

文章编号:1001-9081(2006)09-2162-04

基三分层网络中一种基于查表的确定路由算法

乔保军^{1,2}, 石峰¹, 计卫星¹, 刘滨¹

(1. 北京理工大学 计算机科学与工程系, 北京 100081;

2. 河南大学 数据与知识工程研究所, 河南 开封 475001)

(qbj@henu.edu.cn)

摘要:路由算法对互连网络的通信性能和并行系统性能的发挥起着重要作用。针对基三分层互连网络,提出一种基于查表的使消息沿两节点间近似最短路径传递的分布式确定路由算法 TDRA。该算法充分利用基三分层互连网络的层次特性,其路由表中仅保存各节点的同族节点和部分组的路由信息,路由表所占存储空间小,路由效率高。

关键词:并行计算; 互连网络; 分布式确定路由算法

中图分类号: TP393.03 **文献标识码:** A

Table-lookup determined routing algorithm for triple-based hierarchical interconnection network

QIAO Bao-jun^{1,2}, SHI Feng¹, JI Wei-xing¹, LIU Bin¹

(1. Department of Computer Science and Engineering, Beijing Institute of Technology, Beijing 100081, China;

2. Institute of Data and Knowledge Engineering, Henan University, Kaifeng Henan 475001, China)

Abstract: Efficient routing algorithm is very essential to the performance of the interconnection network and the whole parallel computing system. This paper presented TDRA (Table-lookup Deterministic Routing Algorithm) in triple-based hierarchical interconnection network, which always delivers the message along the approximately shortest path between any two nodes. Because the route table of any nodes only stores the route information for its cognation nodes and some special group, the cost of storage is very low and the route performance is high. Finally, the analysis based on the simulation of TDRA shows it is not only very simple and easy to implement, but also highly effective.

Key words: parallel computing; interconnection network; distributed deterministic routing algorithm

0 引言

由于直接互连网络(Direct Interconnection Network)具有较好的扩展性,随着系统中节点数量的增加,整个系统的通信能力和处理能力也随之增加,直接互连网络已成为构建大规模并行计算系统的主流体系结构^[1]。目前许多实验性和商用多计算机或多处理器系统(MIT Alewife Machine^[2]、Intel Paragon^[3])基本上都使用这种低延迟高带宽的直接互连网络方式。

基三分层互连网络(Triple-based Hierarchical Interconnection Network, THIN)是一种新的直接互连网络,网络拓扑简单并具有明显的层次性。该网络中最低层节点之间采用全互连结构,较高层次网络之间的连接链路相对较少,适合于通信局部性^[4]较高的嵌入式并行应用领域。互连网络的通信性能在很大程度上决定了这些并行计算系统的性能,而路由算法对网络通信效率的提高起着重要作用。

本文针对 THIN 提出一种基于查表的确定路由算法 TDRA。TDRA 算法充分利用网络的层次特性,在每个节点的路由表中只保存该节点的同族节点和部分组(层)的路由信息,使消息沿着两点间的一条近似最短路径进行传递。

TDRA 算法实现简单,路由表所占存储空间小,查找效率高。

1 基三分层互连网络概述

本节主要从网络构造和节点编码方案两个方面介绍 THIN。

1.1 网络构造

THIN 是一种层次化的、可扩展的互连结构。该结构的第 0 层是单个节点,通过 3 条通信链路将 3 个节点彼此互连形成一个三角形,从而构成该结构的第 1 层。第 1 层是构造 THIN 的基本构件,如图 1 所示。

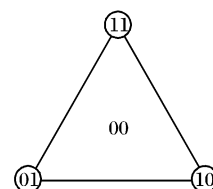


图 1 一层基三分层互连网络

利用这个基本构件,通过使用迭代函数族(IFS)可以构造一个任意层次的 THIN。假设该结构的迭代函数族为 $IFS\{F_1, F_2, F_3\}$,将 1 层 THIN 看作是 1 次迭代后的网络

收稿日期:2006-03-27; 修订日期:2006-06-27

作者简介:乔保军(1975-),男,河南焦作人,讲师,博士研究生,主要研究方向:ASIC 设计方法、嵌入式计算; 石峰(1961-),男,吉林松源人,教授,博士生导师,主要研究方向:EDA 设计方法、嵌入式计算; 计卫星(1980-),男,陕西西安人,博士研究生,主要研究方向:嵌入式计算; 刘滨(1975-),男,河北石家庄人,博士研究生,主要研究方向:嵌入式计算。

$N(1), N(k)$ 表示 k 次迭代后得到的 k 层网络,那么 THIN 的构造过程就可以表述为:

$$N_{k+1} = \bigcup_{i=1}^3 F_i(N_k) \quad (1)$$

即在基本构件的基础上,将每个节点用一个低层网络替代,从而得到更高层的一个三角形网络结构。重复这一过程,我们可以构造任意层次的 THIN,其构造过程如图 2 所示。

1.2 节点编码规则

为了能快速准确地定位网络中的节点,THIN 采用一种充分利用网络层次特性的节点编码。

定义 1 THIN 中节点的基本编码集 $BC = \{01, 10, 11\}$ 。

THIN 的构造是一个迭代的过程,因此该网络节点的编码也是一个迭代过程。节点编码规则如下:

(1) $k = 1$ 时,网络 $N(k)$ 中有三个节点,每个节点用两位编号表示,编码形式为 $b_1b_0 (b_1b_0 \in BC)$,并从 01 到 11 按逆时针排列,组编码 00 则表示该网络中的所有节点。如图 1 所示。

(2) 假设 $N(k-1)$ 的编码已经完成, $N(k)$ 可由三个 $N(k-1)$ 子网构成。 $N(k)$ 中每个节点编码由 $N(k-1)$ 中节点的编码前面拼接上该 $N(k-1)$ 所替代的 $N(1)$ 中对应节点的编码而得,编码长度为 $2k$ 位,如图 2 所示。

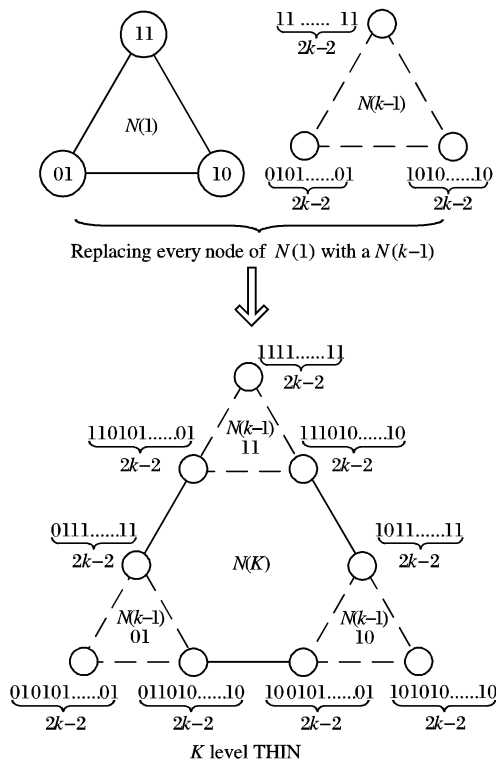


图 2 K 层基三分层互连网络构造过程

(3) 对于任意形式为 $b_{2k-1}b_{2k-2} \cdots b_{2i-1}b_{2i-2} \cdots b_1b_0$ 的编码,若存在 $i (1 \leq i \leq k)$ 使得 $b_{2i-1}b_{2i-2} = 00$,则该编码是一个组编码;否则为一般的节点编码。

这种分层节点编码方案结构清晰,不仅可以标识网络中的单个节点,还可以对节点进行分组,便于网络组播。图 3 给出了含有 27 个节点的三层 THIN 的图示。为了便于描述,本文用 $n(i)$ 代表地址编码为 i 的节点,用 $C_{i,j}$ 来表示连接节点 $n(i)$ 和 $n(j)$ 之间的通信链路。

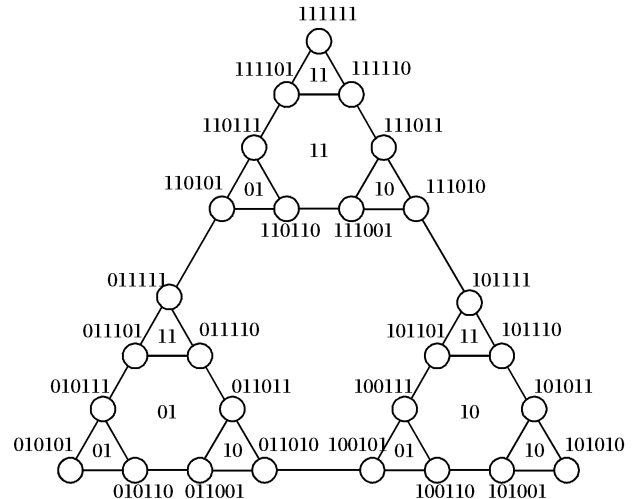


图 3 3 层基三分层互连网络

2 TDRA 算法设计

本文在权衡性能、成本和易实现性的基础上,充分利用 THIN 自身的层次性,提出一种基于查表的确定路由算法 TDRA (Table-lookup Determined Routing Algorithm)。TDRA 算法要求节点接收到消息后,由节点自身决定是接收该消息或是查找节点路由表确定该消息的输出通路以续传给其他相邻节点。TDRA 在确定路由时,不需要整个网络的状态信息,从而避免了为每个节点提供全局路由信息所额外生成的通信开销和节点存储开销。

为了加快路由表查找速度,简化算法实现,本文参考文献 [5] 的思想,将节点用于寻径的信息分成两个表存放,一个是输出通道状态表 CST (Channel Status Table),保存了节点上所有输出通道的连接工作状态;另一个是 TDRA 路由表 RT (Route Table),保存了从当前节点去往其他目的节点所应选择的输出通道信息。

2.1 输出通道状态表的设计

输出通道状态表的结构如表 1 所示。

表 1 输出通道状态表

输出通道	通道忙状态位(1 位)	通道错误状态位(1 位)	最近输出消息标识
------	-------------	--------------	----------

输出通道项用来标识 THIN 中一个节点的所有输出通道,本文用一个节点到其相邻节点的通信链路来表示该节点对应的输出通道。因为在 THIN 中,节点度的最大值等于 3,所以在每个节点的输出通道状态表中,最多存放 3 个输出通道的信息。

通道忙状态位用来表明对应的输出通道的工作状态,如果该输出通道被占用,忙状态将被置位。

通道错误状态位用来标识输出通道是否出现故障。如果某个输出通道未被占用,却又不响应占用请求时,就认为该输出通道出现故障,便将其通道错误状态置位,避免以后其他消息路由时,在该输出通道的申请上浪费时间。

最近输出消息标识用来记录对应输出通道当前或最近输出的消息标识,我们用消息的源节点和目的节点地址作为消息的标识。节点接收到消息后,如果需要续传该消息,节点将

用消息的标识作为关键字在 CST 中检索,判断是否有输出通道最近传送过相同标识的消息。如果检索成功,对应的输出通道将作为该消息续传的传输通路,如果检索不成功,则在节点的 TDRA 路由表中查找以确定消息的传输通路。

2.2 TDRA 路由表的构建

TDRA 路由表的结构如表 2 所示。

表 2 TDRA 路由表结构

目的地址	输出通道
------	------

输出通道项含义和 CST 中的输出通道项的含义相同。

目的地址项存放了从当前节点出发的可达地址。TDRA 路由表充分利用了 THIN 的层次性,在节点路由表中并不存放到所有其他节点的路由信息。这种设计方案使得节点路由表的存储开销小,特别适合嵌入式系统对硬件资源的要求。为了清楚说明路由表中的目的地址的可能取值情况,我们给出下列定义:

定义 2 对于 $N(K)$ 中的任意两个节点 $n(i)$ 和 $n(j)$,如果这两个节点的编码除了最低两位不同,其余 $2k-2$ 位都相同,那么我们称这两个节点为同族节点。

定义 3 $N(K)$ 中任一节点 $n(b_{2k-1}b_{2k-2}\cdots b_1b_0)$ 的 TDRA 路由表中的目的地址集为 DD , 其中:

$$DD = \{b_{2k-1}b_{2k-2}\cdots b_{2i+1}b_{2i}b'_{2i-1}b'_{2i-2}\underbrace{00\cdots 00}_{2^{i-2}\text{个}} \mid 1 \leq i \leq kb'_{2i-1}b'_{2i-2} \in BCAB'_{2i-1}b'_{2i-2} \neq b_{2i-1}b_{2i-2}\}$$

定义 3 说明了任意节点的 TDRA 路由表中的目的地址包括有该节点的同族节点的编码和一些特定组的编码(不包含该节点的不同层的编码)。图 3 中节点 $n(111010)$ 的 TDRA 路由表如表 3 所示。

表 3 $n(111010)$ 的 TDRA 路由表

目的地址	输出端口
111011	$C_{111010,111011}$
111001	$C_{111010,111001}$
111100	$C_{111010,111011}$
110100	$C_{111010,111001}$
010000	$C_{111010,111001}$
100000	$C_{111010,101111}$

2.3 TDRA 算法设计

下面分别给出了 TDRA 路由算法和 TDRA 路由表查找算法的具体描述。

TDRA 算法描述

TDRA()

```
{ if 有消息到达节点 then
{ 从消息头中抽出源和目的节点地址: S_address, D_address
if (D_address = 当前节点地址 C_address) then
{ receive_message(); /* 该节点接收此消息 */
return; }
else /* 该消息还未到达目的节点,需为此消息寻径 */
{ 用当前消息的标识作为关键字,在输出通道状态表(CST)
中检索;
/* 检索成功,表明 CST 中存在相同标识消息的路由信息 */
if succeed then
{ /* 输出通道忙,阻塞消息的传递,等待通道空闲 */
if (该通道忙状态位 = 1) then
block_message() UNTIL (忙状态位 = 0);
将通道忙状态置位;
send_message();
```

```
return;
}
else
/* CST 中没有该消息的寻径信息,调用函数 lookup_RT 查找
TDRA 路由表 RT,以确定该消息对应的输出通路 */
{ output = lookup_RT(D_address);
/* 输出通道忙,阻塞消息的传递,等待通道空闲 */
if (output 的通道忙状态位 = 1) then
block_message() UNTIL (忙状态位 = 0);
将通道忙状态置位;
send_message();
更新 CST 中对应端口的最近输出消息标识项;
Return;
}
}
```

lookup_RT 的算法描述

函数名: lookup_RT

输入: 消息的目的地址(目的节点的编码)

输出: 消息将被转发到的输出通道

lookup_RT(D_address)

```
{ /* 在 K 层基三分层互连网络中,节点编码长度是 2K 位 */
Mask_code = 11...11 (2k 位); /* 设置运算掩码的初始值 */
/* 将目的节点编码和运算掩码按位进行与运算 */
Index_code = D_address & Mask_code;
While (Mask_code ≠ 00...00 (2k 位))
{ 以 Index_code 作为关键字在 RT 中检索;
if succeed then
return 对应的输出端口;
else
{ Mask_code 左移 2 位;
Index_code = D_address & Mask_code;
}
}
```

从 TDRA 路由表的构建以及 TDRA 路由表查找算法可知,一个节点运行 TDRA 算法为消息进行寻径的过程为:如果消息的目的节点和当前节点属于同族节点,则选择连接两节点的数据通道将消息传递给目的节点;否则将该消息传递给目的节点所属且不包含当前节点的最高层上。由于这种分层寻径过程,会导致 TDRA 算法为某些节点间的消息传递所确定的路径不是两点间的最短路径。如图 3 所示,节点 n_{111010} 到 n_{011010} 的最短路径是 $< C_{111010,101111}, C_{101111,101101}, C_{101101,100111}, C_{100111,100101}, C_{100101,011010} >$, 路径长度为 5。而按照 TDRA 为消息寻径时,消息将沿着 $< C_{111010,111001}, C_{111001,110110}, C_{110110,110101}, C_{110101,011111}, C_{011111,011110}, C_{011110,011011}, C_{011011,011010} >$ 进行传递,路径长度等于 7。我们把由 TDRA 算法确定的两点之间的路径称为近似最短路径。

3 仿真与分析

TDRA 路由表中并不存放节点到所有其他节点的输出通道信息,而是仅存放了去往同族节点和一些层的路径信息。TDRA 路由表中路由条目随着网络层数的增加呈线性增长,使得路由表的规模不会随着网络规模的增大而迅速膨胀。这种路由表构建方案的扩展性强,存储开销小,路由表查找效率高,比较适合构造规模较大且对系统存储开销要求高的直接互连网络。图 4 给出了节点的 TDRA 路由表(TDRA_Router)和存放去往所有节点路由信息的全路由表(All_Router)在不

同层 THIN 中,路由表中路由条目的取值情况。

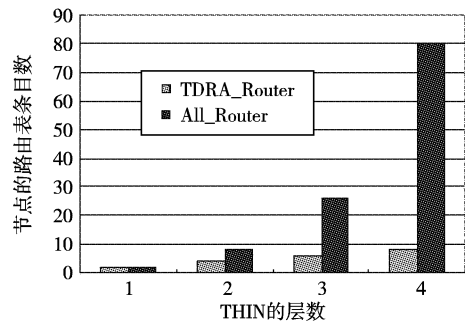


图4 TDRA_Router 和 All_Router 路由表条目数比较

尽管因为 TDRA 路由表中仅存放了部分路由信息,导致为某些节点间的消息选择的路径不是最短路径,但通过模拟仿真比较网络的平均最短距离和平均近似最短距离,我们发现在整个网络的消息传输过程中这个近似最短路径可以被看作是最短路径。

定义 4 网络的平均最短距离等于网络中任意两点之间的最短路径长度之和与路径总数之比。

定义 5 网络平均近似最短距离等于网络中任意两点之间按照 TDRA 求得的近似最短路径长度之和与路径总数之比。

表 4 平均最短距离和平均近似最短距离的取值比较

层数	平均近似最短距离	平均最短距离
1	0.667	0.667
2	1.778	1.778
3	3.926	3.893
4	8.197	8.102
5	16.733	16.523
6	33.800	33.370
7	67.930	67.068

模拟仿真中采用了文献[6]给出的消息排队模型,并假设网络中每条链路都是全双工通信方式,使用两个单工链路

取代每条全双工链路,并将每条单工链路看作为一个消息处理中心,消息处理中心根据消息到达的顺序对消息进行处理。

表 4 给出了在不同层的 THIN 中,平均最短距离和平均近似最短距离的取值情况。从表中我们可以看出,两者相差很小,在某种程度上我们可以把平均近似最短距离看作平均最短距离。

TDRA 路由算法充分利用 THIN 的层次特性,在路由表中仅存放部分目的地址的路由信息,使路由表存储开销小,路由算法设计简单,查找速度快,可以保证算法的高效性,从而实现高带宽、低延迟的互连网络。

4 结语

针对基三分层互连网络,本文提出的 TDRA 算法,是一个使消息沿着近似最短路径进行传递的分布式确定路由算法。该算法充分有效地利用网络自身的层次性,提高网络传输性能,而且算法设计简单易于实现。我们将对 TDRA 路由算法作进一步的仿真研究,了解该算法在不同网络负载分布情况下的性能表现,并进一步和其他路由算法进行性能比较。

参考文献:

[1] NI LM, MCKINLEY PK. A survey of wormhole routing techniques in direct networks[J]. Computer, 1993, 26(2): 62 - 76.

[2] AGARWAL A, BIANCHINI R, CHAIKEN D, *et al.* The MIT Alewife machine[A]. Proceedings of the IEEE[C], 1999, 87(3): 430 - 44.

[3] Intel corporation. Paragon XP/s Product Overview[R]. Beaverton, OR, Supercomputer Systems Division, 1991.

[4] TIRUVEEDHULA V, BEDI JS. Performance analysis of a hierarchical interconnection network using hypercubes[A]. Proceedings of the 35th Midwest Symposium[C], 1992. 867 - 870.

[5] 杜毅,李三立. k-ary n-cube 网络中高速开关 TH-Switch 的设计与路由算法[J]. 计算机学报, 1999, 22(1): 16 - 23.

[6] DANDAMUDI SP, EAGER DL. Hierarchical interconnection networks for multicomputer systems[J]. IEEE Transactions on computers, 1990, 39(6): 786 - 797.

(上接第 2161 页)

是将一个地址转换为另外一个 IP 地址的技术。它主要有三个功能:一是隐藏内部网络真正的 IP,使“黑客”无法直接攻击内部网络;二是让内部网络使用保留地址,使得节约有限的 IP 地址;三是实现负载均衡,提升网络性能。

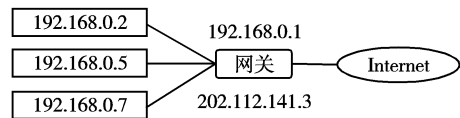


图3 NAT 拓扑结构

这是将包过滤技术应用于网关上的一种方案。虽然 NAT 的实现通常也是由路由器来实现的,但传统的基于硬件的路由器价格往往较高,而且在应用过程中,常遇到的问题都是将两个或两个以上的局域网互联,也不一定必须使用昂贵的专用路由器。并且通过研究软件实现方法也为后续更高级的应用如 VPN,数据加密打下基础。

由上述分析可知,我们可以得到所有经过网关的数据包,并获得所有数据包的源、目的 IP 地址。那么接下来需要做的就是替换所有数据包中的源 IP 地址,换为网关的地址,重构好数据包后继续按照目的地址发送出去。采用这种软件实现 NAT 的方式有更多的灵活性,如可以控制共享一个 IP 地址发

出流向外部网络的数据流,而不接受外部网络向内部网络发出的连接请求,比如远程的工作站等。而双向的静态 NAT 常被安装有防火墙的大型网络使用。另外,当 ISP 的地址发生变化时,局域网内部不需要改变原有的地址规划,只需在 NAT 上修改目标地址即可。

4 结语

无论是网络监控还是软路由,其实都是包过滤技术的具体应用,所不同的是对所捕获的数据包进行如何的处理。应该综合各种技术应用,在主动性方面多加研究,从而更加全面的提高网络的安全性。

参考文献:

[1] BAKER A, LOZANO J. Windows 2000 设备驱动程序设计指南[M]. 北京:机械工业出版社,2001.

[2] MICOSOFT. Microsoft Windows XP Driver Development Kit[M]. America: Microsoft Corportion, 2001.

[3] 李晓莺,曾启铭. NDIS 网络驱动程序的研究与实现[J]. 计算机应用, 2002, 22(4): 60 - 61.

[4] 郭兴阳,高峰,唐朝京. 一种 NDIS 中间层数据包过滤方法[J]. 计算机工程, 2004, 30(17): 102 - 103.