一种基于 Ethernet 新的可靠多播算法*

申 俊 郑纬民 王鼎兴 沈美明

(清华大学计算机科学技术系 北京 100084)

摘要 可靠多播算法在并行处理中有着广泛的应用.文章在分析基于共享介质 Ethernet 的3种可靠多播算法的基础上,提出了一种新的可靠多播算法:MAK(multicast acknowledgment)多播算法,并且以实验数据验证了此算法的高效性.

关键词 并行处理,通信,可靠性,算法,延迟.

中图法分类号 TP393

在网络通信中,点对点通信是最基本的通信形式,它满足了应用程序的基本要求,但在并行处理中还广泛使用到一种操作——collective 操作[1],如广播、全局 reduction、barrier 同步等.而支持 collective 操作最基本的通信功能是多播(Multicast).所谓 multicast 就是把一条消息传送到多个指定的目的结点.multicast 功能的高效实现对并行计算的很多方面都非常有益,如数据的分发、全局事件的通知、分布共享存储器的更新等操作就需要 multicast 的支持.因此,multicast 的实现对并行系统的整体性能有较大程度的影响.

multicast 通信对网络并行计算很有用处,现在很多并行程序设计语言和环境都提供了对 multicast 操作的支持,如 HPF(ligh performance fortran)、PVM(parallel virtual machine)在它们的运行系统中都实现了 multicast 功能.但是,由于这些系统大多是在上层实现 multicast 功能,不能充分利用网络硬件的特点,而且上层通信的开销往往比底层大,使得 multicast 性能不太理想.因此,如能把 multicast 的功能放在底层通信协议中直接实现,则可较大程度地提高 multicast 的性能,又能减小上层软件的开销.不过要在通信协议层实现 multicast,其难度比上层实现要大,特别是要实现可靠的 multicast 操作.这主要是由于 multicast 通信是一种全局操作,涉及多个结点的协作,协议的处理过程显然比点对点通信复杂,而且目前很多网络在硬件上不支持 multicast 通信,即使支持,也只是提供不可靠的 multicast 通信,因此如何在通信协议层实现可靠 multicast 成为对机群系统通信机制进行研究的一个重要目标.[3]

一般来说,设计 multicast 算法,主要考虑3个问题:①如何确定各结点之间的连接关系,使得 multicast 通信时网络冲突最小;②如何设计消息的传送路径[3,4],使得通信延迟尽可能小;③在实现可靠 multicast 时,如何使应答消息 ACK(acknowledgment)尽可能少,而且响应速度尽可能快.不过针对不同的网络硬件就有不同的考虑,本文着重讨论如何在 Ethernet 上实现可靠 multicast.下面先介绍一下 Ethernet 的硬件结构.

1 Ethernet 硬件结构

Ethernet 是一种共享总线类型的网络.它采用 CSMA/CD(carrier sense multiple access with collision detection)技术进行网络通信,其地址定义符合 IEEE 802.2 的 LAN 寻址标准.[5]在这个标准中规定,目的地址的最高位如果为1,则此地址是 multicast 地址;如果地址所有的位全为1,则这个地址是广播地址;如果地址最高位为0,说明是普通 unicast(点对点的通信方式)地址,只有与本地地址相符合的消息才能被该结点接收.

如图1所示,当发送方把数据包送到网络上时,网上所有的结点都能查看此数据包的目的地址.如果数据包的目的地址属于一个 multicast 组,则此数据包被接收.multicast 数据包接收的过程和 unicast 包很相似,而且目的地址的匹配通常是由硬件来完成的,因此可以说 Ethernet 能够支持 multicast 操作,但这种 multicast 是不可靠的,因为它不能保证在所有的目标结点,如 $d1$,$d2$,$d3$ 和 $d4$ 都一定能够正确收到源结点 s 发来的消息.

* 本文研究得到国家863高科技项目基金资助.作者申俊,1969年生,博士生,主要研究领域为并行机群系统,通信系统.郑纬民,1946年生,教授,博士导师,主要研究领域为并行处理.王鼎兴,1937年生,教授,博士导师,主要研究领域为并行/分布处理.沈美明,女,1938年生,教授,博士导师,主要研究领域为并行/分布处理.

本文通讯联系人,申俊,北京 100084,清华大学计算机科学技术系

本文 1997-04-14 收到原稿,1997-06-05 收到修改稿.

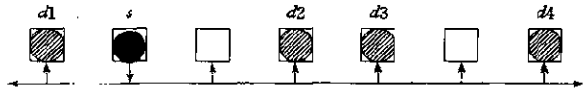


图1　Ethernet multicast过程

2　常用 multicast 算法分析

在很多开关网络中，如 ATM，Myrinet，网络支持在多条链路上同时进行通信，因此在研究 multicast 算法时，需要精心设计 multicast 树，使 multicast 的效率最高. 而 Ethernet 则不同. Ethernet 上所有的结点共享一条总线进行消息传递，任何时候网络中只能存在一条点对点的通信通路，因此开关网络上采用的 multicast 树的方法就不适用于 Ethernet. 但 Ethernet 支持不可靠的 multicast，这是 Ethernet 最大的特点. 我们可以直接在其不可靠 multicast 通信基础上实现可靠的通信机制，使得 Ethernet 的硬件特性得以充分发挥.[1]

在 Ethernet 上实现可靠 multicast 的方法有多种，下面将对其中最有代表性的 3 种算法的优缺点进行分析.

(1) 叠代算法

叠代算法是一种最简单的可靠多播算法. 在这个算法中，消息以可靠点对点通信方式分别给各目的结点发送消息，如图 2 所示. 由于点对点的通信是可靠的，因此整个 multicast 也可自然保证是可靠的. 叠代算法处理简单，不过由于发送结点开销大，其延迟时间会随结点数的增加而线性增大，算法的可扩展性很不好. 另外，由于算法没有利用 Ethernet 支持 multicast 的功能，因此通信效率也不高. 叠代算法一个典型的应用就是 PVM 中 multicast 的实现. PVM 之所以采用这种算法实现可靠 multicast，关键是考虑叠代算法简单，适用范围广，可以应用到各种类型的网络系统.

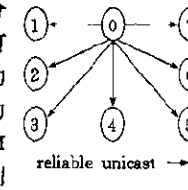


图2　叠代算法

(2) ACK 算法

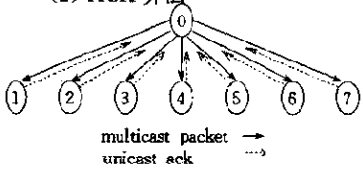


图3　ACK算法

ACK 算法通过消息应答机制，在 Ethernet 不可靠 multicast 基础之上实现可靠的通信，即发送方先以 multicast 方式把消息包同时发给各个结点，然后等待所有目标结点的应答消息 ack. 接收方收到一个 multicast 包后，立即以点对点的通信方式给发送方返回一个 ack. 发送方如果在规定的时间内收到 ack 的个数与目标结点数相同，说明消息已经可靠到达各目的结点，一次通信过程结束. 否则认为数据丢失，于是，发送方重新多播前一个数据包，直到每个结点都正确收到此数据为止，如图 3 所示. ACK 算法减少了发送结点的通信开销，但由于应答消息数随结点数的增加而增多，势必会引起网络通信的竞争，同时发送方处理 ACK 的开销也线性增长. 可见 ACK 算法的可扩展性也不太好，性能相对也低.

(3) NAK(negative acknowledgment)算法

第 3 种方法是 NAK 算法，其目的主要是为了减少 ACK 算法中应答消息过多的问题. ACK 算法是通过接收方对每个数据包发送一个应答消息来确认消息已被接收. 而 NAK 算法却是采用一种完全相反的方式来确认消息的正确接收. 在 NAK 算法中，接收方只对认为已经丢失的数据包发送一个错误应答消息 nak，当发送方收到一个 nak 消息后就重发此数据包. 由于网络中数据包丢失的比例比被正确接收的数据包的比例要小得多，因此 NAK 算法能够减少应答的次数，降低网络竞争的可能性. NAK 算法如图 4 所示.

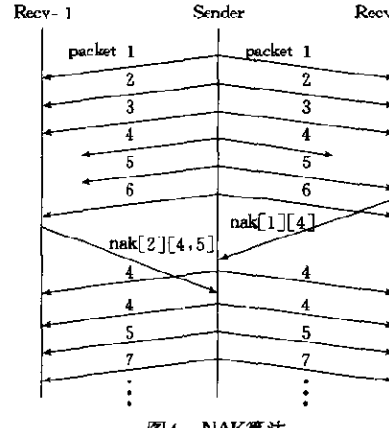


图4　NAK算法

在 NAK 算法中，当多播数据包能够正确传送给各个目标结点时，网络中就不存在应答数据包，因此减小了网络通信负荷，同时还降低了算法的处理开销. 只有当有数据包丢失时，比如结点 Recv-1 检测到丢失两个数据*：packet 4 和 packet 5，于是 Recv-1 马上给 Sender 方发送 nak 消息. nak 消息中包含两项内容，第 1 项代表丢失

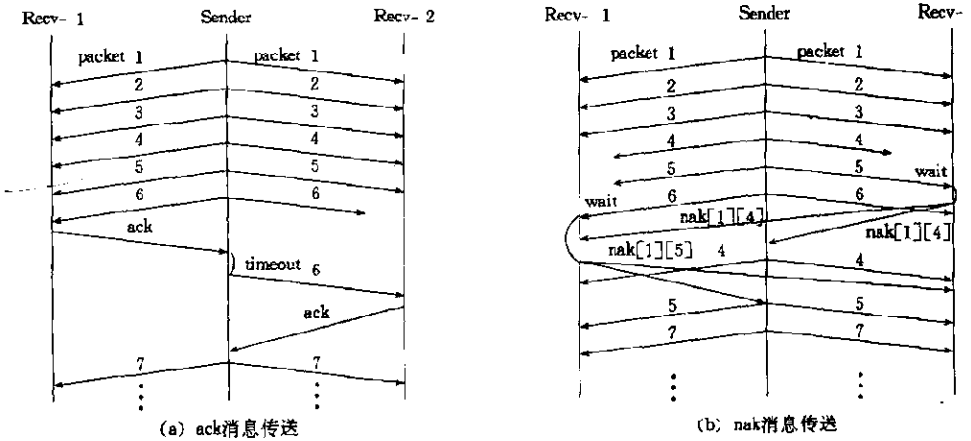
* 消息包校验出错等效为消息包的丢失，其处理方式与丢失处理相同.

数据包个数，第2项说明丢失数据包的序号. Sender 收到 nak 消息后，立即重发 nak 指定序号的数据包：packet 4 和 packet 5，这样就保证了多播的可靠性.

3 新的可靠多播算法 MAK

3.1 MAK 算法的原理

从前面3种多播算法的介绍中可以看出，叠代算法性能最差，ACK 算法其次，NAK 算法最好. 不过我们对 NAK 算法的消息传送过程进行研究发现，它存在3点主要不足. 首先是 NAK 算法对发送方要求较大的缓冲区. 这是由于 NAK 算法中只存在反映数据包丢失的 nak 消息，而 nak 消息只能说明有数据丢失，而不能说明其他数据包是否已经被所有结点正确接收，因此为了保证多播的可靠性，NAK 算法需要在消息发送方开设较大的缓冲区，使已发数据能够在发送缓冲区中保存较长时间. 但这样做仍然不能完全解决问题. 比如在数据包不丢失的情况下，发送方 Sender 一直得不到任何应答消息，发送方就需要保存所有的数据包，显然不管开设多大的缓冲区都不能满足要求，因此在实际缓冲区大小有限的情况下，NAK 算法并不一定能保证多播的完全可靠. 针对 NAK 算法这一缺点，我们在消息传送过程中增加发送一些 ack 消息，用来表示消息的正确接收. 虽然这样带来了一定的通信开销，但可以减少对缓冲区的需求. 其具体实现如图5(a)所示. 在收/发双方均开设相等大小的缓冲区，当发送方缓冲区满时，发送方要求接收方主动发送 ack 消息，如果发送方收到所有目标结点的 ack 应答，表明缓冲区内的所有消息都已被所有结点正确接收，发/接双方都可释放其缓冲区，否则发送方需要采用超时重发机制重发最后一个数据包，直到等待收到所有的 ack 应答为止.



(a) ack消息传送　　(b) nak消息传送

图5　MAK算法

NAK 算法的另一不足之处是接收方不能确定最后一个数据包是否丢失，这是由于 NAK 算法是通过后一个数据包的序号判断前面消息传递中是否有数据包丢失. 因此对最后一个数据包而言，就无法确定此数据是否被正确接收. 而前面对 NAK 算法的改进中，只要让发送方把最后消息包的序号通知各接收方，接收结点收到最后一个数据包后，主动向发送方发送 ack 应答，这样发送方就可根据 ack 应答消息确定最后的数据包是否到达所有目标结点，于是解决了最后一个数据包的可靠传输问题.

NAK 算法的第3个不足之处是其性能还有待于进一步提高. 如图4所示，当接收方 Recv-1 和 Recv-2 都检测到丢失数据包4时，它们都分别向 Sender 发 nak 消息，由于 Sender 分别处理了两次 nak 消息，就连续重发了两次数据包4，带来了不必要的网络通信开销. 造成这种现象的原因是，由于 NAK 算法中接收方同样是以点对点通信方式发送 nak 消息，那么当有多个结点都发现有数据包丢失时，它们都给发送方发送 nak. 但对发送方来说，它只要收到其中一个 nak 就可以进行消息的重发，而不必处理每个相同的 nak，这样就能进一步减少系统中应答消息的传送，有利于提高通信的吞吐率. 因此我们进一步改进了 NAK 算法，如图5(b)所示. 当接收结点检测到有数据包丢失时，接收方并没有立即发送 nak 消息，而是设置一个随机等待时间，只有当接收方的等待时间超过设置值时，而且又没有收到来自其它结点的相同 nak 消息，这时才以多播的方式向发送方和其它接收结点发出 nak 消息. 比如 Recv-2 随机等待一段时间后仍然没有收到数据包4或其它结点发出的 nak 消息，于是 Recv-2 便以多播方式发出 nak 应答. Sender 收到 nak 后立即重发数据包4，而 Recv-1 收到此 nak 消息后，修正已经产生的新 nak 消息内容，使得它从 $nak^{[2,4,5]}$ 变为 $nak^{[1,5]}$，

这样就避免数据包不必要的重复发送，提高了通信效率.

下面通过状态转换图对 MAK 算法进行形式化描述.

3.2 MAK 算法描述

在多播通信时有两类结点：发送结点和接收结点. 其中发送结点在发送数据包的过程中存在 3 种状态，它们的转换关系如图 6 所示.

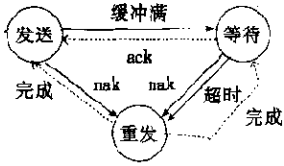


图6　发送结点状态转换图

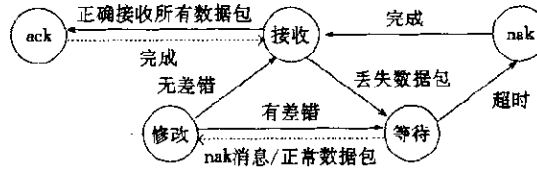


图7　接收结点状态转换图

(1) 发送状态：这是发送结点的常态. 只要发送缓冲区未满，就处于发送状态.

(2) 等待状态：当发送缓冲区满时，系统即从发送状态转换为等待状态，即等待缓冲区被释放. 只有当收到所有目标结点发回的 ack 消息后，才能释放发送缓冲区，并从等待状态返回发送状态. 当系统处于等待状态时，启动一个定时器，定时器计时算法采用设置固定等待时间实现.

(3) 重发状态：即重发消息状态. 这是系统最复杂的一种状态，可以从几个途径到达这个状态：定时器超时可以使系统从等待状态转为重发状态；当收到 nak 消息时，可以从发送或等待状态转为重发状态. 当数据包重发操作完成后自动回到它的前一个状态.

接收结点状态转换图如图 7 所示，它包括 5 个状态：

(1) 接收状态：即正常消息接收状态.

(2) 等待状态：当接收方检测到一个数据包丢失或出错时，则系统处于等待重发消息状态，并启动一个定时器，定时器的超时等待时间随机设置，这样能够使得不同接收结点超时时间各不相同.

(3) NAK 状态，如果定时器超时，则发送结点以多播的方式发送 nak 消息.

(4) 修改出错表状态：当接收方检测到有数据包丢失或出错时，在系统中建立一出错表，用于记录需要重发的消息. 如果处于等待状态的接收方又收到发送结点的数据包或来自其它接收结点的 nak 消息，此结点就需修改出错记录. 如果修改后仍有出错记录，则返回等待状态；否则转为接收状态. 表示数据包已被正确接收.

(5) ACK 状态：当接收方收到最后一个数据包并正确收到所有的其它数据包，则接收方向发送结点发出 ack 消息，并转为接收状态.

4　实验研究

为了对以上算法的性能作进一步的分析，我们在 Ethernet 网上采用不可靠的 UDP(user datagram protocal)协议分别实现了这 4 种可靠 multicast 算法，并对它们的性能进行了测试对比. 我们的实验环境是 10Mbps 普通以太网，8 台 SPARC20 工作站，其中一台作发送结点，其它作为接收结点，并按 1、2、4、6 台进行分组测试，下面是它们的测试结果.

(1) 4 种算法点对点间通信性能

点对点通信性能的对比是用于分析算法实现方式的不可对通信性能产生的影响，如图 8 所示. 叠代算法和 ACK 算法的通信延迟远大于 NAK 算法和 MAK 算法，主要原因是前两种算法是利用停-等协议实现的. 此协议实现方式简单，但延迟时间长；而后两种算法通过缓冲机制有效地降低了通信延迟，但也增加了算法的复杂度. 从图中还可发现，ACK 算法两结点的通信延迟比叠代算法大，这是由于 Ethernet 网中广播通信的开销比 unicast 通信要大. NAK 算法与 MAK 算法相比，由于 MAK 算法复杂些，因此其自身的开销也略大些. 但随着通信开销所占比重的增大，算法复杂引起的开销所起的作用就越来越小了.

(2) 4 种算法多结点多播通信性能

这几种算法多结点多播的延迟时间如图 9 所示. 很明显，叠代算法通信开销增长的幅度最大. 对 1 024 字节的数据包来说，一个结点的通信延迟为 1 790μs，而当 6 个结点时，其延迟时间达到 7 210μs，增长了 3 倍；ACK 算法的性能虽比叠代算法好，但通信延迟仍然随结点数的增加而有较大幅度的增长，如 1 024 字节数据包在一个结点时延迟时间

为 1 883μs，当 6 个结点时，延迟增加到 2 800μs，是原来的 1.5 倍. NAK 和 MAK 算法性能明显比前两种好，它们通信延迟增长幅度很小，当结点数为 6 时，通信延迟的增加幅度不到一个结点通信延迟的 15%. 可见，通过对算法的实际测试，其结果也与前面的分析相吻合.

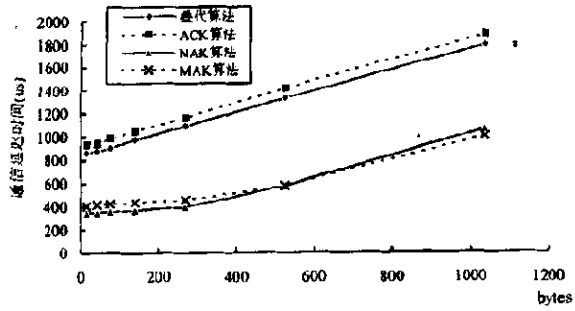
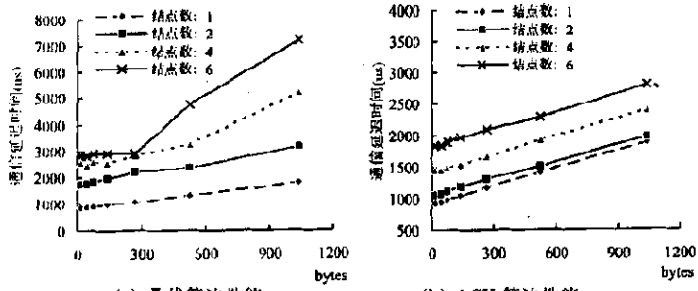
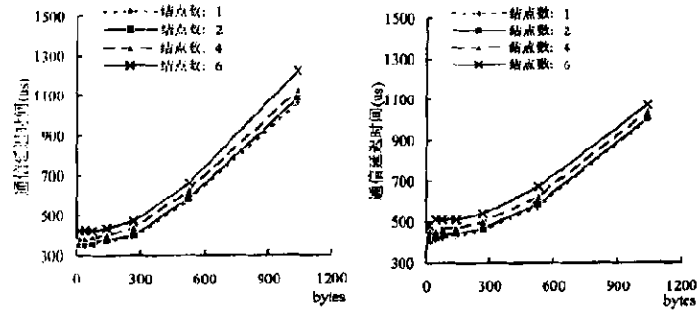


图 8　4 种算法点对点通信性能



(a) 叠代算法性能　　(b) ACK 算法性能



(c) NAK 算法性能　　(d) MAK 算法性能

图 9　4 种算法多结点多播通信性能

(3) NAK 算法与 MAK 算法性能对比

由于这两种算法的性能比较接近，为了进一步比较它们之间的差异，我们分别以 512 和 1 024 字节的数据包为例进行分析，分析结果如图 10 所示. 从图中可看出，数据包大小为 512 字节时，两种算法的性能几乎一样，但随着数据包的增大，MAK 算法的可扩展性明显优于 NAK 算法. 比如，对于 1 024 字节数据包，NAK 算法通信延迟增长率为 31μs/结点，而 MAK 算法的增长率只有 14μs/结点，因此，MAK 算法的性能有了进一步的提高.

5　结束语

可靠多播算法在并行计算中有很广泛的应用，其中常用的算法有 3 种：叠代算法、ACK 算法和 NAK 算法. 这些算法中，NAK 算法的通信延迟时间最小，但它不是一种非常可靠的多播算法，而且性能上还存在有所改进的地方. 于是我们提出了一种新的可靠多播算法：MAK 算法，并通过实验研究验证了此算法的高效性. MAK 算法的提出是基于共享介质的 Ethernet 网，它不适用于开关网络，如 ATM、交换式以太网等，有关这些网络的可靠多播算法还需进一步

研究.

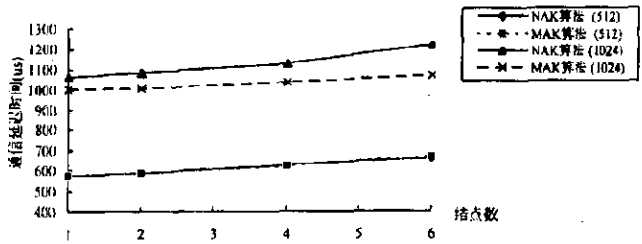


图 10　NAK 算法与 MAK 算法性能对比

参考文献

1 Huang Y, McKinley P K. Efficient collective operations with ATM network interface support. In: Proceedings of the International Conference'96 on Parallel Processing. Los Alamitos, California: IEEE Computer Society Press. Aug. 1996. I-34～I-43

2 Anderson Thomas E, Culler David E *et al*. A case for NOW (networks of workstations). IEEE Micro, Feb. 1995,15(1):54～64

3 Lin Xiao-la, Ni Lionel M. Multicast communication in multicomputer networks. Technical Report, Michigan State University, 1989

4 Susanne E Hambrusch, Ashfaq A Khokhar, Liu Yi. Scalable S-to-P broadcasting on message-passing MPPs. In: Proceedings of the International Conference'96 on Parallel Processing. Los Alamitos, California: IEEE Computer Society Press. Aug. 1996. I-69～I-76

5 周明天,汪文勇. TCP/IP 网络原理与技术. 北京:清华大学出版社,1993
(Zhou Ming-tian, Wang Wen-yong. Network principle and technology with TCP/IP. Beijing: Tsinghua University Press, 1993)

A New Reliable Multicast Algorithm on Ethernet

SHEN Jun　ZHENG Wei-min　WANG Ding-xing　SHEN Mei-ming

(*Department of Computer Science and Technology　Tsinghua University　Beijing*　100084)

Abstract　Multicast is an important communication primitive for parallel programming. In this paper, the authors first analyze three kinds of reliable multicast algorithms generally used on Ethernet, then propose a new reliable algorithm called MAK(multicast acknowledgment) algorithm to obtain low communication latency. Measurements on it also show that the implementation achieves high performance.

Key words　Parallel processing, communications, reliability, algorithms, latency.