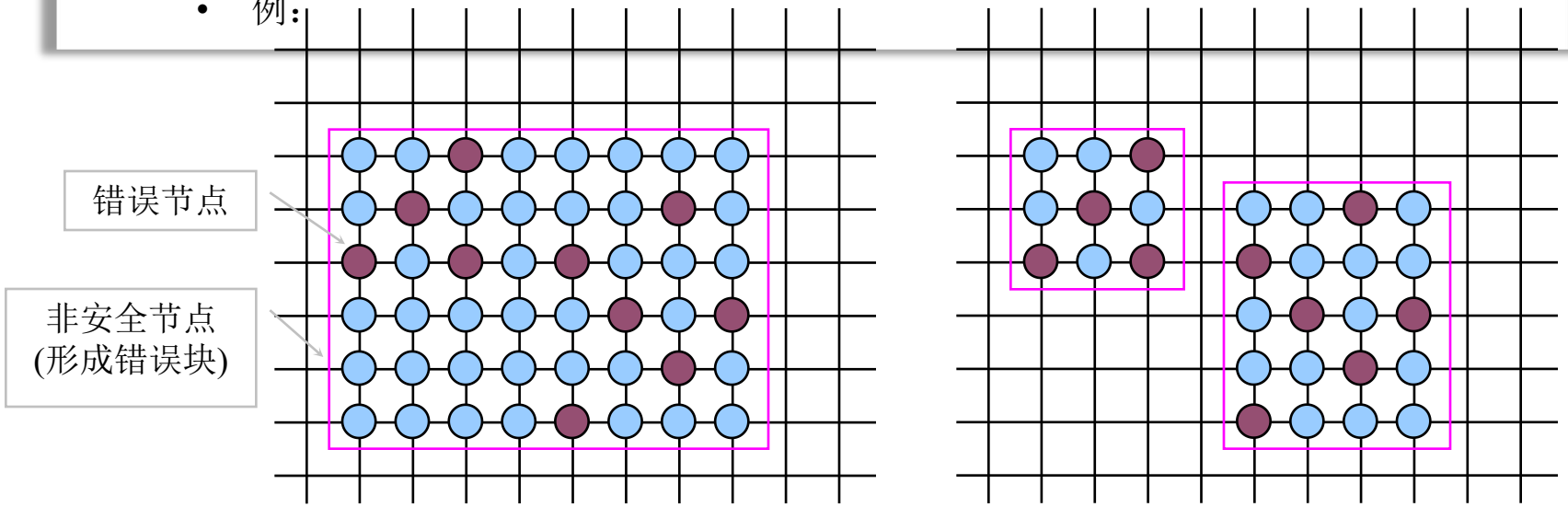
- 容错单播
- 容错单播：不增加空节点和/或链接，通过使用网络中已有的冗余来达到容错。具有一定的适应性，且保证无死锁
- 容错单播路由算法主要考虑：
  - 最短路径或非最短路径
  - 错误类型：链接错、节点错或者二者都有
  - 出错的组件的个数是有限的还是无限的
  - 对错误分布的了解：局部、全局和有限的全局
  - 冗余或非冗余
  - 回退或者前进

路由算法

- 2维网格和圆环中的容错单播—基于局部信息的路由
- 在2维网络中，如果源和目标都有四个邻居，那么2维网的冗余路由可以至少经受三个错误（链接和/或节点）。 所以，2维网络可以实现容错路由
- 将2维网络的错误节点框定在一个凸起的区域（称为错误块），以便路由时绕过错误块，而方便实现2维网络的容错路由
- 错误块框定目标：包含所有错误节点，且框进去的节点数尽量少。有2种方法：
    - 基本安全/非安全节点定义
        - 所有错误节点都是不安全的。所有非错误节点开始时都是安全的
        - 如果一个非错误节点有两个或两个以上的非安全邻居，那么它的状态就变为非安全的
        - 可以证明，基本错误块是分离的且它们间的距离最少是3

路由算法

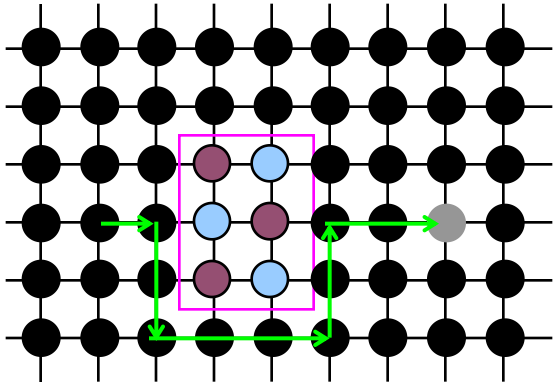
- 2维网格和圆环中的容错单播－基于局部信息的路由
- 扩展安全/非安全节点的定义
  - 所有错误节点都是不安全的。所有非错误节点开始时都是安全的
  - 如果一个非出错节点在两个维度上都有错误的或不安全的邻居，那么它的状态就变为非安全的
  - 可以证明，扩展错误块是分离的且它们间的距离最少是2
  - 例：



基本错误块　　扩展错误块（节点数少）

路由算法

- 2维网格和圆环中的容错单播－基于局部信息的路由
- 框定错误块后，在2维网络上进行容错路由：
- 1.如果没有因为错误块阻塞，那么就按照无错的情况进行路由
- 2.如果在X维（或Y维）阻塞，那么在Y维（或X维）路由
- 3.如果是在X维度受到阻塞，而Y维度的距离已经接近零了，就有必要进行曲折路由。随便选择一个Y方向，开始曲折路由。首先试着从X维度到达目标。继续沿着X方向，直到Y方向正确为止。就是说，沿着出错块的边缘路由。（Y维情况类似）



路由算法

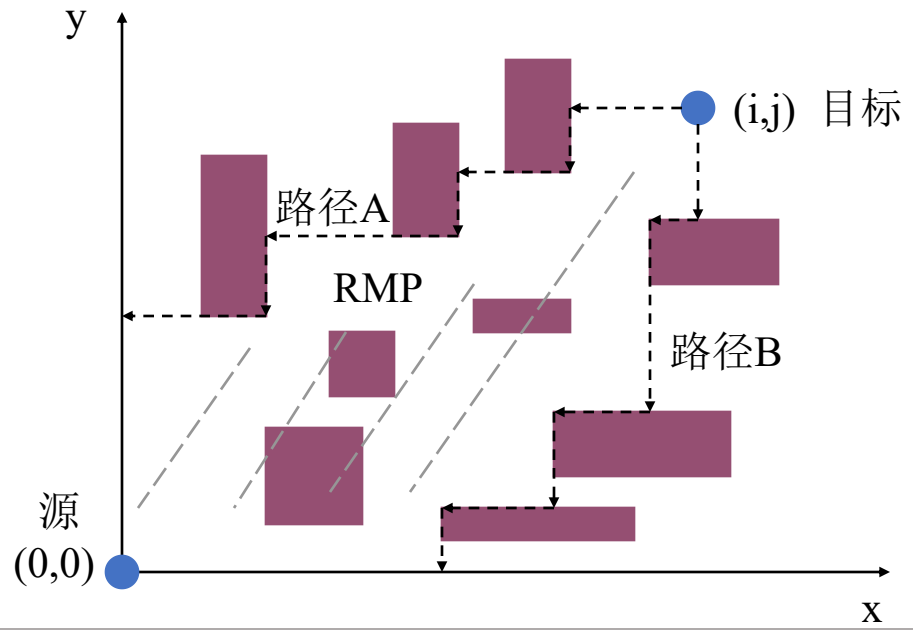
- 2维网格和圆环中的容错单播－基于有限全局信息的路由
- 在掌握了有限全局信息的条件下，Wu提出了有错误块的2维网格的最优容错路由算法
- 首先，可以证明：2维网格中，设节点(0,0)是源节点，节点(i,j)是目标节点。如果没有错误块通过X和Y轴，那么至少存在一个始于(0,0)的最优路径，长度为|i|+|j|
- 定义：
  - 如果这一结果对任意位置的目标节点，任何数目和分布的错误块都是成立的，相应的源叫做**安全的**
  - 扩展上述结果和定义，允许X和Y轴有出错块，只要它们到源的距离分别大于|i|和|j|就行。这样的源节点叫做**扩展安全的**（这里的安全节点的概念和在错误块中使用的有所不同。）

路由算法

- 2维网格和圆环中的容错单播—基于有限全局信息的路由由
- 有2个最优和适应性容错的路由算法：面向目标路由和面向源路由。
- 为简单起见，假定源节点是(0,0)，目标节点是(i,j)，且i,j≥0。就是说，路由永远向东北方向。
- 在面向目标路由中，引入最短路径区域（region of minimal paths, RMP）：对于给定的目标和源，RMP包括了最短路径的所有中间节点

路由算法

- 2维网格和圆环中的容错单播—基于有限全局信息的路由
- 构建RMP：
  - 路径A：做一条从目标节点(i, j)开始到Y轴结束的线，当遇到错误块时，先向南绕过错误块，然后继续向西
  - 路径B：同样，做一条从(i, j)开始向南到X轴截止的线，当遇到错误块时，先向西绕过错误块，然后继续向南



路由算法

- 2维网格和圆环中的容错单播－基于有限全局信息的路由
- 当源是扩展安全时，使用面向目标路由：
  - 源通过一个最优或不是最优的路径向目标发送一个信号
  - 接到信号后，目标发送2个信号：一个向西，一个向南。向西的信号建立RMP的路径A，向南的信号建立RMP的路径B
  - 一旦源收到来自目标的2个信号，就意味着RMP已经建立起来了。源节点就用任何一种适应性最小路由发送路由消息
  - 遇到路径A（或路径B）的边界时，剩下的路由就要沿着路径A（路径B）发往目标

路由算法

- 2维网格和圆环中的容错单播－基于其他错误模型的路由
- 利用矩形错误块模型实现2维网络的容错路由很简单。但矩形错误块框进去较多的非出错节点，这些非出错节点在路由中不起作用，这损失了一些适应性
- 将错误块模型扩展为直角凸多边形模型，进一步缩小了错误块的范围，将更多的非出错节点加入到路由中
- 直角凸多边形模型使用了活跃/不活跃节点的概念：
  - 所有出错节点都是不活跃的，所有安全节点都是活跃的
  - 一个非安全节点开始的时候是不活跃的，但如果它有两个或两个以上的活跃邻居，那么它就是活跃的
  - 因此，对于一个非出错节点，有三种可能情况：1) 安全且活跃；2) 非安全但活跃；3) 非安全且不活跃
  - 所有不活跃的节点组成一个直角凸多边形，作为错误块标

路由算法

- 超立方中的容错单播—基于其他错误模型的路由
  - 已经证明，对n维立方中的任何2点u和w，如果H(u,w)=k，那么从u到w有n条点分离路径。这些路径中，有k条长度为k的路径，有(n-k)条长度为k+2的路径
  - 如果出错组件的数目L小于n，那么从u到w就至少存在1条可用路径。也就是说，这时的容错单播路由是可能的

路由算法

- 超立方中的容错单播—基于其他错误模型的路由
  - 已经证明，对n维立方中的任何2点u和w，如果H(u,w)=k，那么从u到w有n条点分离路径。这些路径中，有k条长度为k的路径，有(n-k)条长度为k+2的路径
  - 如果出错组件的数目L小于n，那么从u到w就至少存在1条可用路径。也就是说，这时的容错单播路由是可能的

路由算法

- 超立方中的容错单播—基于其他错误模型的路由
- 4种基于局部信息的容错单播算法：
  - 1）出错组件小于n，不确保有最优路径
  - 2）出错组件小于n，确保有最优路径
  - 3）出错组件无限制，不确保有最优路径
  - 4）出错组件无限制，确保有最优路径
  - 我们讨论第1种算法，首先定义：
    - 等位序列$[d_1,d_2,\ldots,d_k]$（初始时也叫首选维度）列出了当前节点与目标节点不同的所有维度。如，0010和0111是当前节点（源节点）和目标节点，那么相应的等位序列（首选维度）就是[1,3]
    - 空余维度：那些没有在首选维度中出现的维度。算法中以空余维度标记记录那些可用的空余维度。如上面的空余维度是[2,4]，其标记是0101。

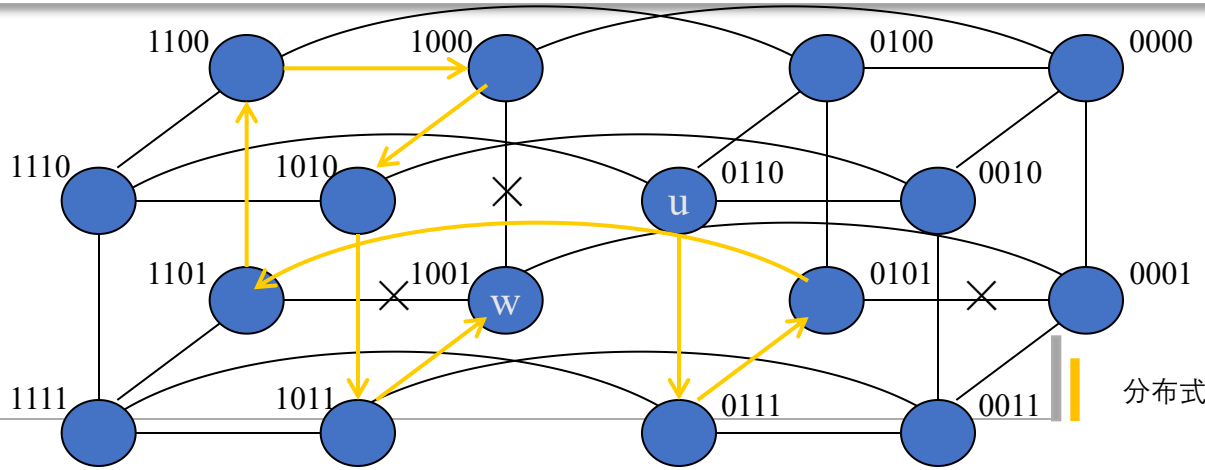
路由算法

- 超立方中的容错单播—基于其他错误模型的路由
- 算法如下：
  - 为了表示一个消息的目标，等位序列要和消息一起传送。空余维度标记（源节点发起时，设为全0）也要一起传送，以便用这些维度来绕过出错组件
  - 因此，消息表示为（$k,[d_1,d_2,\ldots,d_k]$(等位序列),消息,标记(空余维度)）。其中k是剩余路径的长度，当k=0时，消息到达目标。等位序列和空余维度标记在路由过程中随着当前节点的变化也将不断更新
  - 当节点收到一个消息时，节点检查k的值以便知道本节点是否是目标。如果不是，节点将尝试按照剩下的等位序列中指定的维度发送消息，以努力沿着最短路径传送。然而，如果通往最短路径的维度的那个链接出错，这个节点将使用空余维度通过另外的路径发送消息

路由算法

- 超立方中的容错单播－基于其他错误模型的路由
- 例：
  - 假设消息m从u=0110路由到w=1001。在u=0110处的最初消息是(4,[1,2,3,4],m,0000)。节点0110将(3,[2,3,4],m,0000)发送给0110^1＝0111，随后节点0111将(2,[3,4],m,0000)发送给0101。由于0101的第3维链接出现错误，节点0101将发送(1,[3],m,0000)到1101。然而，由于1101的第3维的链接出现错误，节点1101将使用第1维（此时空余维度标记=0100）发送消息(2,[3,1],m,0101)到1100。1100发送(1,[1],m,0101)到1000。但节点1000的第1维链接又出现错误，这时使用第2维（此时空余维度标记=0101）发送(2,[1,2],m, 0111)到1010。之后，消息将经过1011到达目标1001

路径长度＝8，显然不是最优路由。

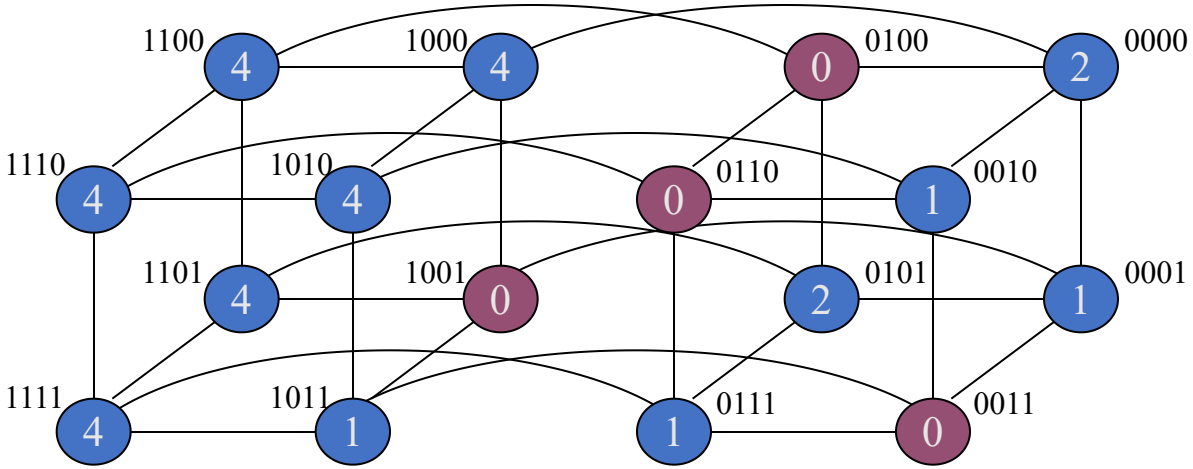


路由算法

- 超立方中的容错单播－基于其他错误模型的路由
- 安全等级是有限全局信息，它实际上就是每个节点周围邻居中失效节点的大致数目
- 定义：设S(a)=k是节点a的安全等级，这里k被称为安全等级，a被称为是k-安全的。一个失效节点是0-安全的，即最低的安全等级；而n-安全的节点是安全级别最高的，也叫安全节点；如果k≠n，那么一个k-安全的节点就是不安全的
- 设$(S_0, S_1, S_2, \ldots S_{n-1})$，$0 \leqq S_i \leqq n$，是n维立方中节点a的邻居节点的安全等级的非递减安全状态序列，即$S_0 \leqq S_i \leqq S_{i+1} \leqq S_{n-1}$。节点a的安全状态定义如下：如果$(S_0, S_1, S_2, \ldots, S_{n-1}) \geqq (0, 1, 2, \ldots, n-1)$，那么S(a) =n；否则，如果$(S_0, S_1, S_2, \ldots S_{k-1}) \geqq (0, 1, 2, \ldots, k-1) \wedge (S_k=k-1)$，那么S(a)＝k

路由算法

- 超立方中的容错单播—基于其他错误模型的路由
- 如下图的出错超立方中，节点1000，1010，1100，1110，1101和1111是安全的（4-安全的）；出错节点0011，0100，0110和1001是0-安全的；节点0001，0010，0111和1011是1-安全的，因为它们每个都有2个出错邻居节点，安全状态序列是（0,0,x,x）；节点0000和0101是2-安全的

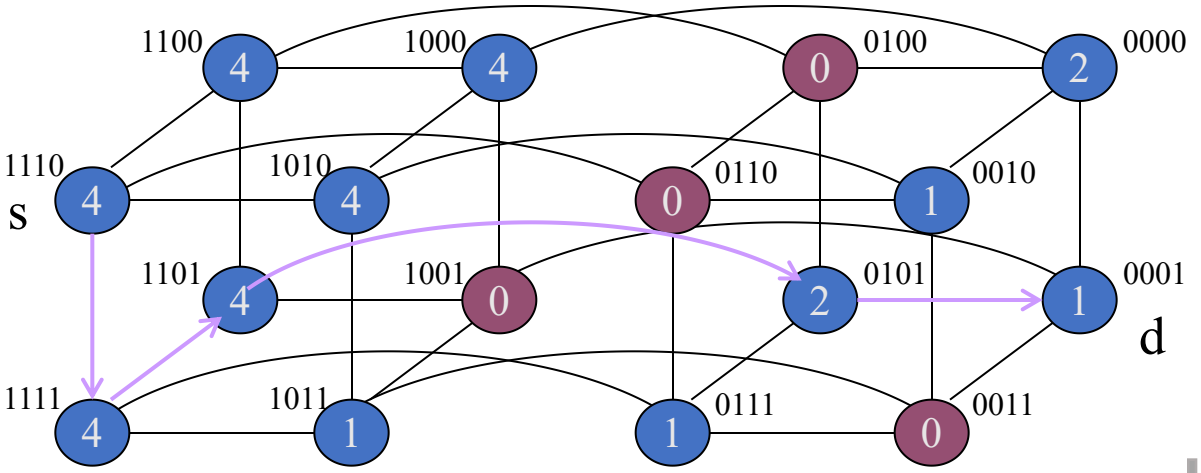


路由算法

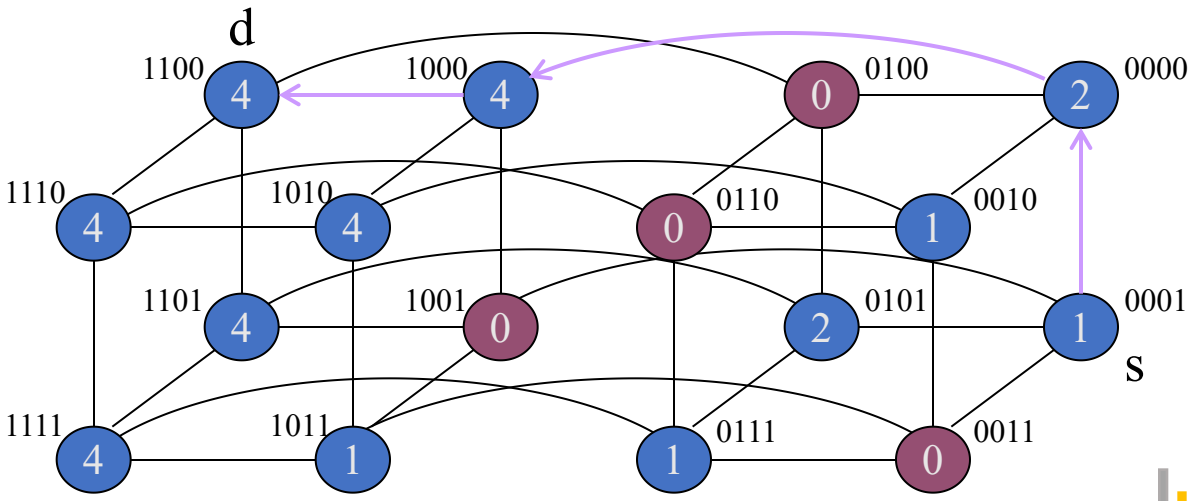
- 超立方中的容错单播－基于其他错误模型的路由
  - 安全等级有以下性质：如果一个节点的安全等级是k（0<k≦n），那么在k海明距离内，至少存在一个从该节点到任意节点的海明距离路径。也就是说，当源的安全等级大于或等于源和目标之间的海明距离的时候，就可以保证最优路由。
  - 实际上，可以在每一步通过选择具有最高安全等级的邻居（相同时，随机选择一个）来产生最优路径。一个引导向量（定义为当前节点和目标节点的按位异或）被附加在路由消息上面

路由算法

- 超立方中的容错单播—基于其他错误模型的路由
- 下图例：源s=1110向d=0001单播。引导向量N=s⊕d=1111，H(s,d)=4，源节点s的安全等级是4，可以使用最优路由。在源节点的首选转发节点中，1111作为3个具有最高安全等级的邻居节点之一被随机选中。引导向量N对应的该位置为无效(清0)后也和消息一起被发送。其后，1111根据引导向量有效位的指示和最高安全等级选择原则，选择邻居节点1101转发。同样，1101选择0101转发，0101选择0001转发到目标



- 超立方中的容错单播—基于其他错误模型的路由
- 事实上，当源节点的安全等级比源s和目标d之间的海明距离H(s,d)=|s⊕d|小的时候，根据引导向量有效位的指示，只要存在一个安全等级不小于H(s,d)-1的邻居，仍然可以通过将消息转发到这个节点来实现最优路由。否则，如果存在一个安全等级不低于H(s,d)+1的空闲邻居节点（引导向量无效位上的邻居），也可以通过将消息转发到这个节点实现次优路由，结果路径的长度是H(s,d)+2。例：



路由算法

- 容错广播
  - 提出了n维立方中一个使用全局信息进行广播的有效算法（广播结构在源节点根据全局信息就可以确定）。算法前提：错误类型是链接错误，错误数目≤n-1
  - 算法包含以下两步：
    - 1）分离过程：在源节点s决定（分离出）等位序列（算法1）
    - 2）容错广播过程：根据得到的等位序列进行广播（算法2）

路由算法

- 容错广播
- 算法1 {分离过程}
  - /*源节点s在Q_n中，初始时m=1*/
  - 1.随机选择一个维度i_m，保证Q_{n-m+1}中的第i_m维没有出错链接（如果初始错误数目≤n−1，这一点总是做得到）。Q_{n-m+1}沿着第i维分成了（Q_{n-m}, Q_{n-m}'）。如果不存在这样的维度，则返回“不可行”
  - 2.如果Q_{n-m+1}中的所有出错链接都位于Q_{n-m}（或Q_{n-m} '）中，则停止并返回（T_{nonop}，$i_m i_{m-1} \ldots i_1$）（或者（T_{op}，$i_m i_{m-1} \ldots i_1$）），这里$i_m\ i_{m-1} \ldots i_1$是通过第一步得到的维度序列。（注意$i_m\ i_{m-1} \ldots i_1$的顺序和它们被选择的顺序是相反的。）否则，将m加1，对Q_{n-m}（包含s）重复步骤1和2，直到m=n为止
  - 通过算法1的过程，将分离出一个完全无错的子立方

路由算法

- 容错广播
- 算法2 {容错广播过程}
  - /*（类型，$i_m\ i_{m-1}\ldots i_1$）是由算法1确定的，s在Q_{n-m}中 */
  - 1.（最优路由）如果类型=T_{op}，Q_{n-m}中没有出错链接，那么可以对Q_{n-m}使用一般基于二分树的广播而没有限制，在Q_{n-m}外部的广播，依据维度顺序cs= $i_m\ i_{m-1}\ldots i_1$
  - 2.（非最优广播）如果类型=T_{nonop}，则Q_{n-m}'中没有出错链接，使用如下步骤：
    - a) s沿着第i_m维将广播数据发送给它的邻居（记为$s^{(im)}$）
    - b) 在Q_{n-m}'中使用基于一般二分树的广播，$s^{(im)}$是源节点
    - c) 在Q_{n-m}中进行广播（s除外）：将广播信息从Q_{n-m}'中沿着第i_m维发送到Q_{n-m}中的相应节点
    - d) 在Q_{n-m+1}＝Q_{n-m}+ Q_{n-m}外部的广播，依据维度顺序cs= $i_{m-1}\ i_{m-2}\ldots i_1$。

路由算法

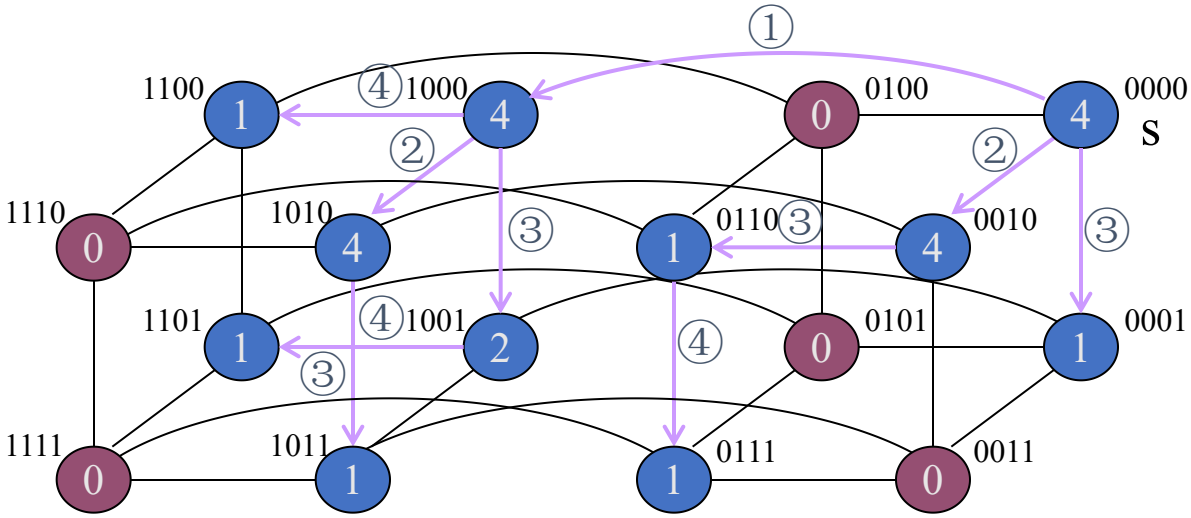
- 容错广播—使用安全等级(有限全局信息)进行广播
  - 本广播算法基于建立一个受伤生成二分树（所有出错节点都是生成树的叶子）
  - 同时，本广播算法也基于安全等级的如下性质：如果一个节点是k-安全的，那么在任何一个l子立方中（l不大于k），存在一个以这个节点为根的l层的受伤生成二分树。因此，如果源节点是n-安全的，那么在n立方中就存在一个n层的受伤生成二分树

路由算法

- 容错广播—使用安全等级(有限全局信息)进行广播
  - 算法实现：
  - 在每个节点保存2个序列：递减安全状态序列dss=（$D_0,D_1,D_2,\ldots,D_{n-1}$）和相应的邻居维度序列nds=（$d_0,d_1,d_2,\ldots,d_{n-1}$）
  - 使用一个n位控制字LABEL来标识分区中的一个子立方，每一位都是一个二进制数，为1的位表示子立方中的一维

路由算法

- 容错广播—使用安全等级(有限全局信息)进行广播
  - 例：如图有出错节点的Q_4，源节点s=0000，每个节点的dss和nds见P154表7-1。算法应用结果如下图（节点上标识了安全等级，链接旁圆圈数字标识了广播的步骤）
  - 算法实际的结果：沿着一个安全状态最好的邻居的维度，建立一个受伤生成二分树， 最后所有出错节点都是生成树的叶子



THANK YOU