文章编号:1000 - 582X(2006)03 - 0069 - 07

局部扭曲立方体单播容错路由算法*

苏 伟,杨 小 帆,唐 荣 旺,陈 文 斌
(重庆大学 计算机学院,重庆 400030)

摘 要:局部扭曲立方体是一种新型的网络拓扑结构.基于此网络拓扑结构,利用安全级概念以及此种网络拓扑结构自身特有的性质设计了一种单播容错路由算法.通过模拟仿真实验对该算法进行了性能评价与分析.当故障节点的数目达到或超过一半时,仍能保持在一个相当高的容错路由成功率上.另外,该算法所选线路在多数情况下是最短距离.

关键词:互连网络;局部扭曲立方体;容错路由
中图分类号:TP301 **文献标识码**:A

现在很多的商用或实验性多处理机系统都采用直接互连网络连接系统中的各个处理机节点.由于多处理机系统的规模越来越大,系统中出现处理机故障或处理机间的链路故障的可能性也随之增加.因此设计较好的容错路由策略,尽可能多地记录系统中存在的最优通路的信息,使得当系统中存在故障的情况下实现更有效的容错路由,达到提高整个系统性能的目的,越发显得重要.对于常用的一种网络拓扑结构,即超立方体,由于它直径小、结构对称、可扩展性强,近年来在针对它的容错路由设计方面,人们做了大量的研究工作.而其中网络中的单播容错路由问题又是网络容错路由中的最为基本的问题.单播即是指找出一条从源节点 u 到目的节点 v 之间可以传送信息的正确节点组成的路径的问题.

在现有算法中,一类最简单的算法是仅要求选择此链路不会形成环路,而节点也仅有它的 n 条邻接链路(节点)的状态信息.这类算法一般是基于局部故障信息的,其中以 Chen[1] 的深度优先搜索(DFS)算法最为典型.它可以保证不论系统中有多少故障,只要消息的源、目的节点之间存在通路,它都能把消息传递到目的节点.这种方法的问题在于大量的消息沿大于(在某些情况下甚至是远远大于)实际存在的最短通路长度的路径传递.

另一类算法则是要求所选通路是消息的源、目的节点间的最短通路,这种算法要求每个节点记录系统中的所有故障信息,即它是基于全局故障信息的.其中 Min[2] 给出的双向搜索算法就属于这一类型.这类算法的困难在于存储和维护全局的故障信息需要大量的存储和通信开销.

还有一类算法则是前两类的折衷,即减弱对所选链路性质的要求,例如仅要求所选通路的长度小于等于两节点间的海明距离加 2,而并不要求最短.这类算法都是基于对故障信息的预处理,通过与邻接节点的信息交换计算每一节点的信息情况.而其中的关键就是如何保证通过计算在每个节点中记录尽量多的关于最优通路的信息.由于它只记录所存在的最优通路中的一部分,因此这类算法的核心问题就是如何减少所舍去的最优通路.其中[3-11]属于这一类算法.Min[3] 采用动态学习的方法,通过记录各邻接节点的故障链路的数目,给路由算法以启发性的信息,从而提高了消息沿最优通路传递的概率.但它所能找到的最优通路与系统中实际存在的最优通路还有很大差距.Lee 和 Hayes[4] 首先提出了不安全节点的概念,并由此得到了一个简单的路由算法.它在每个节点记录邻接节点的状态,即安全、不安全或故障.在故障节点数小于 $n/2$ (n 为超立方体的维数)时,可以在任何无故障节点间传递消息,且路径长度不超过两节点之间的海明距离加 2.Wu[5] 和 Chiu[6] 修改了不安全节点的概念,使这

* 收稿日期:2005 - 11 - 10
基金项目:重庆市自然科学基金课题资助(CSTC,2005BB2191)
作者简介:苏伟(1980 -),男,山西长治人,重庆大学硕士研究生,主要从事并行计算和路由算法研究.

—机制的容错能力达到 $n-1$ 个故障节点,在任何无故障节点间传递消息的路径长度小于等于两节点之间的海明距离加 4. 后来 Wu[7-9] 和 Chiu[10] 分别提出了安全级、安全向量和路由能力的概念,这两个概念都是用来表示系统中各节点沿最优通路传递消息的能力. 它们共同的特点是能以较少的步数把消息传递到目的节点,并且所选通路的长度在源节点处即可确定. 随后 Shih[11] 基于不安全节点设计的虫孔容错路由算法也只具有故障节点不超过 $\lceil n/2 \rceil$ 的容错能力. 从不安全节点到安全向量,再到路由能力,这类算法的容错能力和沿最优通路传递消息的能力都有了一定提高. 但是当故障节点数目变得很大时,上述几个算法都不同程度地表现出其路由效果不是很理想.

而对于超立方体的变体,即交叉立方体. Agrawal[12] 在 1996 年利用维序对其进行容错路由,所提出的算法也仅仅是只在考虑故障节点数目小于 n 的情况下,并未涉及到大量故障节点存在的情况.

笔者在只考虑故障节点存在的情况下,基于安全级的概念[7],设计了一种关于局部扭曲立方体网络(由 Yang[13] 在 2004 年提出)的单播容错路由算法,该算法是比较有效的,当故障节点的数目达到或超过一半时,能仍保持在一个相当高的容错路由成功率上,其中绝大部分等于非故障情况下源目的节点间最短路径的长度,这也就保证了不大于海明距离的情况也占有很高的比例,且平均容错路由的路径长度也小于其直径. 之所以要基于局部扭曲立方体,是因为它作为一种新型的网络拓扑结构,超立方体的一种变体,保持了超立方体的很多优点,对称性,递归性等. 另外其直径也大约只有超立方体的一半,在路由能力方面强于超立方体,拥有更好的容错和嵌入属性. 目前对局部扭曲立方体容错路由的研究还没有一种算法提出. 虽然当故障节点比例太大时,要找出最优路径可能不是线性时间就可以解决的问题. 但作者所提算法还是有意尽可能去找最优路径. 因此等于非故障情况下源目的节点间最短路径长度的占有很高的比例,且不大于海明距离的情况所占的比例也很高.

1 预备知识

1.1 局部扭曲立方体

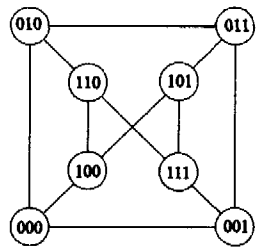
对于文中所提到的图论术语请参阅相关图论专著. n 维局部扭曲立方体简称为 LTQ_n,它和 n 维超立方体 Q_n 同构,具有相同数目的节点和边. 将 LTQ_n 中的每个节点用一个 n 位二进制编码 $x_0x_1\cdots x_i\cdots x_{n-1}$ 来表示,其中 $x_i \in (0, 1)$,下标 i 也代表维.

定义 1[13] 当 $n \geqslant 2$,LTQ_n 被定义如下:

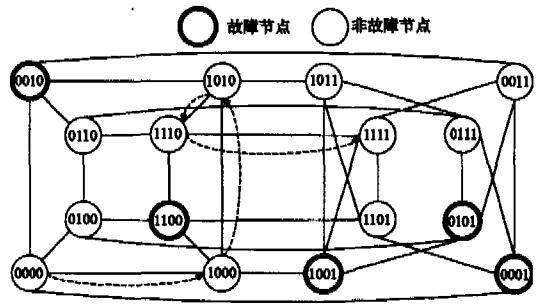
1) LTQ_2 由标记为 00、01、10 和 11 的 4 个节点,以及 4 条边(00, 01)、(00, 10)、(01, 11)和(10, 11)组成.

2) 当 $n \geqslant 3$ 时,LTQ_n 由两个相同的 LTQ_{n-1} 按照如下步骤获得. 在其中一个 LTQ_{n-1} 的每个节点标记前添加 0,记为 $0LTQ_{n-1}$;在另一个 LTQ_{n-1} 的每个节点标记前添加 1,记为 $1LTQ_{n-1}$. 然后将 $0LTQ_{n-1}$ 中的每个节点 $0x_2x_3\cdots x_n$ 和 $1LTQ_{n-1}$ 中对应的节点 $1(x_2+x_n)x_3\cdots x_n$ 相连,其中'+'表示模 2 加.

图 1 是局部扭曲立方体的两个例子展示. 对于 LTQ_n 中相邻的节点,它们的标记最多有 2 位相继位不同.



(a) LTQ_3 当 $n=3$



(b) LTQ_4 当 $n=4$

图 1 LTQ_n 当 $n=3,4$

另外,将 Q_{n-1}($n-1$ 维超立方体)的所有节点标记末尾添加 0,记为 $Q_{n-1}0$;将 $Q_{n-1,2}$($n-1$ 维 2—扭曲立方体,定义见[13])的所有节点标记末尾添加 1,记为 $Q_{n-1,2}1$. 则沿 $n-1$ 维 LTQ_n 可被分成 $Q_{n-1}0$ 和 $Q_{n-1,2}1$. 图 1(b)可看出 LTQ_4 由 Q_30 和 $Q_{3,2}1$ 构成.

对于 $0 \leqslant i \leqslant n-1$,用 $e_{n,i}$(简称为 e_i)来表示长度为 n 的二进制字符串,其第 i 位为 1,其余位全部为 0,也即:

$e_0 = 1000\cdots00$, $e_1 = 0100\cdots00$, $e_2 = 0010\cdots00$, $\cdots$, $e_{n-1} = 0000\cdots01$.

对于 $0 \leqslant i \leqslant n-3$,用 $E_{n,i}$(简称为 E_i)来表示长度为 n 的二进制字符串,其第 i 和 $i+1$ 位为 1,其余位全部为 0. 另外,$E_{n-2} = e_{n-2}$, $E_{n-1} = e_{n-1}$. 也即:

$E_0 = 1100\cdots00$, $E_1 = 0110\cdots00$, $E_2 = 0011\cdots00$, $\cdots$, $E_{n-2} = 0000\cdots10$, $E_{n-1} = 0000\cdots01$.

故在 LTQ_n 中,若节点 x 属于 $Q_{n-1}0$,可以用 e_0, e_1,

…,e_{n-1}来表示节点 x 的第 0,1,…,$n-1$ 维,即节点 x 的第 i 维邻节点为 $x+e_i$. 若节点 x 属于 $Q_{n-1,2}1$,也可以用 E_0, E_1,…, E_{n-1}来表示节点 x 的第 0,1,…,$n-1$ 维,即节点 x 的第 i 维邻节点为 $x+E_i$,其中 $0\leqslant i\leqslant n-1$. 在此用 e 来代表维 e_0,e_1,…,e_{n-1},用 E 来代表维 E_0, E_1,…, E_{n-1}. 以后的维叙述都用此种方式.

在此将 Yang[13] 中所提及的算法介绍如下,它将在笔者所提出的算法中用到.

```
Procedure Scan(x = x0x1…xn-1){
  if(x = 0^n) return(∅);
  else{
    r = min{ i:xi = 1};
    case n-2≤r≤n-1:
      return(Scan(x+er) ∪ { er} );
    case(r≤n-3) && (xr+1 = 0):
      return(Scan(x+er) ∪ {er} );
    case(r≤n-3) && (xr+1 = 1):
      return(Scan(x+Er) ∪{Er});
    }
}
Procedure e_Scan(x = x0x1…xn-1) {
  if(x = 0^n) return (∅);
  else{r = min{ i: xi = 1 };
       return(e_Scan(x+er) ∪ {er} ); }
}
Procedure E_Scan(x = x0x1…xn-1) {
  if(x = 0^n) return(∅);
  else{r = min{i: xi = 1 };
       return(E_Scan(x+Er) ∪ { Er} ); }
}
```

例如在 LTQ_6 中,令节点 $x=000000$,$y=101101$,其 $x+y=101101$,则 $Scan(x+y)=\{e_0, E_2, e_5\}$,若节点 x 要到达节点 y,必须经过维 e_0、e_5 以及维 E_2. 令节点 $x=000000$,$y=111100$,其 $x+y=111100$,则 $e\_Scan(x+y)=\{e_0, e_1, e_2, e_3\}$. 令节点 $x=000001$,$y=111101$,其 $x+y=111100$,则 $E\_Scan(x+y)=\{E_0, E_2\}$. 故 2 个节点间的最短路径长度可以通过上述算法求得.

1.2 安全级

安全级的概念在 1995 年首次被 Wu[7] 提出,它能够集中地反映出在互连网络拓扑结构中每个节点的相邻故障节点的分布和数量,是进行容错路由的有力工具. 对于节点 a,用 $S(a)$ 来表示此节点的安全级,如果 $S(a)=k$,则称作节点 a 是 k – safe. 对于故障节点,它们都是 0 – safe.

定义 2[7]　令$(S_0, S_1, S_2, \cdots, S_i, \cdots, S_{n-1})$为节点 a 的 n 个相邻节点的安全级的排序,且 $0\leqslant S_i\leqslant n$, $S_i\leqslant S_{i+1}$及 $0\leqslant i\leqslant n-1$. 节点 a 的安全级被定义如下:如果$(S_0, S_1, S_2, \cdots, S_{n-1})\geqslant(0, 1, 2, \cdots, n-1)$,则 $S(a)=n$;如果$(S_0, S_1, S_2, \cdots, S_{k-1})\geqslant(0, 1, 2, \cdots, k-1)\wedge(S_k=k-1)$,则 $S(a)=k$.

下面的算法能够计算出 Q_n 中所有节点的安全级,也可以将它应用在 LTQ_n 中,计算出 LTQ_n 中所有节点的安全级.

```
Algorithm  Global_Status(GS)[7]
{
  初始化所有非故障节点安全级为 n,所有故障节点为 0,round = 1;
    while(round≤Δ)
    {
    Node_Status(a(i)), 0≤i≤2^n - 1;
    round = round + 1;
    }
}
Procedure Node_Status(a(i))
{
  if(S0, S1, S2, …, Sn-1)≥( 0, 1, 2, …, n-1)
      S(a) = n;
  else if(S0, S1, S2, …, Sk-1)≥ (0, 1, 2, …, k-1)∧(Sk = k-1)
      S(a) = k;
}
```

此算法已被 Wu[7] 证明在 $\Delta=n-1$ 次循环之后,所有节点的安全级都处于一个稳定状态. 故取 $\Delta=n-1$. 在此规定节点 a 的 n 个相邻节点的安全级的升序排列为$(S_0, S_1, S_2, \cdots, S_{n-1})$,简称为 ASS;其对应的相邻节点的维的排序为$(d_0, d_1, d_2, \cdots, d_{n-1})$,简称为 NDS. 对于图 1(b)中含有故障节点的 LTQ_4,应用 GS 算法之后,可以得到所有节点的 ASS 和 NDS. 如表 1 所示.

表 1　ASS 和 NDS

节点	ASS	NDS	节点	ASS	NDS
0000	0,0,1,1	3,2,1,0	0001	0,1,1,1	3,2,1,0
0010	1,1,2,4	3,2,1,0	0011	0,0,0,2	3,2,1,0
0100	0,0,1,2	3,0,1,2	0101	0,1,1,1	0,3,2,1
0110	0,1,1,4	1,2,3,0	0111	0,0,2,2	2,1,3,0
1000	0,0,1,4	3,1,0,2	1001	0,1,2,2	0,3,2,1
1010	0,1,2,4	0,2,3,1	1011	0,1,1,4	2,0,1,3
1100	1,1,1,4	3,1,0,2	1101	0,0,2,2	3,0,2,1
1110	0,2,2,4	2,3,0,1	1111	0,1,1,4	1,2,0,3

2 容错路由

2.1 容错路由算法

在算法介绍前,假设 LTQ_n 中的每个节点的安全级,ASS 以及 NDS 都已通过算法 GS 得到. 该文的算法是基于局部扭曲立方体自身特有的性质,将节点安全级考虑进来而设计的. 该算法在保证容错路由成功的基础上,还能够使路由距离尽可能地达到最小.

在容错过程中,首先找到一条源节点到达目的节点尽可能最优的消息传播路径,然后再将消息沿此路径从源节点传递到目的节点. 在容错路径的寻找过程中,首先将源节点的父节点记录为 NULL(即空值),并放入路由列表 List(其负责存放已经路由过的非故障节点),然后从源节点开始进行容错路由. 分 2 种情况:①当前节点属于 $Q_{n-1}0$. 首先求得在非故障情况下当前节点到达目的节点所必须经过的维 M(即最短路径). 然后在当前节点从 M 中选择一个 e 维,此 e 维所对应的邻节点(List 中的节点除外)的安全级最大,将被选的邻节点加入到 List 中,并将被选中的邻节点记录为当前节点的子节点,当前节点记录为被选中的邻节点的父节点,然后此被选中的邻节点也即变为当前节点;若在当前节点,M 中所有的 e 维对应的邻节点(List 中的节点除外)的安全级都为 0,则首先从 M 中选择一个 E 维,且和此 E 维本身的维数相同的 e 维所对应的邻节点(List 中的节点除外)的安全级最大,将被选的邻节点加入到 List 中,并将被选中的邻节点记录为当前节点的子节点,当前节点记录为被选中的邻节点的父节点,此被选中的邻节点也即变为当前节点;若此种情况不存在,则再从当前节点的邻节点中选择安全级最大(不为 0)的节点(List 中的节点除外),将被选的邻节点加入到 List 中,并将被选中的邻节点记录为当前节点的子节点,当前节点记录为被选中的邻节点的父节点,此被选中的邻节点即变为当前节点;若上述情况都不存在,则回朔到当前节点的父节点,即当前节点的父节点再次成为当前节点. ②当前节点属于 $Q_{n-1,2}1$. 首先求得在非故障情况下当前节点到达目的节点所必须经过的维 M(即最短路径). 然后在当前节点从 M 中选择一个 E 维,此 E 维所对应的邻节点(List 中的节点除外)的安全级最大,将被选的邻节点加入到 List 中,并将被选中的邻节点记录为当前节点的子节点,当前节点记录为被选中的邻节点的父节点,此被选中的邻节点即变为当前节点;若在当前节点,M 中所有的 E 维对应的邻节点(List 中的节点除外)的安全级都为 0,则选择当前节点的邻节点中安全级最大(不为 0)的节点(List 中的节点除外),将被选的邻节点加入到 List 中,并将被选中的邻节点记录为当前节点的子节点,当前节点记录为被选中的邻节点的父节点,此被选中的邻节点即变为当前节点;若上述情况都不存在,则回朔到当前节点的父节点,即当前节点的父节点再次成为当前节点.

若容错路径存在,此过程直到当前节点变为目的节点为止,然后将信息从源节点传播到目的节点;若容错路径不存在,则此过程直到当前节点回朔到源节点的父节点为至,此时容错路由失败. 算法描述如下:

```
//Msg 为信息,x,y 为源、目的节点
Algorithm FT_Route (Msg, x = x0x1…xn-1, y = y0y1…yn-1)
{/*说明:x→parent 和 x→child 分别表示在容错路由过程所产生路径中的节点 x 的父节点和子节点*/
  s = x; d = y;          //源、目的节点
  //源节点的父节点赋为空值
  s→parent = NULL;
  /*List 负责存放已经路由过的非故障节点,初始为空*/
  List = ?;
  List = List + {s};
  /*寻找源、目的节点间尽可能最优的容错路径*/
while(s≠d  &&  s≠NULL)
    Node_Route(s, d, List);
  If(s = NULL)
    Return(failure);  //路由失败
  Else{
      s = x; d = y;
      /*将信息 Msg 从源节点传播到目的节点*/
      while(s≠d)
      {Send Msg from s to s→child;
        s = s→child; }
    }
}
Procedure Node_Route(x, y, List)
{
  M = Scan(x + y);
  switch(xn-1, yn-1)
  {
  case(0, 0): Me = e_Scan(x + y);
              If(|Me| ≤ |M| + 2)
                  M = Me;
              Else M = M + {en-1, en-1};
              break;
  case(1,1): ME = E_Scan(x + y);
              If(|ME| ≤ |M| + 2)
                  M = ME;
              Else  M = M + {En-1, En-1};
              break;
```

```
case(0, 1): break;
case(1, 0): break;
}
/*当前节点 x ∈ Q_{n-1}0 的情况*/
If(x_{n-1} =0 )
  for(j = n - 1  downto  0)
    /* d_j ∈当前节点 x 的 NDS, S_j ∈当前节点 x 的 ASS, x 的第 d_j维邻节点的安全级不为 0*/
    If(e_dj  ∈M  &&  S_j≠0 )
    {If(x + e_dj ∈List)  continue;
     Else
     {next_node = x + e_dj;
      x→child = next_node;
      next_node(parent = x;
      x = next_node;
      List = List + {x};
      break; }
    }
    /* x 的第 d_j维邻节点的安全级为 0*/
    Else If(e_dj ∈M  &&  S_j = 0)
    {Tag = 0;
     /*找一个同 E 维本身维数相同的 e 维所对应的安全级最大的未曾路由过的邻节点*/
     for(i = n - 1  downto  0 )
       If ( E_di ∈M  &&  S_i ≠0)
       {/*因为 E_di = e_di + e_di+1, 且 0≤d_i≤n - 3*/
        If(x + e_di ∈List)continue;
        Else
        {next_node = x + e_di;
         x→child = next_node;
         next_node→parent = x;
         x = next_node;
         List = List + {x};
         Tag = 1;  break; }
       }
     /*找一个安全级最大的未曾路由过的邻节点*/
     If(Tag = 0)
     {for(i = n - 1  downto  0 )
        If(S_i = 0 )  //回溯到父节点
        {x = x→parent; break; }
        Else If(x + e_di ∈List)
                continue;
             Else
             {next_node = x + e_di;
              x→child = next_node;
              next_node→parent = x;
              x = next_node;
              List = List + {x};
              break; }
      break;
     }
}
```

```
/*当前节点 x∈Q_{n-1,2}1 的情况*/
Else If(x_{n-1} = 1)
  for(j = n - 1  downto  0 )
  /* d_j ∈当前节点 x 的 NDS, S_j ∈当前节点 x 的 ASS, x 的第 d_j维邻节点的安全级不为 0*/
    If(E_dj ∈M  &&  S_j≠0 )
    {If(x + E_dj ∈ List)  continue;
     Else
     { next_node = x + E_dj;
       x→child = next_node;
       next_node→parent = x;
       x = next_node;
       List = List + {x};
       break; }
    }
    /* x 的第 d_j维邻节点的安全级为 0*/
    Else If(E_dj ∈M  &&  S_j = 0 )
    {/*找一个安全级最大的未曾路由过的邻节点*/
     for(i = n - 1  downto  0)
       If(S_i = 0 )  //回溯到父节点
       {x = x→parent; break; }
       Else If(x + E_di ∈List)
               continue;
            Else
            {next_node = x + E_di;
             x→child = next_node;
             next_node→parent = x;
             x = next_node;
                 List = List + {x};
             break; }
     break;
    }
}
```

例,看图 1(b)和表 1,令 $x=0000, y=1111$,源节点 x 将发送信息 Msg 到目的节点 y. 首先记录源节点的父节点 x→parent = NULL,将 x 加入 List 中,即List = {0000}. 当前节点为 $x=0000$,首先通过 Scan($x+y$)计算,得到 $M=\{E_0, e_2, e_3\}$,节点 $x=0000$ 的 NDS 为(3, 2,1,0),AAS 为(0,0,1,1),$e_{d1}=e_2 \in M$ 且 $S_1=0$,故

考虑$\{E_0\}$,由于$e_{d3}=e_0$对应的邻节点安全级$S_3=1$,故选择第0维,next_node$=x+e_0=1000$,节点$x\rightarrow$child = next_node = 1000,节点 next_node→parent = $x=0000$, x = next_node = 1000, List = List + {1000} = {0000, 1000}. 当前节点为$x=1000$,通过 Scan$(x+y)$计算,得到$M=\{E_1,e_3\}$,节点$x=1000$的 NDS 为(3,1,0,2), AAS 为(0,0,1,4),$e_{d0}=e_3\in M$且$S_0=0$,故考虑$\{E_1\}$,由于$e_{d1}=e_1$对应的邻节点安全级$S_1=0$,故选择安全级最大的邻节点对应的维$e_{d3}=e_2$,$S_3=4$,即选择第2维,next_node$=x+e_2=1010$,节点$x\rightarrow$child = next_node = 1010,节点 next_node→parent = $x=1000$,x = next_node = 1010, List = List + {1010} = {0000, 1000, 1010}. 当前节点为$x=1010$,通过 Scan$(x+y)$计算,得到$M=\{e_1,e_3\}$,节点$x=1010$的 NDS 为(0,2,3,1), AAS 为(0,1,2,4),$e_{d3}=e_1\in M$且$S_3=4$,故选择第1维,next_node$=x+e_1=1110$,节点$x\rightarrow$child = next_node = 1110,节点 next_node→parent = $x=1010$,x = next_node = 1110, List = List + {1110} = {0000, 1000, 1010, 1110}. 当前节点为$x=1110$,通过 Scan$(x+y)$计算,得到$M=\{e_3\}$,节点$x=1110$的 NDS 为(2,3,0,1), AAS 为(0,2,2,4),$e_{d1}=e_3\in M$且$S_1=2$,故我们选择第3维,next_node$=x+e_3=1111$,节点$x\rightarrow$child = next_node = 1111,节点 next_node→parent = x = 1110,x = next_node = 1111, List = List + {1111} = {0000, 1000, 1010, 1110, 1111}. 此时节点x即为目的节点y,故源节点到目的节点的信息传播容错路径找到. 然后将信息 Msg 沿 0000 -> 1000 -> 1010 -> 1110 -> 1111 进行传播. 如图1(b)中带箭头的虚线所示.

2.2 模拟实验结果评价与分析

为了评价此算法的性能,分别模拟了在 LTQ_6 和 LTQ_7 中存在故障节点的情况下(在 LTQ_6 中选择存在的故障节点的数目分别为0、5、10、15、20、25、30、35;在 LTQ_7 中选择存在的故障节点的数目分别为0、6、10、20、30、40、50、60、65.),利用该算法实现传递消息的能力. 对于每个给定的故障节点数,我们随机选择100种故障节点分布模式,对于每种分布模式都要在所有的源、目的节点对(任何两个不同的非故障节点)之间进行一次消息传递. 把所有在源、目的节点对之间传递的消息作为100%,统计了源、目的节点间成功实现了消息传递所占的比例,容错路由的路径长度等于非故障情况下源目的节点间最短路径的长度所占的比例,以及不大于海明距离所占的比例,同时也统计了平均容错路由的路径长度. 如图2(含不同故障节点数目时的平均消息传递成功率、容错路由的路径长度等于非故障情况下源目的节点间最短路径的长度所占的比例及小于海明距离所占比例)和图3所示.

从图2和图3中可以看到,当故障节点个数小于维数n时,平均消息传递成功率是100%.,其中等于

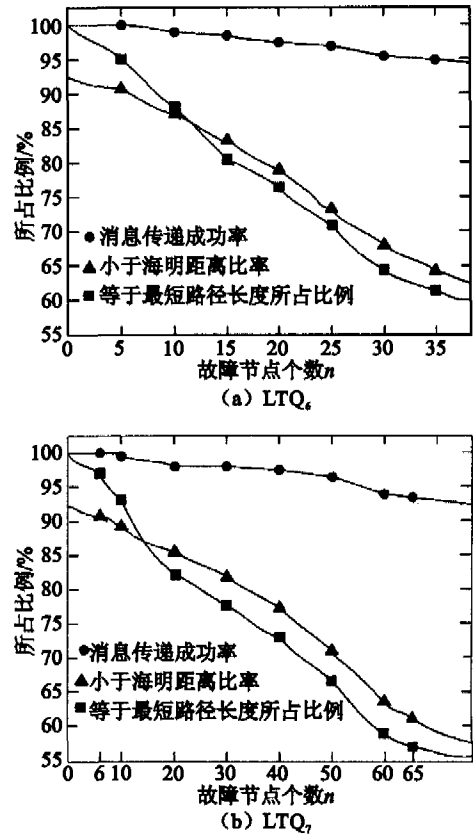


图2 在 LTQ_6 和 LTQ_7 中不同故障节点数目时的路由容错性能

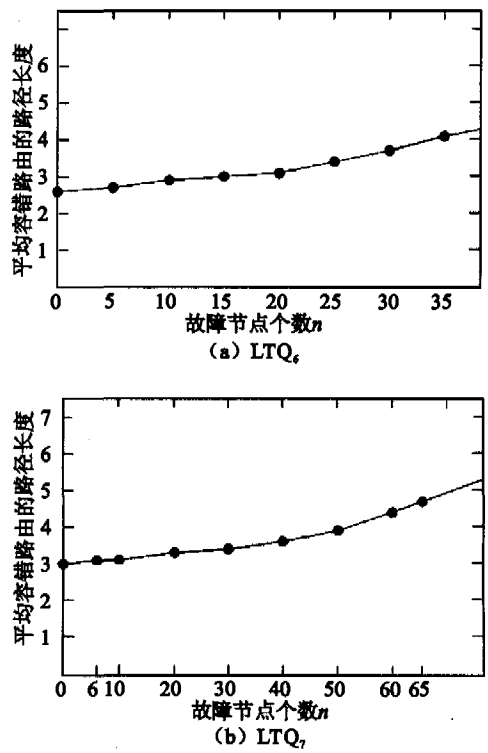


图3 在 LTQ_6 和 LTQ_7 中不同故障节点数目时的平均容错路由的路径长度

非故障情况下源目的节点间最短路径长度的占到了95%以上,小于海明距离所占的比例也在90%以上,而平均容错路由的路径长度仅为2.7和3.1,远小于它们的直径5. 当故障节点的数目达到总共节点数的

一半时,即 2^{n-1},平均消息传递成功率能仍保持在93%以上,其中等于非故障情况下源目的节点间最短路径长度的占到了 60% 左右,有一定的降低,但也保持了很高的比例,小于海明距离所占的比例在 60% 以上,而此时的平均容错路由的路径长度分别为 4 和 4.7,也小于它们的直径 5. 这说明该算法具有较高的容错能力,当有大量的故障节点出现时,仍能成功的实现消息传递,且保证容错路由的路径长度尽可能小.

3　结　论

基于一种新型的网络拓扑结构——局部扭曲立方体,在只考虑故障节点存在的情况下,利用安全级概念以及此种网络拓扑结构自身特有的性质设计了一种单播容错路由算法. 这是针对此种网络拓扑结构首次提出的关于容错路由的算法. 经过模拟实验结果的评价与分析,可知此算法具有较好的容错路由能力. 当故障节点的数目达到或超过一半时,仍能保持在一个相当高的容错路由成功率上,其中绝大部分等于非故障情况下源目的节点间最短路径的长度,这也就保证了不大于海明距离的情况也占有很高的比例,且平均容错路由的路径长度也小于其直径. 另外由于局部扭曲立方体的直径大约只有超立方体的一半,其在路由能力方面强于超立方体. 而对局部扭曲立方体容错路由的研究还很少,这有待进一步去研究发掘,设计出更好更有效的容错路由算法.

参考文献:

[1] CHEN M S, SHIN K G. Depth-first Search Approach for Fault-tolerant Routing in Hypercube Multicomputers [J]. IEEE Transactions on Parallel and Distributed Systems, 1990, 1(2): 152 - 159.

[2] MIN Y H, LI Z C, MIN Y L. Routing Algorithms in Injured Hypercubes [Z]. Proceedings of APPT '95, China: Beijing, 1995.

[3] MIN Y L, MIN Y H. A Fault-tolerant and Heuristic Routing Algorithm for Faulty Hypercubes [J]. Journal of Computer Science&Technology, 1995, 10(6): 536 - 544.

[4] LEE T C, HAYES J P. A Fault-tolerant Communication Scheme for Hypercube Computers [J]. IEEE Transactions on Computers, 1992, 41(10): 1 242 - 1 256.

[5] Wu J, Fernandez E B. Reliable Broadcasting in Faulty Hypercube Computers [J]. IEEE Transactions on Computers, 1992, 44(5): 122 - 129.

[6] CHIU G M, WU S P. A Fault-tolerant Routing Strategy in Hypercube Multicomputers [J]. IEEE Transactions on Computers, 1996, 45(2): 143 - 156.

[7] WU J. Safety Levels-an Efficient Mechanism for Achieving Reliable Broadcasting in Hypercubes [J]. IEEE Transactions on Computers, 1995, 44(5): 702 - 706.

[8] WU J. Reliable Unicasting in Faulty Hypercubes Using Safety Levels [J]. IEEE Transactions on Computers, 1997, 46(2): 241 - 247.

[9] WU J. Adaptive Fault-tolerant Routing in Cube-based Multicomputers Using Safety Vectors [J]. IEEE Transactions on Parallel and Distributed Systems, 1998, 9(4): 321 - 334.

[10] CHIU G M, CHEN K S. Use of Routing Capability for Fault-tolerant Routing in Hypercube Multicomputers [J]. IEEE Transactions on Computers, 1997, 46(8): 953 - 958.

[11] SHIH J D. Fault-tolerant Wormhole Routing for Hypercube Networks [J]. Information Processing Letters, 2003, 86(2): 93 - 100.

[12] AGRAWAL N, RAVIKUMAR C P. Fault-tolerant Routing in Multiply Twisted Cube Topology [J]. Journal of Systems Architecture, 1996, 42(4): 279 - 288.

[13] YANG X F, EVANS D J, MEGSON G M. The Locally Twisted Cubes [J]. International Journal of Computer Mathematics, 2005, 82(4): 401 - 413.

An Unicast Fault-tolerant Routing Algorithm on Locally Twisted Cubes

SU Wei, YANG Xiao-fan, TANG Rong-wang, CHEN Wen-bin

(College of Computer, Chongqing University, Chongqing 400030, China)

Abstract: The locally twisted cube is a newly topological structure of network. The authors we design a unicast fault-tolerant routing algorithm on the locally twisted cube by utilizing safety levels and the feature of the network. The performance of the proposed algorithm is evaluated through simulation experiments. When the number of faulty nodes reaches or exceeds half of the total, it can still achieve a quite high percentage of successful routing. An additional advantage of the routing algorithm is that it is highly probable that the selected route be a shortest route between the associated nodes.

Key words: interconnection network; locally twisted cube; fault-tolerant routing

(编辑　成孝义)