Mesh 网络耐故障虫孔路由

段新明　杨愚鲁

（南开大学信息技术科学学院计算机科学与技术系　天津 300071）

摘　要　耐故障是互连网络设计中的一个重要问题。本文提出了一种新的耐故障路由算法，并将其应用于使用虫孔交换技术的 Mesh 网络。由于使用了较低的路由限制，这一算法具有很强的自适应性，可以在各种不同故障域的 Mesh 网络中保持路由的连通性和无死锁性；由于使用了最小限度的虚拟通道，这一算法所需的缓冲器资源很少，非常适宜构建低成本的耐故障互连网络；由于根据本地故障信息进行绕行故障节点的决策，这一算法的路由决策速度较快并且易于在互连网络中实现。最后网络仿真试验显示，这一算法具有良好的平滑降级使用的性能。

关键词　Mesh 网络，路由算法，耐故障，无死锁

Fault-tolerant Wormhole Routing in Mesh

DUAN Xin-Ming　YANG Yu-Lu

(Department of Computer Science, Nankai University, Tianjin 300071)

Abstract　Fault-tolerance is an important issue for the design of interconnection networks. In this paper, a new fault-tolerant routing algorithm is presented and is applied in Mesh networks employing wormhole switching. Due to its lower routing restrictions, the presented routing algorithm is so highly adaptive that it is connected and deadlock-free in spite of the various fault regions in Mesh networks. Due to the minimal virtual channels it uses, the presented routing algorithm only employs as few buffers as possible and is suitable for fault-tolerant interconnection networks with low cost. Since it chooses the path around fault regions according to the local fault information, the presented routing algorithm makes routing decisions quickly and is applicable in interconnection networks. Moreover, a simulation is conducted for the proposed routing algorithm and the results show that the algorithm exhibits a graceful degradation in performance.

Keywords　Mesh networks, Routing algorithm, Fault-tolerance, Deadlock-free

1　引言

耐故障是并行计算机互连网络设计中面临的主要问题之一，耐故障指网络在部分部件失效的情况下正常运作的能力。现代网络设备是非常健壮的，但是在某些特殊的领域（如国防建设等），即使网络设备失效的概率非常小，也必须使网络具有耐故障工作的能力。

当网络发生故障时，增加路由的自适应性可以使消息绕过故障区域，然而在实际应用中，这可能会破坏路由算法的无死锁特性。一个 9×9 的 Mesh 网络如图 1 所示，假设消息使用西向优先路由算法[1]路由，如果网络中出现故障节点，消息需要使用非最短路径绕过故障节点以保持路由的连通性。消息可能需要绕过故障节点的 8 种模式如图 1 中带箭头的折线所示，其中折线的实线部分表示西向优先路由算法所允许的消息转弯，而虚线部分则表示算法所不允许的消息转弯。显然，为了使路由能够绕过南向、北向和西向的故障节点而增加路由算法的自适应度，会违反西向优先路由算法对路由所做的限制，从而导致死锁的发生。

虫孔交换技术由于其较低的资源需求以及较低的网络传输时延在互连网络中得到了广泛的应用[2]，但同时因为虫孔交换技术允许消息连续占据网络中的多条通道，所以一旦出现故障节点，虫孔网络发生死锁的概率很高，从而严重影响网络性能。设计耐故障的无死锁路由算法是解决这一问题的有效途径。

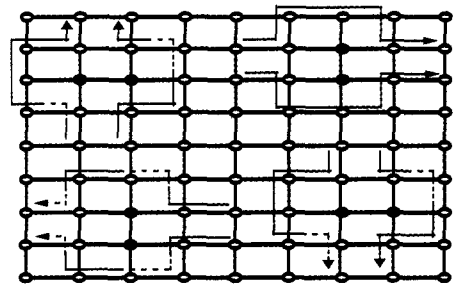

图 1　Mesh 中路由绕过故障节点的 8 种模式

针对于此，本文提出一种 Mesh 网络耐故障路由算法，这一算法能够适应网络中各种矩形故障域而保持路由的连通性，同时算法仅使用了最低限度的路由限制以及最低限度的虚拟通道。最后，这一算法基于本地故障信息指导路由决策，因而其路由决策更快并且更易于实现。

2　故障模型

互连网络中故障发生的类型、故障域的形状和对故障部件的处理决定了耐故障路由算法的结构以及算法实现的策

段新明　博士生，主要研究方向为并行机系统结构、可重构系统、NoC 系统；**杨愚鲁**　博士、教授、博士生导师，主要研究方向为网格、并行机系统、可重构系统等。

略。

通常，耐故障互连网络中每个节点都有自检和检测相邻节点的能力，而检测机制可以检测两种基本类型的故障：节点的故障和链路的故障。当一个节点发生故障时，所有与此节点连接的物理链路在相邻的路由器上被标记为故障。当一条物理链路发生故障时，这条链路上的所有虚拟通道都标记为故障。

单个节点或链路的故障是故障元件的最小单位。如图 2 所示，相邻的故障节点和链路将会连接成故障域，由于耐故障路由算法对凹形故障域的处理比较困难，因此需要对故障域的结构加以限制。在 Mesh 网络中，故障域的结构被限制为凸形，即矩形。实现矩形故障域的方法如下：当一个节点检测到 2 个或 2 个以上的相邻节点或链路发生故障时，它的状态也变为故障，这样的检测在经过有限的次数后，网络中就会形成矩形的故障域。

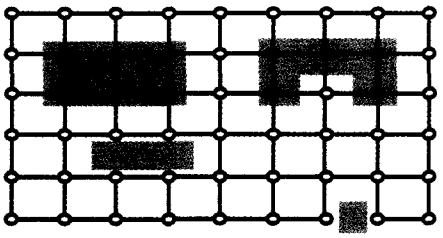

图 2　Mesh 中不同的故障域

对故障部件的处理也非常重要。故障节点由于不能再发送和接收消息，发往这些节点的消息会无限制地占用和阻塞缓冲器资源，导致新的死锁，因此必须由故障节点的邻居节点清除这些消息。

3　耐故障路由

在引出 Mesh 网络耐故障路由算法之前，先介绍无死锁路由的相关理论。

3.1　无死锁路由基础理论

定义 1　互连网络 I 是一个强连通的有向图 $I=G(N, C)$，其中 N 为 I 中所有节点的集合，C 为所有节点的输出通道集合，通道 $c_i \in C$ 的尾节点和头节点分别表示为 $tail(c_i)$ 和 $head(c_i)$。

定义 2　给定互连网络 I、路由算法 R 和 2 个通道 $c_i, c_j \in C$，如果存在一个路径，消息在离开 c_i 后立刻进入 c_j，则称 c_i 到 c_j 存在通道依赖。

定义 3　给定互连网络 I 和路由算法 R，R 的通道依赖图是一个有向图 $D_c=G(C, E_c)$，其中 C 为 I 中所有通道的集合，$c_i、c_j \in C$。如果 c_i 到 c_j 存在通道依赖，则通道的对 $\langle c_i, c_j \rangle \in E_c$。

引理 1　给定互连网络 I 和连通的路由算法 R，如果 R 的通道依赖图 D_c 中没有环存在，则 R 是无死锁的。

证明　Duato[3] 已经证明了引理 1。

由于 Duato 定理所使用的通道依赖图与互连网络的规模紧密相关，因此本文提出通道类的概念，通道类是由 Mesh 网络中所有具有相同方向以及相同虚拟通道编号的通道构成的。

定义 4　给定 Mesh 网络 I、路由算法 R 和 2 个通道类 cc_g，cc_h，如果 cc_g，cc_h 中分别存在通道 $cc_g.c_i$，$cc_h.c_j$，使得 $cc_g.c_i$ 到 $cc_h.c_j$ 存在通道依赖，则称 cc_g 到 cc_h 存在通道类依赖。

定义 5　给定 Mesh 网络 I 和路由算法 R，R 的通道类依赖图是一个有向图 $D_{cc}=G(CC, E_{cc})$，其中 CC 为 I 中所有通道类的集合，cc_g，$cc_h \in CC$，如果 cc_g 到 cc_h 存在通道类依赖，则通道类的对 $\langle cc_g, cc_h \rangle \in E_{cc}$。

定义 6　方向函数 $DR: CC \rightarrow P(x+, x-, y+, y-)$，其中 $P(x+, x-, y+, y-)$ 是 $x+, x-, y+, y-$ 的一个幂集，DR 返回通道类集合中通道类的方向。

定理 1　给定 Mesh 网络 I 和连通的路由算法 R，如果 R 的通道类依赖图 D_{cc} 中任意一个强连通分量 $D_{cc1}=G(CC_1, E_{cc1})$，满足 $DR(CC_1) \neq \{x+, x-, y+, y-\}$，那么 R 是无死锁的。

证明：

(i)在 Mesh 同一维中的路由是无死锁的。

(ii)当消息在 Mesh 的 2 个维中路由时，根据定理条件，存在 D_{cc1} 中的一组通道类 cc_g，$g=1, 2, \cdots, r$，r 为自然数，满足 cc_g 到 cc_{g+1} 存在通道类依赖(其中 $cc_{r+1}=cc_1$)。

又根据通道类的定义，存在 cc_g，$g=1, 2, \cdots, r-1$ 中的一组通道 c_g，$g=1, 2, \cdots, r-1$，满足 c_g 到 c_{g+1} 存在通道依赖以及 $head(c_g)=tail(c_{g+1})$。

设通道类 cc_g，$g=1, 2, \cdots, r$ 在 $\{x+, x-, y+, y-\}$ 各方向上的个数分别为 $\{a_{x+}, a_{x-}, a_{y+}, a_{y-}\}$，则 $head(c_r)=tail(c_1)+(a_{x+}+a_{x-}+a_{y+}+a_{y-})$。

根据定理条件 $DR(CC_1) \neq \{x+, x-, y+, y-\}$，显然 $head(c_r) \neq tail(c_1)$，即 R 的通道依赖图无环，又根据引理 1，定理 1 成立。

3.2　耐故障路由算法

为了在矩形故障域的 Mesh 网络中保持路由的连通性，耐故障路由算法必须具备如图 1 所示的 8 种转向模式，同时算法需要将每条物理通道分为 3 条虚拟通道，以避免绕行故障域时产生新的死锁。

不规则 Mesh 路由算法 R_{FT} 描述如下，其中 x_c、y_c 表示当前节点的 x 维、y 维坐标，x_d、y_d 表示目的节点的 x 维、y 维坐标，x_{offs}、y_{offs} 表示当前节点与目的节点在 x 维、y 维的偏移量。“around”是一个字位长的消息头控制域，其值为“$X+$”、“$X-$”、“$Y+$”和“$Y-$”，分别表示消息从 $X+$、$X-$、$Y+$ 和 $Y-$ 方向上绕过故障域。$x+$. channel≠null 表示在 $X+$ 方向上遇到故障域，其它方向类似。“prev”是消息头控制域，保存消息上一个跳步选择的路由通道。$s()$ 函数从多个通道中选择一个空闲通道。

Algorithm　R_{FT} Routing(x_c, y_c):

```
begin
    x_offs←x_d − x_c
    y_offs←y_d − y_c
    if x_offs=0 and y_offs=0 then
      吸收当前消息
    else
      if x_offs=0 and y_offs>0 then around←Y+
      if xoffs=0 and yoffs<0 then around←Y−
        if around=X+ then 绕行 X+故障域子函数
        if around=X− then 绕行 X−故障域子函数
        if around=Y+ then 绕行 Y+故障域子函数
        if around=Y− then 绕行 Y−故障域子函数
        endif
      end
        绕行 X+故障域子函数:
        if x+. channel≠null then
        chn←x_0+
      else
        if prev=y_1± then chn←prev else chn←s(y_1+, y_1−)
      endif
      绕行 X−故障域子函数:
```

```
    if x−.channel≠null then
    chn←x0−
    else
      if prev=y2±then chn←prev else chn←s(y2+, y2−)
    endif
    绕行 Y+故障域子函数:
    if y+.channel≠null and yoffs≠0 then
    chn←y0+
    else
      if prev=x1±then chn←prev else chn←s(x1+, x1−)
    endif
    绕行 Y−故障域子函数:
    if y−.channel≠null and yoffs≠0 then
      chn←y0−
    else
    if prev=x2±then chn←prev else chn←s(x2+, x2−)
endif
```

在 Mesh 网络耐故障路由算法中，消息按照先 X 维、后 Y 维的顺序绕过故障域，每一个消息首先根据其源节点和目的节点的偏移量被定为 X+或 X−类消息。X+或 X−类的消息首先从 X 维方向绕过故障域，直至消息的 X 维偏移量为 0，消息被重新标记为 Y+或 Y−类，然后消息从 Y 维方向绕过故障域，直至抵达目的节点。

Mesh 网络耐故障路由算法的通道类依赖图 $D_{cc}=G(CC, E_{cc})$ 如图 3 所示。

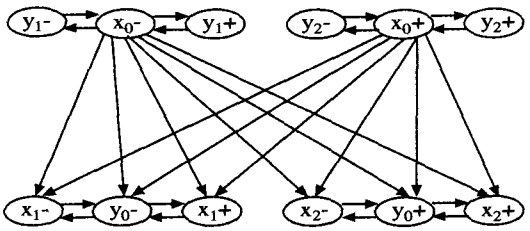

图 3　耐故障路由算法的通道类依赖图

定理 2　Mesh 网络耐故障路由算法是无死锁路由算法。

证明：Mesh 网络耐故障路由算法的通道类依赖图 $D_{\alpha}=G(CC, E_{\alpha})$ 如图 3 所示，通道类依赖图 D_{α} 的所有强连通分量如图 4 所示。

图 4 中 4 个强连通分量的通道类集合分别为 $CC_1=\{x_0-, y_1-, y_1+\}$、$CC_2=\{x_0+, y_2-, y_2+\}$、$CC_3=\{y_0-, x_1-, x_1+\}$、$CC_4=\{y_0+, x_2-, x_2+\}$。它们的方向集合分别为 $DR(CC_1)=\{x-, y-, y+\}$、$DR(CC_2)=\{x+, y-, y+\}$、$DR(CC_3)=\{y-, x-, x+\}$、$DR(CC_4)=\{y+, x-, x+\}$。显然对任意一个强连通分量，都有 $\{x+, x-, y+, y-\}\neq DR(CC_i)$ 成立(其中 $i=1, 2, 3, 4$)，根据本文提出的定理 1 可以证明耐故障路由算法是无死锁算法。

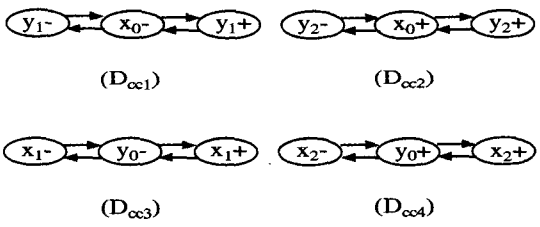

图 4　通道类依赖图的 4 个强连通分量

4　性能分析

Mesh 网络耐故障路由算法的通道类依赖图共有 12 个通道类(如图 3 所示)，如果删除图 3 中的通道类，路由算法将失去连通性，此时通过增加通道类的依赖使算法重新连通又会导致算法产生死锁(根据定理 1)，因此 Mesh 网络耐故障路由算法使用了最小限度的虚拟通道。本文提出的 Mesh 网络耐故障算法与 Boppana[4] 提出的耐故障路由算法的成本比较如表 1 所示，Boppana 算法需要将网络中每条物理通道分为 4 条虚拟通道，高于本文算法的 3 个。在路由算法中使用更少的虚拟通道将意味着在网络中使用更少的缓冲器，因此本文提出的路由算法具有更低的网络成本。

表 1　本文算法与 Boppana 算法的比较

	对故障域 Mesh 网络的连通性	虚拟通道数量/物理通道
Boppana 算法	连通	4
本文算法	连通	3

耐故障路由算法在一个 9×9 的 Mesh 网络中进行了仿真试验，在网络中没有故障节点、存在 5%的故障节点以及存在 10%的故障节点共 3 种情况下，路由算法的性能如图 5 所示。

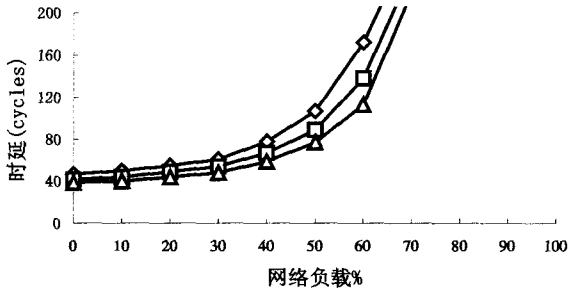

图 5　耐故障算法在不同故障域 Mesh 中的性能比较

由图 5 可以看到，随着 Mesh 网络中故障节点的增多，耐故障路由算法的传输时延缓慢地增加，网络系统能够随着故障域的扩大而平滑地降级使用。例如，在网络负载率为 40%的情况下，耐故障路由算法在 10%故障节点的 Mesh 中的传输时延仅比在无故障节点的 Mesh 中的传输时延高出 30%，因此本文提出的路由算法具有较强的耐故障性。

总结　耐故障是互连网络设计中的一个重要问题。本文提出一种新的 Mesh 网络耐故障路由算法，这一算法具有足够的自适应性，只要发生故障的网络在物理上是连接的，本文提出的算法总是连通的。此外，本文提出了新的基于通道类依赖图的无死锁路由判定条件，这一判定条件与网络的规模无关。通过这一无死锁路由判定条件，可以证明本文提出的 Mesh 网络耐故障路由算法是无死锁的，并且算法仅使用了最低限度的虚拟通道，最后网络仿真的结果显示，算法具有随着网络故障域的增大而平滑降级使用的性能，因此这一路由算法完全可以在耐故障 Mesh 网络中得到应用。

参 考 文 献

1　Glass C J, Ni L M. The turn model for adaptive routing. In: Proceedings of the 19th International Symposium on Computer Architecture, May 1992. 278～287

2　Ni L M, McKinley P K. A survey of wormhole routing techniques in direct networks. IEEE Computer, 1993, 26(2): 62～76

3　Duato J. A new theory of deadlock-free adaptive routing in wormhole networks. IEEE Trans on Parallel and Distributed Systems, 1993, 4(12): 1320～1331

4　Boppana R V, Chalasani S. Fault-tolerant wormhole routing algorithms for mesh networks. IEEE Transactions on Computers, 1995, 44(7): 848～864