正方形 Mesh 中的 Gossiping 算法

The Gossiping Algorithm in Square Mesh

王倬彦　应　晶

（浙江大学计算机系　杭州310027）

Abstract　We offer a fast gossiping algorithm in Square mesh in this paper. We adopt H* model and assume that a packet can only travel across one edge in each timestep.

Keywords　Gossiping, Mesh, Timestep

1　介绍

在并行计算和分布式计算中，处理器之间的通信是一个重要课题。在一个计算机网络中，如果网络中的每一个处理器都要把一个消息传给其他所有处理器，这样的通信问题我们称之为 Gossiping。由于 Gossiping 问题具有大通信量的特点，它通常可以被用来衡量一个互联网络的通信能力。同时，Gossiping 也经常作为一些实际计算的嵌入式操作来使用。例如在矩阵的乘法和 Load Balancing 中都使用到了 Gossiping 算法。

Krumme 等提出四种研究 Gossiping 问题的模型：H*（half-duplex, all-port），H1（half-duplex, one-port），F*（full-duplex, all-port），F1（full-duplex, one-port）。在本文中，我们采用 H* 模型。在 H* 模型中，任意两个处理器之间的链路都是单向连通的，即在任一时刻，链路只能向一个方向传输数据。同时，在某一时刻，处理器可以同时接收任意条链路传来的消息。另外，我们还要假设在链路中传递的数据包（packet）只能包含一个消息的大小，即每次链路中只能传递一个消息。

整个 Gossiping 过程由时间片组成，我们规定数据包在每个时间片内只能传过一条边（相邻的两个处理器之间的链路）。这样我们就可以用所用时间片的多少来评价算法的效率。在3-D square mesh M 中，某个 Gossiping 算法 G 所需花费的时间片记为 G(M)。

2　预备知识

一个 square mesh 有 n×n 个结点。每个结点都可以用坐标(i,j)唯一标识。第一个结点的坐标为(0,0)，最后一个结点的坐标为(n−1,n−1)。每个结点代表一个处理器，而两个结点之间的边就代表处理器之间的链路。

下面我们给出一个简单的定理来指出在 square mesh 中，Gossiping 算法所使用的时间片的可能的最小值。

定理1　在一个 n×n 的 square mesh M 中，G(M)的最小值为$(n^2+n)/2$。

证明：在 M 中，共有 n^2 个结点，每个结点要将一个消息发送到其他 n^2-1 个结点，所以所有消息至少要经过 $n^2(n^2-1)$ 条边。而 M 总共有 2n(n−1)条边。所以至少有一条边要被使用 $n^2(n^2-1)/2n(n-1)=(n^2+n)/2$ 次。

3　正方形 mesh 中的 gossiping 算法

我们给出的在二维平面 square mesh 中的 gossiping 算法如下：首先，将所有结点分为偶结点和奇结点。结点(x,y)是偶结点如果 x+y 是偶数。如果 x+y 是奇数，该结点就是奇结点。

第一阶段：所有偶结点在行的方向上广播自己的消息，同时所有的奇结点在列的方向上广播自己的消息。

第二阶段：每行和每列将第一阶段中得到的消息做线性 gossiping。

第一阶段分成三种情况，当每行和每列的结点数 n 为奇数时，有两种情况，当每行和每列的结点数 n 为偶数时，有一种情况。不失一般性，我们拿出 square mesh 中的一行来研究。第一阶段的三种情况如图1所示：我们0，1，…来代表一行中的结点，用实心圆点代表在某一时间片中，相邻两个结点之间的链路正在传送消息，用空心圆点代表相邻两个结点之间的链路空闲着，黑色连线表示同一个消息不断传递的路径。n 为奇数并且行的第一个结点为奇结点的执行情况如(a)中所示。n 为奇数并且行的第一个结点为偶结点的执行情况如(b)中所示。n 为偶数的执行情况如(c)中所示。由图中可以得出，第一阶段所需要的时间片为 n−1。

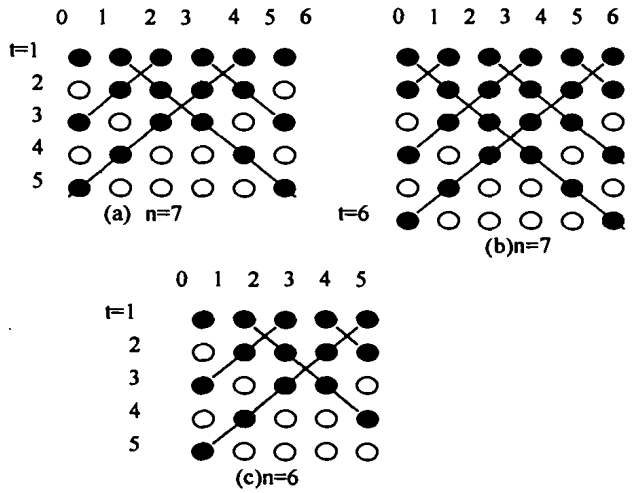

图1　第一阶段的具体执行情况

第二阶段线性 gossiping 算法如图2所示。不失一般性，我们拿出 mesh 中的一行来研究。为了方便表示，我们使用 v_0，v_1，…，v_{n-1} 来表示一行中的 n 个结点，用 v_c 代表中央结点，c=(n−1)/2(n=奇数)，c=n/2(n=偶数)。每个结点有两个队列变量，L_i 和 R_i，这两个变量分别用来储存 v_i 向左边和右边传

王倬彦　硕士生，主要研究方向：计算机软件、算法。应　晶　教授。

送的消息。一开始，他们被初始化为 v_i 在第一阶段中从列的方向上得到的消息（如果是在列上做线形 gossiping，则将他们初始化为 vi 在第一阶段中从行的方向上得到的消息）。我们只关注左边半行的执行，右边半行是相似的。算法在左半部分的执行可以看作由三个子操作完成，一个子操作是 $v_0, v_1, \cdots, v_{c-1}$ 的消息传送到 v_c（途经所有的中间结点），第二个子操作是 $v_{c-1}, \cdots, v_1$ 的消息传送到 v_0（途经所有的中间结点），第三个子操作是 v_c 积聚的 $v_c, v_{c+1}, \cdots, v_{n-1}$ 传送到 $v_0, v_1, \cdots, v_{c-1}$。

对于 n 为奇数的情况，由于在每一列中奇结点和偶结点的个数不同，导致第一阶段结束后行上的结点从列的方向上得到的消息个数是不同的：或者偶结点得到的消息数比奇结点得到的消息数多一个，或者相反。我们来研究前一种情况，（后者情况基本相似），偶结点有 $(n+1)/2$ 个消息，奇结点有 $(n-1)/2$ 个消息。

```
◇Ri=Li={vi 在第一阶段中从列的方向上得到的消息}0≤i≤n-1;
◇node vi,i<c,位于中央结点左边的结点
  Repeat(以下的两个 if 语句并行执行)
  ·If Ri≠ϕ
        将 Ri 中的第一个消息发送到 vi+1并且将这个消息从 Ri 中删除
   else
        从 vi+1(Li+1)得到一个消息并将这个消息加入 Li(if i>0);
  ·if vi-1传来一个消息
   将此消息加入 Ri;
  until done;
◇node vi,i>c,位于中央结点右边的结点
  repeat
  (与位于中央结点左边的结点的算法相似，只是所有方向相反)
  until done;
◇中央结点 Vc
  (以下两个 repeat 语句并行执行)
  ·repeat
        从左边获取一个消息并将这个消息加入 Rc;
   until 没有可以获得的消息(Rc-1=ϕ)
   ·repeat
        从右边获取一个消息并将这个消息加入 Lc;
   until 没有可以获得的消息(Lc+1=ϕ)
```

图2　第二阶段线形 Gossiping 算法

图3显示了在 n=7时第2阶段的执行情况，我们只列出了左半行的执行情况，因为左右是完全对称的。在该图中，大的长方形含有 $(n+1)/2$ 个消息，小的长方形含有 $(n-1)/2$ 个消息。从图中可以计算出第2阶段所用的时间片为：

$$(n+1)^2/2+(n-1)^2/2+(n-1)/2-1=(n^2+n-2)/2=S_{odd}$$

上面式子中的第一项对应图中的大长方形，第二项对应图中的小长方形，第三项对应空心圆点。

用同样的方法，我们可以计算出 n 为奇数时另一种情况下（偶结点在第一阶段得到的消息比奇结点得到的少一个）第2阶段所需的时间片：$(n^2+n-4)/2=S'_{odd}$

明显的，$S_{odd}>S'_{odd}$，所以第二阶段线形算法在所有行和列上执行所需的时间片为 S_{odd}。

在 n 为偶数的情况下，第一阶段结束后，偶结点和奇结点得到的消息数是一样的，都为 n/2个。因此，我们可以很容易得到 n 为偶数时第二阶段的执行时间为：

$$n*n/2+n/2-1=(n^2+n-2)/2=S_{even}=S_{odd}。$$

综合以上两个阶段，该算法总共所需的时间片为 $S_m=S_{第一阶段}+S_{第二阶段}=(n^2+3n-4)/2$。.

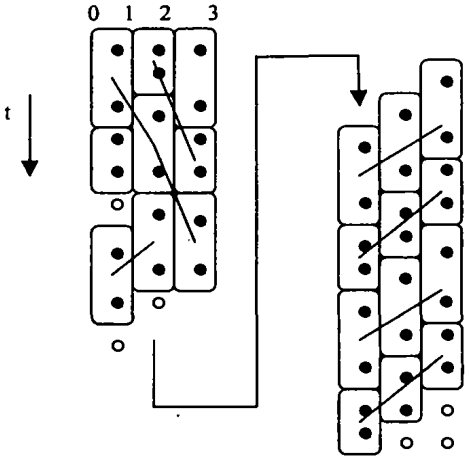

图3　n=7时第2阶段执行情况（只显示左半边）

总结　在本文中，我们给出了 square mesh 中的 gossiping 算法，时间复杂度为 $1/2O(n^2)$，与我证明的时间复杂度是一样的。但是这个算法还不是最优的，主要因为第一和第二阶段的耦合不够紧密。

参考文献

1　Bermond J-C, Gargano L, Rescigno A A, Vaccaro U. Fast gossiping by short messages. SIAM J. on computing, 1998, 27: 917～941

2　Krumme D W, Fast Gossiping for the Hypercube, SIAM J. Computing, 1992, 21(2): 365～380

（上接第180页）

提出的异步接口模式抽象了异步接口的一般性实现方法，在独立于特定操作系统的前提下能构造出同时支持同步和异步接口的客户端软件。在异步接口模式中，异步请求通过队列委托给独立的收发线程执行，对于返回结果，客户端可采用回调、等待或查询等方式处理。异步接口模式可用于构造灵活高效的异步编程接口。目前，异步接口模式已经在消息队列中间件的实现中得到了成功的应用。

参考文献

1　Coulouris G, Dollimore J, Kindberg T. Distributed Systems: Concepts and Design. Addison-Wesley, 2000

2　Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, 1995

3　Schmidt D C, Stal M, Rohnert H, Buschmann F. Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects. John Wiley & Sons, 2000

4　中国科学院软件研究所软件工程技术中心报告. 消息队列中间件 ISMQ 的设计和实现. 2000

5　Lavender R G, Schmidt D C. Active Object: an Object Behavioral Pattern for Concurrent Programming. in Pattern Languages of Program Design (J. O. Coplien, J. Vlissides, and N. Kerth, eds.), Addison-Wesley, 1996

6　Schmidt D, Cranor C. Half-Sync/Half-Async -- An Architectural Pattern for Efficient and Well-structured Concurrent I/O. In: Proc. of the 2nd Pattern Languages of Programs conf. 1995

7　Schmidt D C, et al. Leader/Followers: A Design Pattern for Efficient Multi-threaded Event Demultiplexing and Dispatching. In: proc. of the 7th Pattern Languages of Programs Conf. Aug. 2000