MyVIA:一种基于 Myrinet 的 VIA 设计与实现

焦振强[1] 谢 军[1] 陈 渝[2] 都志辉[2]

(中国科学与技术大学研究生院(北京)电子学部 北京 100039)[1]

(清华大学计算机与科学系 高性能计算研究所 北京 100084)[2]

MyVIA :One of Design and Implementation of Virtual Interface Architecture (VIA) based on Myrinet

JIAO Zhen-Qiang[1] XIE Jun[1] CHEN Yu[2] DU Zhi-Hui[2]

(Department of electronics, USTC(Beijing), Beijing 100039, China)[1]

(Department of Computer Science, Tsinghua University, Beijing 100084, China)[2]

Abstract The Virtual Interface Architecture (VIA) is a kind of user-level communication architectures. It is the industrial standard of cluster communication, can provide data communication with low delay and high bandwidth. We analyze different VIA implementation and implement our own design which we call MyVIA based on myrinet. In this thesis , we introduce VIA's principle and background , and discuss some important aspect of MyVIA.

Keywords Cluster, User-level communication , Registered memory , Virtual-physical address translation

1. 背景

近年来工业技术的进步,促进了集群技术的形成与发展,成为当今计算机技术研究的热点。在集群技术中,网络通信是很重要的一个环节,一个高效、可靠的网络通信层,是集群系统的底层基石。

在集群系统中,各主机之间通信频繁,通信量大,延迟小,这是传统的通信协议(如 TCP/IP)所不能胜任的。1997 年,Compaq, Intel, Microsoft 三公司共同提出了虚拟接口体系结构(Virtual Interface Architecture[1], VIA)的定义规范,作为集群通信的工业标准。

依据 VIA 定义规范,在世界范围内很多科研机构展开了实验研究,并且已有不少成果,比如 BVIA[2]、MVIA[3]、FirmVIA[4]、THVIA[5]等等。我们研究了各种不同的 VIA 实现方案,并研制出基于 Myrinet 的 MyVIA,在性能上比目前唯一的同类 VIA 实现—BVIA 有较大的提升。

2. VIA 原理

传统的数据传输协议(如 TCP/IP),影响其传输性能的主要有两方面的因素:第一协议本身结构庞大,过于复杂,发送和接收过程中要处理很多环节。第二从实现方式上来看,传统的数据传输协议一般在核心态下实现,发送和接收数据都要经过操作系统调用,其间有多次的数据拷贝,这些操作带来了很大的延迟。

VIA 针对传统数据传输协议的弱点,定义了一种简洁的通信模型。它提供一种用户层通信的机制,用户具有直接对网卡进行访问的能力,在数据的发送和接收路径中,没有操作系统的介入,从而有效地消除了传统传输协议的弱点。

根据 VIA 规范,应用程序通过虚拟接口 VI(Virtual Interface)来同网络设备进行交互,完成数据传输。每一个 VI 是一个传输端点,不同节点上的 VI 可在逻辑上相连,形成一个双向的传输通道。一个进程可以有多个 VI,与多个远端主机上的 VI 相连。每个 VI 由两个工作队列组成:一个发送队列和一个接收队列。应用程序以 VIA 描述子(VIA descriptors)的形式来进行发送和接收数据的请求,它把描述子放进工作队列,VI 提供者依据 VI 发送队列中描述子的信息进行相应的数据发送,或依据 VI 接收队列中描述子的信息进行数据的接收。每个工作队列都有一个门铃(Doorbell)与之相关联,应用程序把描述子放入工作队列中后要"按"相应的门铃来通知 VI 提供者工作队列中有了新的描述子。VI 提供者检测到门铃后,对描述子进行处理。一个描述子描述一次数据的接收或发送,它包含一个控制段 CS(Control Segment),包含零个或多个数据段 DS(Data Segmenst)。数据段中有缓冲区地址及长度等信息。VI 提供者完成了一次数据发送或接收后,要设置描述子控制段中状态域的完成标志,应用程序可以通过检查状态域来判断描述子的处理情况。另外,在远程直接存储访问 (RDMA)方式下,描述子包含控制段 CS 及地址段 AS (Address Segment),这时地址段中包含有远程相连主机内存中的缓冲区地址。

VIA 是一种用户层的通信模型,它通过引入注册内存等机制,避开操作系统的介入,能提供用户程序和网卡接口之间的直接交互。注册内存的过程,包括锁定这块内存区域于物理内存中和向 VI 网卡接口提供虚-实地址转换两部分,这些操作,只能在核心态下完成。引入注册内存机制后,一块内存区域,只需陷入核心注册一次,以后就可以在用户态下多次地使用,从而把引起较大延迟的内核操作从数据传输的关键路径中剔除出来。

VIA 定义了两种数据传输方式:传统的发送/接收(S/R)方式和远程直接存储访问(RDMA)方式。在 S/R 方式下,发送 VI 的发送队列中的一个描述子描述一次数据发送,描述

焦振强 硕士生,主要研究方向为高性能通信,并行计算。谢 军 硕士生,主要研究方向为高性能通信,并行计算。陈 渝 博士后,主要研究方向为高性能通信,并行操作系统,并行容错技术,并行编译,网络计算。都志辉 博士后,副教授,主要研究方向为并行算法,高性能通信,网格计算,并行编译。

子内含有待发送数据的起始地址及数据长度等信息，VI 网卡对描述子进行处理时根据描述子中的信息把数据传送到网络上。而在与发送 VI 逻辑上相连的接收 VI 的接收队列内则应有描述子来描述这次数据接收，接收到数据后根据描述子中的信息把数据放到接收缓冲区中。在 RDMA 方式下，发起数据传输的一端不但要指明本地数据存放的地址，还要指出远程主机上数据存放的地址。这些信息也都是存放在描述子中的。这种方式下，不要求远程 VI 中必须有相应的描述子与传输发起端对应。

3. Berkeley 实现分析

Berkeley 大学在 Millennium Project[6]中设计并实现了一个 VIA 原型，简称 BVIA。其基于的通信硬件是 Myricom 公司的高性能 Myrinet 网络，我们详细分析了 BVIA 的实现，对其性能进行了测试。

1. 硬件环境

我们的实验环境是：双 Intel Pentium III 733 CPU 微机，512M 内存，Linux Redhat6.2(内核版本 2.2.14)，Myrinet LANai4.1 网卡，16 端口的 Myrinet 交换机。

LANai4.1 网卡有一个主频为 33MHz 的 CPU，容量为 1M 字节的 SRAM，能提供 2×1.28 Gbps 的网络连接，32 位 PCI 总线接口，CPU 访问卡上内存也是 32 位的。在网卡上还有三个 DMA 引擎，可分别控制主机↔网卡间，网卡→网络发送，网卡←网络接收的 DMA 数据传输。在网卡上执行的程序，称为 Myrinet Control Program，简称 MCP 程序。

2. BVIA 实现中的几个主要方面

1)虚-实地址转换　BVIA 的虚-实地址转换过程是在网卡上进行的。当注册内存时，主机锁定物理内存页面，求出对应页面的实际物理地址，保存于主机内存中，并把内存中保存地址转换信息的区域的物理地址指针提交给网卡，网卡进行地址转换时根据上述物理地址指针以 DMA 方式主动从主机内存中获得需要的转换信息。为了减少 DMA 操作，BVIA 引入了 TLB(Translation Lookaside Buffer)缓存。在进行上述的转换后，就把转换的结果保存到这个 TLB 表里，以后需进行地址转换时，首先从 TLB 表里查找，找到了就可以直接使用，如找不到，再进行上面所述的转换过程。这样，一般地，一个虚-实地址转换仅在第一次时需经过复杂的转换过程，以后就可以直接从 TLB 中查找得到，大大提高了效率。

2)门铃的实现　对于每个工作队列，用网卡上内存中的一个 8 字节区域来实现门铃机制。在加载内核驱动模块时，网卡上的全部 1M 内存区域，被映射到了内核地址空间中。在创建 VI 时，这块内核地址空间中的一个页面被再次映射到用户空间中去，其中的 16 字节被用来作为这个 VI 发送队列和接收队列的门铃对。这样，可在用户态下直接对相应的门铃进行操作，而不必陷入核心态。

3)VI 的发送、接收过程　VI 的发送过程简述如下：首先，应用程序构造发送描述子，放入 VI 的发送队列中，并按下门铃。网卡上的 MCP 程序执行逻辑根据门铃中的信息进行地址转换后，使用物理地址以 DMA 方式把描述子从主机内存中拷贝到网卡上，然后分析描述子中的信息，进行地址转换，通过 DMA 控制器把欲发送数据从主机内存中拷贝到网卡上的缓冲区内，再通过网卡上的发送 DMA 发往网络，最后，置主机内存中描述子状态域的完成标志，这也是以 DMA 方式完成的。至此发送过程完成。

VI 的接收过程与发送过程类似，但方向相反。主要不同的地方是网卡上有 2 个接收缓冲区，在对一个接收缓冲区内的数据进行处理的时候，另一块缓冲区可以同时接收数据。

4. MyVIA 设计实现

在分析 BVIA 的过程中，我们发现其实现上比较注重全面实现 VIA 的功能，而在性能方面，应还有进一步提升的空间。考虑多方面的因素，我们提出了自己的实现方案，即 MyVIA。下面介绍我们工作的主要方面：

1. 物理描述子及地址转换

在 BVIA 中，虚-实地址转换是在网卡上进行的，尽管有 TLB 表的协助，但在网卡 CPU 功能较弱这个事实下，决定把地址转换的工作放在主机中进行。我们重新设计了另一种描述子，称为物理描述子(Physical Descriptor)简称 PD。物理描述子是 VIA 所定义的描述子的一个子集，其中存放的所有的地址信息都是经过转换后的物理地址。用户描述数据发送的描述子要转换成物理描述子后再传给网卡，MCP 程序根据物理描述子中的信息进行操作。

2. 注册内存管理

对于注册内存，我们在内核中建立了一张数据表，称为注册内存管理表(Registered Memory Manage Table，RMMT)，注册内存的所有信息都保存在 RMMT 里。这块 RMMT 表区域，在用户打开设备时，就把它以只读方式全部映射到用户空间中。这样当应用程序进行发送、接收操作时，就可在用户态下直接访问 RMMT 的内容，进行 VIA 描述子到物理描述子的转换。

另外，为了减少数据发送、接收过程中由于缓冲区不连续而造成的多次 DMA 传输，我们提供了一个接口函数，用这个函数，应用程序可以一次性地分配在物理地址上连续的大小为 MTU(在这里为 100k)的数据缓冲区，对这块缓冲区的注册管理，信息也保存在 RMMT 里。

3. VI 的实现

在 BIVA 中，VI 的发送及接收队列是在主机内存中实现的。MCP 程序根据 doorbell 中的信息，需要一个则以 DMA 方式从主机内存的队列中取一个进行处理。由于 DMA 方式对于小数据量的数据，其效率不高，而一个 VIA 描述子也只有几十个字节大小，经过转换后的物理描述子则更小。基于这种考虑，我们把发送、接收队列的实现放在了网卡上，VIA 描述子转换为 PD 以后，直接以编程 IO 的方式写到网卡上的工作队列中去。而且这种方式下，写 PD 的操作是由主机 CPU 来完成的，不是由网卡 CPU 来主动操作的，从这一点上来讲，简化了 MCP 程序的处理，增加了主机与网卡处理的并行程度。

但这样的处理方式，也引发了一个问题，即网卡内存的限制。网卡上的内存只有 1M，要支持多个 VI，则在网卡上为单个 VI 的发送及接收队列所开辟的缓冲区不可能很大，不能满足应用程序的要求。为了解决这个问题，我们在主机内存中开辟一块内存区域作为 PD 的二级缓存，当网卡上 VI 的工作队列写满了以后，后序的 PD 就暂存到主机内存的二级缓存中。由于 VIA 的发送、接收有严格的顺序性，所以在网卡上的 VI 工作队列及主机内存中的 PD 的二级缓存区域，都是以 FIFO 环形队列的方式来使用的，由于这个特征，我们把网卡上的工作队列形象地称为小环(Small Ring，SR)，而把主机内存中的 PD 二级缓存内的 PD 队列称为大环(Big Ring，BR)。

当应用程序在小环写满，把 PD 缓存在大环中的时候，要把小环头部的一个标志位置位来表示大环中有数据。当小环中的 PD 个数到达了一个下限值以后，并且检测到了这个标志位就向主机发出中断，主机在中断处理过程中把大环中的 PD 搬移到小环中去。大环被搬空后，要把小环头部的标志位清除。MCP 程序中 PD 处理的下限值，应仔细选取，它不能取得太大，否则会产生太多中断，对主机造成影响，也不能取得太小，应能保证 MCP 在处理余下的 PD 过程中，主机中断处理能够把缓存在大环中的 PD 写到小环中，这样才能使 MCP 程序全速进行数据的发送、接收，不影响效率。这个值应在实验中具体确定。

我们以 VI 中的一个工作队列为例（发送队列或接收队列），来图例说明 SR，BR 及中断处理程序之间的关系。如图 1 所示。

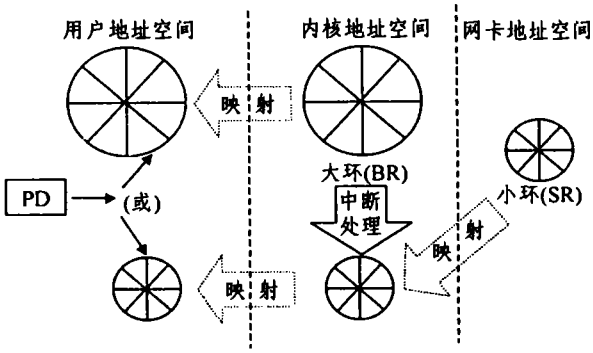

图 1　BR、SR 及中断处理程序逻辑关系

4. 关于 doorbell

对于 doorbell，在 BVIA 中是有明确的实现的。追究门铃机制的本质，是为了通知网卡有描述子待发送或有描述子用于接收。在我们的实现中，没有明确的门铃概念，因为通过对网卡上小环控制信息的检测，就能确定是否有描述子待发送，这实际上就是门铃的功能。

5. 发送、接收处理过程

BVIA 在处理发送、接收的过程中，都是以线性的方式来进行的。比如在发送的时候，当用主机-网卡间 DMA 把数据从主机内存中拷贝到网卡上以后，再利用网卡→网络的发送 DMA 把网卡上的数据发送到网络上，这个时候，主机-网卡间的 DMA 控制器是空闲的，完全可以利用它来进行别的操作。而在 BVIA 中，网卡上 CPU 的处理仅仅是在等待 DMA 完成的同时，再去检测一下 doorbell 罢了。这样的处理方式，没有发挥网卡上各功能部件的处理能力。

为了改善发送、接收处理过程，在详细地分析了网卡上的内存资源利用情况之后，我们提出了自己的传输控制策略，称之为基于状态机的软流水机制。

在卡上内存资源允许的情况下，我们又增加了一个 100k 的发送缓冲区，这样在卡上共有 2 个发送，2 个接收共 4 个缓冲区。我们把这 4 个缓冲区及 3 个 DMA 控制器（主机-网卡间，发送，接收）都作为一种资源用相应的数据结构加以描述，而把发送、接收的处理过程分解为各个阶段。这样，数据传输的过程变成了各描述子利用资源在不同的中间状态下一步步跃迁直至完成的过程。

在我们的设计中借鉴了 CPU 指令流水处理的作法。在极端情况下，在网卡上可以有 2 个发送，2 个接收共 4 个描述子同时进行处理。各个描述子的处理在不同的阶段需不同的资源，它们对资源的使用是采用竞争的方式，即一个描述子获得需要的资源后把它锁定，使用完之后再解锁。在一个描述子请求一个资源而得不到或者等待某项操作完成的时候，不是让 CPU 处于盲目的等待状态，而是继续其它描述子的处理，在 CPU 主循环再次对此描述子进行处理的时候，再来检测它所请求的资源状态或者操作完成状态。

在此值得提出的是，在实际的实现过程中，我们发现网卡上接收 DMA 控制器资源的使用，实际上是依赖于接收缓冲区资源的，而不用对它进行单独控制。这样程序中所定义的 DMA 资源实际上只有 2 个（主机-网卡间 DMA，发送 DMA）。

5. 性能测试与比较

实验室中，我们对 BVIA 及 MyVIA 的单边延迟和带宽进行了测试，结果如图 2、图 3 所示。

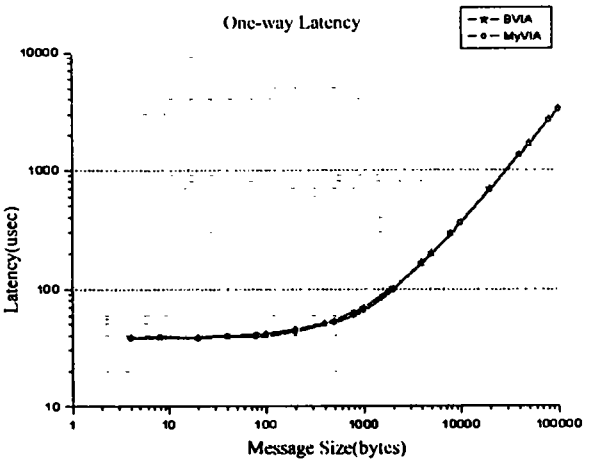

图 2　单边延迟

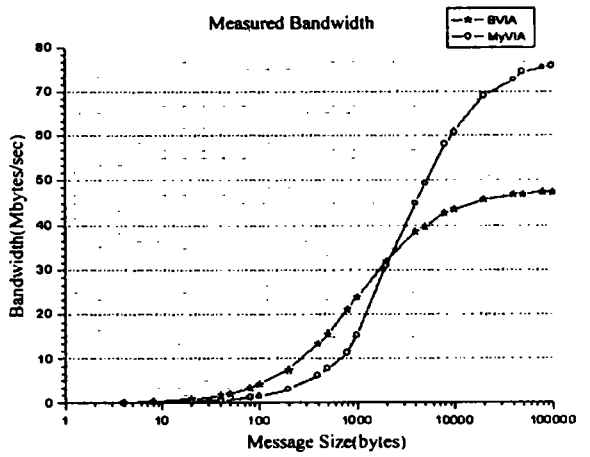

图 3　带宽

为了能够看清小数据包时的细节和整个数据 MTU 范围内的性能，在单边延迟测试图中，横坐标和竖坐标都采用了对数坐标。在带宽测试图中，横坐标采用了对数坐标。

图 2 中单边延迟的测试结果，是用所测大小的数据包在两台主机间往返 1000 个来回后求出单方向上的平均时间得到的。由图中，我们可以看到，BVIA 与 MyVIA 的测试结果曲线几乎重叠，在性能上相差不大。

图 3 中带宽的测试结果，是由 一台主机连续向另一台主机进行 2000 个所测大小的数据包的满负荷发送，而不管接收

（下转第 37 页）

后，任务管理组件要求环境管理组件检查每一个组成任务的服务的状态，保持相应的状态和文件，接着在(3)中环境管理组件回收这些服务，并且保存相应的状态和文件到一个文件服务器(4)。然后，任务管理组件检查演讲者的时间安排，了解到他的行踪动向，然后传送相应的信息(包括抽象任务描述)到汽车的任务管理组件(5)；这个消息将触发汽车的任务管理组件发生相应的状态转移(b)，同时将引起它要求汽车环境管理组件(6)去得到更新的演讲者的工作描述(7)。

通过以上的操作，汽车任务管理组件将了解到那些文件是继续演讲者工作必需的文件，然后任务管理组件将要求汽车的环境管理组件提取这些文件(8)，汽车环境管理组件检查当地的文件是否已经更新，如果没有，则提取这些更新的文件(9)。一旦汽车的方位监测组件了解到演讲者已经进入汽车环境，它就会立即通知任务管理组件(10)，这将引起任务管理组件经历相应的状态转变(c)。然后将向环境管理组件提出要求完成任务的服务支持(11)，接着环境管理组件在相应的分配的虚拟服务组件上恢复执行的状态，然后，工作可以在汽车环境中继续进行了。

小结 遍在计算涉及到一个范围较广的环境，而且环境中的设备，不仅数量众多，而且结构各异。如何在这样的环境中，为用户提供连续、平滑的服务是软件工作者所面临的挑战。本文中所提出的基于中间件的体系结构框架主要包括两个隔离层：虚拟服务和环境管理组件屏蔽下层设备的异构性，任务管理组件向用户提供个人空间的概念。同时将用户任务视为多个组件的联合，在不同的环境中，可以用不同的组件组合完成同一个任务，同时提供与移动代码的兼容和支持小设备的接入。

参考文献

1 Weiser M. The computer for the twenty-first century. Scientific American, 1991, 265(3): 94～104

2 Arnold K, O'Sullivan B, Scheifler R, Waldo J, Wollrath A. The Jini Specification. Addison-Wesley, 1999

3 Wang Z, Garlan D. Task-Driven Computing: [Technical Report, CMU-CS-00-154]. School of Computer Science, Carnegie Mellon University, May 2000

4 Sousa J P, Garlan D. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. In: Proc. of the 3rd Working IEEE/IFIP Conf. on Software Architecture 2002, Montreal, Aug. 2002

5 Gosling J, Joy B, Steele G. The Java Language Specification. Addison-Wesley, 1996

6 Grimm R, et al. Systems Directions for Pervasive Computing. In: Proc. of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), Elmau, Germany, May 2001. 128～132

(上接第 29 页)

方接收的情况下所得到的结果。由图中可以看到，在小于约 2000 字节大小数据包的情况下，MyVIA 的带宽性能较 BVIA 为低，而在 2000 字节以上，MyVIA 的带宽性能要比 BVIA 高。在 MTU 范围内，BVIA 的带宽最大可达到 47.46Mbytes/sec，而 MyVIA 可达到 76.2Mbytes/sec。

分析在发送 2000 字节下的小数据包时，MyVIA 的带宽性能比 BVIA 低的原因，是由于 MyVIA 实行新的基于状态机的软流水机制，对网卡上各资源进行开锁、解锁的竞争使用方式后，使 MCP 程序的控制逻辑复杂了许多，而网卡上的主频为 33MHz 的 CPU 的处理能力较弱，并且在此时的满负荷发送情况下，由于小数据包的发送较快，网卡上主机-网卡间 DMA 控制器与发送 DMA 控制器没有能够进行并行处理或并行程度不够大，从而使 MyVIA 比 BVIA 在软件控制上的执行延迟大的弱点充分暴露出来。在 2000 字节以上时，网卡上主机-网卡间 DMA 控制器与发送 DMA 控制器的并行处理所带来的好处抵消了 MyVIA 在复杂的软件控制上的引起的延迟，从而使带宽比起 BVIA 得到了进一步的提升。

6. 当前工作总结及今后的任务

Myrinet 网络提供了一个灵活的硬件平台，其网卡的可编程特性使我们可以设计实现自己的各种数据传输方案，并进行实验验证。

我们对 BVIA 的分析及 MyVIA 的设计，是对 VIA 研究的进一步深入。MyVIA 的实现取得了一定的成绩但同时也暴露出不少的问题，需要我们在今后的工作中加以解决。

我们下一步的工作，将在现有的基础上对 MCP 程序进行调整，尤其是对小数据包的处理。并且，我们准备将整个设计方案移植到 Linux2.4 内核下，使用 Myricom 公司的 LANai9.x 网卡，这是其最新的产品，网卡上 CPU 的主频为 200MHz，卡上可用内存 2M，可以进行 64 位 PCI 总线数据的传输处理。

总结 本文介绍了 VIA 的原理，分析了 Berkeley 大学在 Myrinet 网络上对 VIA 的实现，并介绍了我们自己的 MyVIA 实现的主要方面，最后通过 BIA 与 MyVIA 的性能比较分析，指出了我们实现中的不足之处，对我们今后的工作做出指导。

参考文献

1 Virtual Interface Architecture Specification. Version 1.0. Compaq, Intel and Microsoft Corporations, Dec 16, 1997. Available at http://www.viarch.org

2 Buonadonna P, Geweke A. An Implementation and Analysis of the Virtual Interface Architecture. University of California at Berkeley, Computer Science Department, Berkeley, California, May 1998. Available at: http://www.millennium.berkeley.edu/proj/Vineyard/

3 M-VIA: A High Performance Modular VIA for Linux. http://www.nersc.gov/researc h/FTG/via/

4 Banikazemiy M, et al. Effient Virtual Interface Architecture (VIA) Support for the IBM SP Switch-Connected NT Clusters. Available at: http://nowlab.cis.ohio-state.edu/projects/via/

5 Du Zhihui, et al. Hardware Based TH-VIA User Level Communication System Supporting Linux Cluster Connected by Gigabit THNet. DCABES 2001 Proceeding

6 The UC Berkeley Millennium Project. http://www.millennium.berkeley.edu

7 Bhoedjang R A F, Rühl T, Bal H E. User-Level Network Interface Protocols. 0018-9162/98/ IEEE November 1998

8 董春雷，郑纬民. 基于 Myrinet 的用户空间精简协议. 软件学报，1999.3