Torus 网络自适应容错路由算法

段新明 武继刚 张大坤
(天津工业大学计算机科学与软件学院 天津 300387)
(中国科学院软件研究所计算机科学国家重点实验室 北京 100190)

摘 要 在应用于大规模并行计算机的互连网络的设计中,容错问题是其中的一个关键问题和难点问题。提出了一种基于 Torus 虫孔交换网络的容错路由算法,这一算法使用了矩形故障模型,无论故障区域大小多少和如何分布,算法始终是无死锁的,而且具有足够的自适应性,只要故障节点没有断开网络的连接,算法就能够通过选路使消息绕过故障区域,保持路由的连通性。同时,算法仅需要使用 3 个额外的虚拟通道。最后算法在不同故障率的 Torus 网络中进行了仿真实验,结果显示这一算法具有良好的平滑降级使用的特性。

关键词 带环网格,路由算法,容错,无死锁

Adaptive Fault-tolerant Routing in Torus Networks

DUAN Xin-ming WU Ji-gang ZHANG Da-kun
(School of Computer Science and Software, Tianjin Polytechnic University, Tianjin 300387, China)
(State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract Fault tolerance is one of the most dominant issues for the design of interconnection networks of large-scale multiprocessor systems. A new fault tolerant routing algorithm for wormhole torus network was proposed. The routing algorithm provides enough adaptability so that it is always connected as long as fault regions do not disconnect the network. In spite of the variety of fault components in torus, the proposed routing algorithm is always connected and deadlock-free. At the same time, the proposed algorithm only employs extra three virtual channels. The result of simulation shows that the proposed routing algorithm is of feasibility of gracefully degraded operation.

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

1 引言

Torus 网络被广泛应用于各种并行计算机系统中[1],而容错问题是并行计算机系统设计中的一个关键问题,容错是指并行计算机互连网络应对部件故障的能力,容错问题主要考虑的问题包括部件失效率、故障修复能力以及网络的平滑降级使用能力。网络故障诊断技术是一个具有挑战性的问题,本文假定这样的故障诊断技术已经存在,重点研究如何利用这些故障信息设计高性能的、可靠的容错路由算法。

当网络部件发生故障时,路由算法应该具有更高的自适应性,使消息能够绕过网络中的故障区域[2,3],但是这样往往会影响路由算法的无死锁特性。例如在一个 Torus 网络中使用西向优先路由算法[4]路由消息,一旦消息在路由的过程中遇到了故障区域,其必须使用非最短路径绕行故障区域以保持路由的连通性。如图 1 所示,消息绕行矩形故障区域包括由西向东(a)、由东向西(b)、由南向北(c)和由北向南(d) 4 种情况。8 种消息绕行矩形故障区域的方式如图 1 中 8 条带箭头的折线所示,折线中的实线部分表示西向优先路由算法允许的转弯,而虚线部分表示算法禁止的转弯。从图 1 中可以看到,消息在绕行西向、南向和北向的故障区域时使用的路径是西向优先路由算法所不允许的,增加消息的自适应性有可能带来死锁。

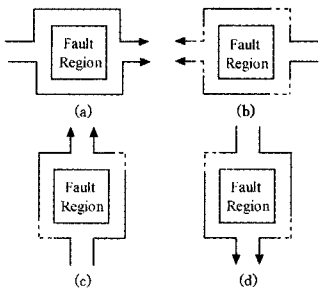


图 1 模板及其 ROI 区域

本文提出一种基于 Torus 虫孔交换网络的容错路由算法;第 2 节介绍算法使用的故障模型;第 3 节详细介绍 Torus 虫孔网络容错路由算法;第 4 节对这一算法进行仿真实验;最后总结本文。

2 故障模型

设计容错路由算法需要首先确定故障模型[6]。为了使用

到稿日期:2011-07-10 返修日期:2011-09-30 本文受国家自然科学基金(60970016)资助。
段新明(1970—),男,博士,副教授,主要研究方向为互连网络技术、片上网络技术,E-mail:duanxm@126.com;武继刚(1963—),男,博士,教授,主要研究方向为软硬件划分技术、并行计算、容错处理器重构技术;张大坤(1962—),女,博士,教授,主要研究方向为图形图像处理、虚拟现实技术。

最少的故障信息进行容错路由决策，本文采用得到广泛应用的矩形故障模型，具体描述如下。

1) 任何一个链路或节点都可能发生故障，这些故障是不可使用的，不能通过一个故障链路或故障节点传输消息。

2) 本算法所使用的故障模型是静态的，即在消息路由的过程中，不能有新的故障部件出现。

3) 消息的源节点和目的节点都没有故障。

4) 一旦一个节点发生故障，其内部的所有物理链路也将被标记为故障，这些故障信息被记录在其邻居节点。

5) 当一个物理链路发生故障时，共享这一物理链路的所有虚拟通道也将被标记为故障。

6) 如果一个节点的两个邻居节点或链路被标记为故障，那么这个节点也将被标记为故障，即使这个节点没有故障发生。这样经过有限步骤之后，故障节点和被标记为故障的非故障节点可以构成一个完整的矩形故障区域。

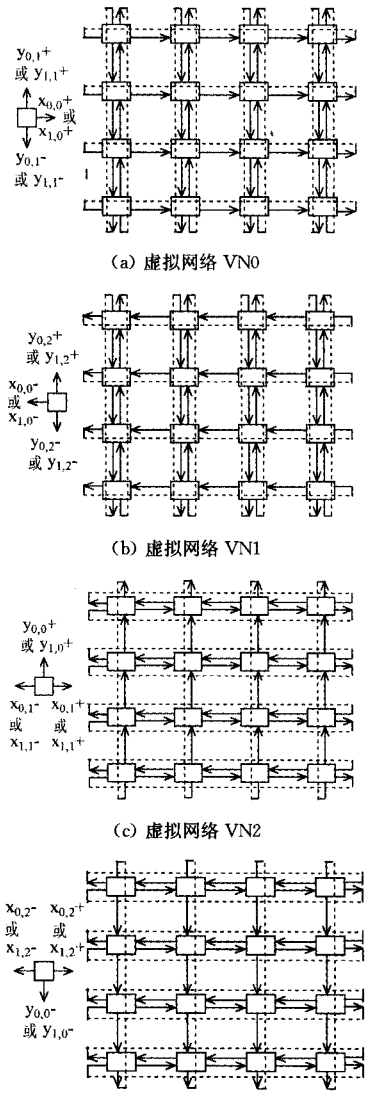
矩形故障区域可以降低容错路由算法的复杂度。

3 Torus 网络容错路由算法

在无故障的 Torus 网络中，由于存在回绕通道，消息在同一维中传输即有可能发生死锁。为了避免这种死锁，Torus 网络每一条物理通道需要分为 2 个虚拟通道，虚拟通道 0 和 1。当消息的目的节点坐标大于源节点坐标时，消息使用虚拟通道 0 路由，否则消息使用虚拟通道 1 路由。这 2 个虚拟通道组成 4 个虚拟网络 VN_{x0}，VN_{x1}，VN_{y0} 和 VN_{y1}，其中 VN_{x0} 由 X 维中虚拟通道 0 组成；VN_{x1} 由 X 维中虚拟通道 1 组成；VN_{y0} 由 Y 维中虚拟通道 0 组成；VN_{y1} 由 Y 维中虚拟通道 1 组成。消息使用 X-Y 路由算法在无故障的 Torus 网络中 X 和 Y 两个维上路由可能有 4 种情况，即由 VN_{x0} 到 VN_{y0}；由 VN_{x0} 到 VN_{y1}；由 VN_{x1} 到 VN_{y0} 以及由 VN_{x1} 到 VN_{y1}，这 4 种情况显然都不会产生新的死锁，消息在无故障的 Torus 网络中传输是无死锁的。

消息在有故障的 Torus 网络中路由时，可能在 X 正方向、X 负方向、Y 正方向或 Y 负方向上遇到故障区域。为了使消息绕过这 4 个方向上的故障区域保持路由的连通性，在 Torus 原有的虚拟通道 x_0，x_1，y_0 和 y_1 的基础上，再额外加入 3 个虚拟通道，新的 Torus 网络虚拟通道集为 $\{x_{0,0}, x_{0,1}, x_{0,2}, x_{1,0}, x_{1,1}, x_{1,2}, y_{0,0}, y_{0,1}, y_{0,2}, y_{1,0}, y_{1,1}, y_{1,2}\}$，其中虚拟通道 $x_{0,0}$，$x_{0,1}$，$x_{0,2}$ 是在原始虚拟通道 x_0 基础上增加的 0，1 和 2 三个虚拟通道，其它新增加的虚拟通道与此相类似。此外，每一个虚拟通道又可以分为正方向虚拟通道（用＋表示）和负方向虚拟通道（用－表示），例如 $x_{0,0}+$ 和 $x_{0,0}-$ 分别表示 $x_{0,0}$ 上的正方向通道和负方向通道。如图 2 所示，本文使用这些虚拟通道构成 4 个独立的虚拟网络 VN0，VN1，VN2 和 VN3。

如图 2(a)所示，虚拟网络 VN0 由虚拟通道 $x_{0,0}+$，$x_{1,0}+$，$y_{0,1}$ 和 $y_{1,1}$ 组成，用来使消息绕行 X 正方向的故障区域。虚拟网络 VN0 使用的路由函数如下所示，其中 $x+$. channel≠null 表示消息在向东路由的过程中未遇到故障，一个字位的“round”表示向东传输的消息由北边（＋）还是南边（－）绕行遇到的故障区域，y_{offs} 表示消息的目的节点和当前节点在 Y 维的偏移量，xr 表示消息的目的节点在 X 维上坐标大于源节点(0)或者小于源节点(1)，yr 表示消息的目的节点在 Y 维上坐标大于源节点(0)或者小于源节点(1)，chn 为所选通道。



(a) 虚拟网络 VN0

(b) 虚拟网络 VN1

(c) 虚拟网络 VN2

(d) 虚拟网络 VN3

图 2 Torus 中虚拟网络

```
Routing function of virtual network 0
Begin
  if x+. channel≠null then
    chn←x_{xr,0}+
    if y_offs>0 then around←"+" else around←"-"
  else
    if around="+" then chn←y_{yr,1}+ else chn←y_{yr,1}-
  endif
end
```

VN1 如图 2(b)所示，由虚拟通道 $x_{0,0}-$，$x_{1,0}-$，$y_{0,2}$ 和 $y_{1,2}$ 组成，用来使消息绕行 X 负方向的故障区域。虚拟网络 VN0 使用的路由函数如下所示，其中 $x-$. channel≠null 表示消息在向西路由的过程中未遇到故障区域。

```
Routing function of virtual network 1
Begin
  if x-. channel≠null then
    chn←x_{xr,0}-
    if y_offs>0 then around←"+" else around←"-"
  else
```

```
    if around="+" then chn←$y_{yr,2}$+ else chn←$y_{yr,2}$−
  endif
end
```

VN2 如图 2(c)所示，由虚拟通道 $y_{0,0}+$，$y_{1,0}+$，$x_{0,1}$ 和 $x_{1,1}$ 组成，用来使消息绕行 Y 正方向的故障区域，虚拟网络 VN2 使用的路由函数与 VN0 相似。VN3 如图 2(d)所示，由虚拟通道 $y_{0,0}-$，$y_{1,0}-$，$x_{0,2}$ 和 $x_{1,2}$ 组成，用来使消息绕行 Y 负方向的故障区域。虚拟网络 VN3 使用的路由函数与 VN1 相似。

Torus 容错路由算法通过调用这 4 个虚拟网络的路由子函数实现无死锁路由，其算法描述如下，其中 x_c、y_c 分别表示当前节点的 x 维、y 维坐标，x_d、y_d 分别表示目的节点的 x 维、y 维坐标，x_{offs}、y_{offs} 分别表示当前节点与目的节点在 x 维、y 维的偏移量。dim 是消息头控制域，初值为"X"，其值为"X"时表示消息在 X 方向上路由并绕行遇到的故障区域，其值为"Y"时表示消息在 Y 方向上路由并绕行遇到的故障区域。如果在初始状态 $y_{offs}>0$，则消息头控制域 around 初值为"+"，否则为"−"。

```
Algorithm R_rank-0(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
    chn←internal
  else
    if dim="X" and x_offs=0 then
      dim←"Y"
    endif
    if dim="X" then
      if x_offs>0 and |x_offs|≤radius_x
        then Routing function of vn_1
      if x_offs>0 and |x_offs|>radius_x
        then Routing function of vn_2
      if x_offs<0 and |x_offs|≤radius_x
        then Routing function of vn_2
      if x_offs<0 and |x_offs|>radius_x
        then Routing function of vn_1
    else // dim="Y"
      if y_offs>0 and |y_offs|≤radius_y
        then Routing function of vn_3
      if y_offs>0 and |y_offs|>radius_y
        then Routing function of vn_4
      if y_offs<0 and |y_offs|≤radius_y
        then Routing function of vn_4
      if y_offs<0 and |y_offs|>radius_y
        then Routing function of vn_3
    endif
end
```

Torus 容错路由算法使消息按照先 X 维、后 Y 维的顺序绕行故障区域，每一个消息将首先被定为 X 类消息，X 类的消息使用 VN0 或 VN1 的路由子函数绕行 X 方向上遇到的故障区域，直至消息的 X 维偏移量为 0。此时消息转变为 Y 类消息。然后消息使用 VN2 或 VN3 的路由子函数绕行 Y 方向上遇到的故障区域，直至抵达目的节点。

定理 1 Torus 容错路由算法是无死锁路由算法。

证明：

(1)4 个虚拟网络中的通道集是不相交的(显然成立)。

(2)算法在每一个虚拟网络中是无死锁的(显然成立)。

(3)算法允许消息转移到别的虚拟网络不会造成算法死锁。

算法允许消息转移到别的虚拟网络共有 4 种情况。(a) VN0 转到 VN2；(b)由 VN0 转到 VN3；(c) 由 VN1 转到 VN2；(d) 由 VN1 转到 VN3。

假设在情况(a)中算法的通道依赖图有环存在。那么

$$\exists c_1, c_2, \cdots, c_r \in vn_0 \begin{cases} c_i \rightarrow c_{i+1}, i=1,2,\cdots,r-1 \\ c_r \rightarrow c_1 \end{cases}$$

其中"→"表示通道依赖，由(2)可知 $c_1, c_2, \cdots, c_r$ 必然包括 2 个虚拟网络中的通道，因此 $c_1 \in$ VN0，$c_r \in$ VN2。由于算法不允许消息由 VN2 转到 VN0，因此假设不成立。根据 Duato 理论[5]可知，算法在情况(a)中是无死锁的。同理可以证明算法在情况(b)－(d)中也是无死锁的。

由(1)－(3)可知定理 1 成立。

4 仿真实验与结果分析

本文设计了一个微片级别的路由仿真器，其中设定 Torus 网络规模为 10×10，使用虫孔交换技术，每个消息平均 32 个微片，缓冲器大小为 128 个微片，对消息进行路由决策、消息穿过交换开关、消息在物理链路传输分别需要一个时钟周期。Torus 网络容错路由算法在不同故障率的 Torus 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 3 种情况下，路由算法的性能如图 3 所示。

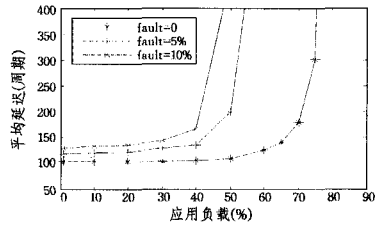


图 3 耐故障算法在不同故障域 Mesh 中的性能比较

由图 3 可以看到随着 Torus 网络中故障节点的增多，容错路由算法的传输延时缓慢地增加，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 30%的情况下，耐故障路由算法在 5%故障节点的 Torus 中的传输延时仅比在无故障节点的 Torus 中的传输延时高出 30%，因此本文提出的路由算法具有较强平滑降级使用功能。

结束语 容错问题是互连网络设计中的重要问题，提出了一种新的容错路由算法，并将其应用到 Torus 结构的虫孔交换网络。这一容错路由算法具有较低的路由限制和较高的自适应性，无论基于 Torus 网络中故障区域的数量多少或如何分布，算法总能保持路由的连通性和无死锁性。此外，这一算法仅使用了 3 个额外的虚拟通道，因此算法只需要较少的缓冲器资源以及相对较低的功耗。这一算法因为使用本地故障信息进行选路以绕行故障区域，所以可以快速进行路由决策，并且在网络中易于实现。最后，仿真实验结果证明，算法具有良好的平滑降级使用的功能，因此这一容错路由算法完全可以在 Torus 互连网络中得到应用。

(下转第 153 页)

新抛出异常 failEnrollServiceFault。

```
<!-- WS-Policy4BPEH language example -->
<wsp:Policy wsu:Id="BPEHPolicySample">
 <!-- failEnrollServiceFault exception definition-->
 <bpeh-dee:Exception ExceptionId="failEnrollServiceException">
    <bpeh-dee:ExceptionType name = "bpeh:failEnrollServiceFault"/>
    <bpeh-dee:ProcessId name = "GraduationApprovalProcess"/>
    <bpel-dee:ScopeId name = "InvokeEnrollApprovalActivity"/>
 </bpeh-cf:Exception>
 <!-- condition definition -->
 <bpeh-dee:Condition ConditionId="InvalidEnrollApprovalDuration">
    <bpeh-exp:Variable>$enrollApprovalTime</bpeh-dee:Variable>
    <bpeh-exp:TimeComparator>Between</bpeh-exp:TimeComparator>
    <bpeh-exp:TimeConstant>2011-01-01</bpeh-exp:TimeConstant>
    <bpeh-exp:TimeConstant>2011-12-30</bpeh-exp:TimeConstant>
 </bpeh-dee:Condition>
 <bpeh-dee:Condition ConditionId="retryExecutionPrecondition">
    <bpeh-exp:Variable>$currentScopeStatus</bpeh-exp:Variable>
    <bpeh-exp:booleanComparatorTypeChoice>==</bpeh-exp:booleanComparatorTypeChoice>
    <bpeh-exp:StringConstant>Faulty</bpeh-exp:StringConstant>
 </bpeh-dee:Condition>
 <bpeh-dee:Condition ConditionId="retryExecutionPostcondition">
    <bpeh-exp:Variable>$currentScopeStatus</bpeh-exp:Variable>
    <bpeh-exp:booleanComparatorTypeChoice>==</bpeh-exp:booleanComparatorTypeChoice>
    <bpeh-exp:StringConstant>Complete</bpeh-exp:StringConstant>
 </bpeh-dee:Condition>
 <!-- primary action definition -->
 <bpeh-act:Action ActionID="enrollApprovalRetry">
    <bpeh-act:Count>2</bpeh-act:Count>
    <bpeh-act:Interval>10</bpeh-act:Interval>
 </bpeh-act:Action>
 <!--rule definition -->
 <bpeh-rul:rule RuleId="RetryRuleOfEnrollApproval">
    <bpeh-rul:priority>1</bpeh-rul:priority>
    <bpeh-rul:condition>InvalidEnrollApprovalDuration</bpeh-rul:condition>
    <bpeh-rul:actionGroup>
       <bpeh-act:Action>enrollApprovalRetry</bpeh-act>
    </bpeh-rul:actionGroup>
    <bpeh-rul:preCondition>retryExecutionPrecondition</bpeh-rul:preCondition>
    <bpeh-rul:postCondition>retryExecutionPostcondition</bpeh-rul:postCondition>
 </bpeh-rul:rule>
 <!--return mode and rethrow definition-->
 <bpeh-rt:Return>Resume</bpeh-rt:Return>
 <bpeh-rt:Rethrows>
    <bpeh-dee:Exception>failEnrollServiceException</bpeh-dee:Exception>
 </bpeh-rt:Rethrows>
</wsp:Policy>
```

图 4 WS-Policy4BPEH 示例

为了实现异常处理策略的重用，采用 WS-AttachmentPolicy 将策略和策略作用域进行绑定，将前面定义的示例策略“BPEHPolicySample”绑定到毕业审核流程的学籍审核活动中。

```
〈wsp:PolicyAttachment xmlns:wsp="…"〉
〈wsp:AppliesTo xmlns:bpeh-dee="…"〉
    〈bpeh-dee:domain〉
        /GraduationApprovalProcess/InvokeEnrollApproval
    〈/bpeh-dee:domain〉
〈/wsp:AppliesTo〉
〈wsp:PolicyReference URI="http://www.sklse.whu.edu.cn/SOAEHPL/policies#BPEHPolicySample"〉
〈/wsp:PolicyAttachment〉
```

结束语 基于策略的面向服务流程异常处理建模方法，将异常处理逻辑从正常业务逻辑中分离出来，简化了异常处理逻辑的开发和维护。本文借鉴已有的研究成果，提出面向服务流程异常处理的策略描述语言 WS-Policy4BPEH，以支持面向服务流程异常及异常处理方式、异常返回方式和异常传播方式的表示。本文介绍了 WS-Policy4BPEH 语言的元模型、语法结构；结合现代远程教育中的“毕业审核流程”应用场景，给出了 WS-Policy4BPEH 语言的策略示例。

策略部署执行之前，分析和验证策略的正确性是十分必要的，而分析验证策略正确性需要策略语言具有良定义的形式化语义。因此，下一步的工作重点是定义WS-Policy4BPEH语言的形式化语义，以支持策略的分析和验证。

参 考 文 献

[1] Business Process Execution Language for Web Services version 2.0 [EB/OL]. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf

[2] Friedrich G, Fugini M, Mussi E, et al. Exception handling for repair in service-based processes[J]. IEEE Trans. Software Eng., 2010, 36(2), 198-215

[3] Liu A, Li Q, Huang L, et al. FACTS: A Framework for Fault Tolerant Composition of Transactional Web Services[J]. IEEE Trans. on Services Computing, 2010, 3(1)46-59

[4] Lerner B S, Christov S, Osterweil L J. Exception Handling Patterns for Process Modeling[J]. IEEE Trans. on Software Engineering, 2010, 36(2), 162-183

[5] Sloman M. Policy Driven Management for Distributed Systems [J]. Plenum Press Journal of Network and Systems Management, 1994, 2(4): 333-360

[6] Damianou N, Dulay N, Lupu E, et al. The ponder policy specification language [C] // The Policy Workshop. Bristol, U. K: Springer-Verlag, LNCS1995, Jan. 2001

[7] Sheng Q Z, Benatallah B, Maamar Z, et al. Configurable composition and adaptive provisioning of Web services [J]. IEEE Transactions on Services Computing, 2009, 2(1): 34-49

[8] Casati F, Ceri S, Paraboschi S, et al. Specificati-on and implementation of exceptions in workflow management systems[J]. ACM TODS, 1999, 24(3): 405-451

[9] W3C Web Services Policy Working Group. Web Services Policy (WS-Policy)1.5[EB/OL]. www.w3.org/TR/ws-policy/, Nov. 2006

[10] Anderson A H. An Introduction to the Web Services Policy Language(WSPL)[C]//IEEE, 2004: 189-192

[11] Pautasso G A C, Heinis T. Autonomic Execution of Service Compositions[M]. Proc. of ICWS. 2005

[12] Tosic V, Erradi A, Maheshwari P. WS-Policy4MASC-A WS-Policy extension used in the MASC middleware[C]//SCC'07. 2007: 458-465

[13] Baresi L, Guinea S. Self-supervising BPEL Processes[J]. IEEE Transactions on Software Engineering, 2010: 99(PrePrints)

[14] Hamadi R, Benatallah B, Medjahed B. Self-adapting Recovery Nets for Policy-driven Exception Handling in Business Processes [J]. Distributed and Parallel Databases, 2008, 23(1): 1-44

[15] Shankar C S, Ranganathan A, Campbell R. An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environment[C]//Mobiquitous 2005: The Second Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services. San Diego, California, July 2005

(上接第 117 页)

参 考 文 献

[1] Adiga N R, et al. Blue Gene/L Torus Interconnection Network [J]. IBM J. Research and Development, 2005, 49: 265-276

[2] 段新明. Mesh 网络耐故障虫孔路由[J]. 计算机科学, 2007, 34(11): 29-31

[3] Chalasani S, Boppana R V. Fault-tolerant wormhole routing in tori[C]//Proceedings of the 8th International Conference on Supercomputing. July 1994: 146-155

[4] Glass C J, Ni L M. The turn model for adaptive routing[C]// Proceedings of the 19th International Symposium on Computer Architecture. May 1992: 278-287

[5] Duato J. A new theory of deadlock-free adaptive routing in wormhole networks[J]. IEEE Trans. Parallel and Distributed Systems, 1995, 4(12): 1320-1331

[6] Jiang Z, Wu J, Wang D. A New Fault Information Model for Fault-Tolerant Adaptive and Minimal Routing in 3-D Meshes [J]. IEEE Trans. Reliability, 2008, 57(1): 149-162