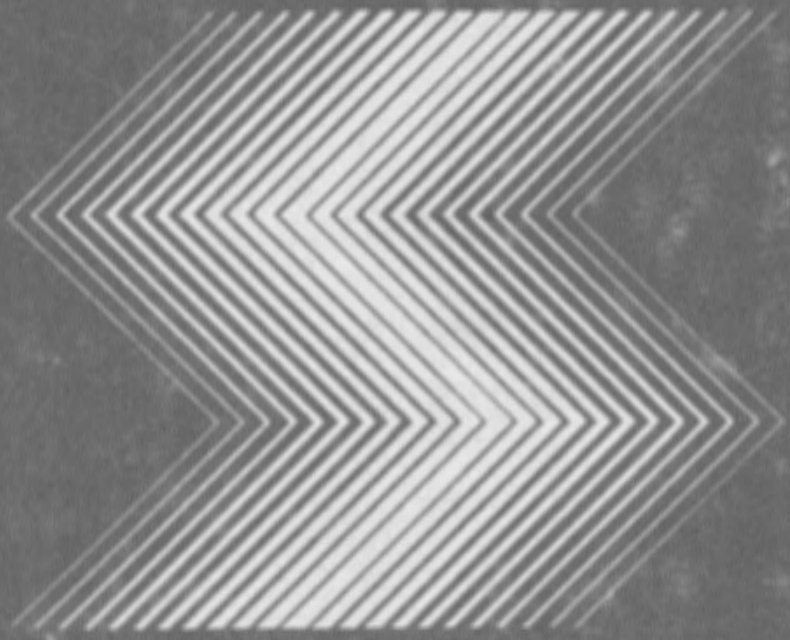
Computer Communications Review
Volume 18, Number 4
August 1988

acm PRESS

SIGCOMM '88 SYMPOSIUM

Communications Architectures & Protocols

Stanford, California
August 16–19, 1988



SPONSORED BY ACM SIGCOMM WITH SUPPORT GIVEN BY THE INFORMATION SCIENCES AND TECHNOLOGY CENTER OR SRI INTERNATIONAL.

The Association for Computing Machinery
11 West 42nd Street
New York, New York 10036

ISBN O-89791-279-9

Additional copies may be ordered prepaid from:

ACM Order Department
P.O. Box 64145
Baltimore, MD 21264

Price:
Members $20.00
All others $26.00

ACM Order Number: 533880

The VMP Network Adapter Board (NAB): High-Performance Network Communication for Multiprocessors

Hemant Kanakia
Computer Systems Laboratory
Stanford University

David R. Cheriton
Computer Science Department
Stanford University

Abstract

High performance computer communication between multiprocessor nodes requires significant improvements over conventional host-to-network adapters. Current host-to-network adapter interfaces impose excessive processing, system bus and interrupt overhead on a multiprocessor host. Current network adapters are either limited in function, wasting key host resources such as the system bus and the processors, or else intelligent but too slow, because of complex transport protocols and because of an inadequate internal memory architecture. Conventional transport protocols are too complex for hardware implementation and too slow without it.

In this paper, we describe the design of a network adapter board for the VMP multiprocessor machine that addresses these issues. The adapter uses a host interface that is designed for minimal latency, minimal interrupt processing overhead and minimal system bus and memory access overhead. The network adapter itself has a novel internal memory and processing architecture that implements some of the key performance-critical transport layer functions in hardware. This design is integrated with VMTP, a new transport protocol specifically designed for efficient implementation on an intelligent high-performance network adapter. Although targeted for the VMP system, the design is applicable to other multiprocessors as well as uni-processors.

1 Introduction

Performance of transport protocols on multi-megabit communication networks tends to be limited by overhead at both the transmitter and receiver. For example, measurements of the V kernel [5] indicate that network transmission time on Ethernet accounts for only about 20 percent of the elapsed time for transport-level communication operations, even with its highly optimized protocol. Similar performance figures have been reported in [15, 17]. Although processor and memory cycle times keep improving, with communication networks moving to gigabit range, we expect the processing to persist as a bottleneck - unless significant improvements in network adapter and transport protocol designs are achieved. We identify three major problems with current designs.

First, the host-to-network adapter interface imposes excessive overhead, particularly on a multiprocessor host, in the form of processor cycles, system bus capacity and host interrupts. The processing overhead arises from calculating end-to-end checksums, packetizing and depacketizing as well as encryption, in the case of secure communication. The memory-intensive processing required by these functions reduces the average instruction execution rate especially for a high-performance processor such as MIPS [14] in which memory reference operations are proportionally much slower than register-only operations. This processing causes the data to move at least twice over the system bus; once from global memory to the processor (or its cache), and once when the packet is copied to a network adapter. The increased traffic wastes system bus bandwidth, a critical resource in multiprocessor machines. In current host-to-network adapter interfaces, a host is interrupted for each packet received or transmitted. These per-packet interrupts force frequent context switching with the attendant overheads, with a high penalty in the multiprocessors system with processor caches, where it may be necessary to fault code and data into the cache before responding to the interrupt. In addition, the context switch may also incur contention overhead when data associated with the network module is resident in another cache. The problem is further aggravated by the prospect of networks moving to the 100 megabit up to the gigabit range using fiber optics [11, 13, 16]. For instance, in a file server attached to a 100 megabit network with the interface interrupting on every 2 kilobyte packet, the network interrupts every 200 microseconds under load, hardly sufficient to do even minimal packet processing.

Second, the so-called "intelligent" network adapters that implement transport-level functions have lower performance at the transport-level as compared to the alternative system where a network adapter does programmed I/O transfers and a host performs transport protocol functions. The primary reason is an inadequate internal memory architecture. Currently, the data transfers into and out of the buffer memory reduces the number of memory cycles available for packet processing. The future system bus technology, with a high transfer rate and the burst-mode transfer, and the future networks, with a high data rate, will make this problem even more acute.

Finally, conventional transport protocols are too complex or awkward for hardware implementation and too slow without it. For a large packet, the processing cost incurred in checksumming and encryption dominates the packet processing, since the cost increases in proportion to the size of a packet. Hardware implementation for such key performance-critical functions would substantially increase performance, but the packet formats of

conventional transport protocols are does not facilitate hardware support or implementations.

An additional factor that motivates rethinking network adapter architecture is the problem of a host being bombarded by packets from one or more other hosts. The packet arrival rate, especially in a high-speed network, can exceed the rate at which a host can process and discard these packets, effectively incapacitating the host for useful computation. Excessive packet traffic can arise either from failures or malicious behavior of a remote host. A well-designed network adapter acts as a "firewall" between the network and the host.

In this paper, we describe the Network Adapter Board (NAB) for the VMP multiprocessor [3], focusing on the architectural issues in the host interface, the adapter board, and the transport protocol. The adapter host interface architecture is designed for minimal latency, minimal interrupt processing overhead and minimal data transfer on the system bus. The NAB uses a novel internal memory and pipelined processing architecture that implements some of the key performance-critical transport layer functions in hardware. The design is coupled to a new transport protocol called *Versatile Message Transaction Protocol (VMTP)* [1]. VMTP is a request-response transport protocol specifically designed to facilitate implementation by a high-performance network adapter. VMTP assumes an underlying network (or internetwork) service providing a datagram packet service, and includes support for multicast, real-time, security and streaming. The interested reader is referred to an Internet RFC [1] for further details. For brevity in this report, we focus on aspects of the protocol relevant to the cost of communication and the design of the VMP network adapter. We describe the expected performance of the NAB prototype being built. Although targeted for the VMP system, the design is applicable to other multiprocessors as well as uni-processors.

The next section describes the host-to-network adapter interface architecture. Section 3 describes the internal architecture of the network adapter board. Section 4 describes details of a prototype of the network adapter. Section 5 indicates the (expected) performance for this adapter. Section 6 compares our design to related work on this problem. We conclude with a summary of the current status, our plans to further evaluate this design and the current open problems in this area.

2 Host-Network Adapter Interface

A request/response model of communications is used for information exchange between a network device and a host. A host requests network services with a control block sent to device, and the device returns a response after the service is provided. These messages are transferred across via an interface that appears to the host software as a 1024-byte control register.

To transmit data, the host software writes a control block, the *Transmit Authorization Record (TAR)*, to this control register. The TAR contains control information describing data to be sent including the pointer to data in physical memory. If the data fits entirely within the control register, the data segment description is omitted from TAR. In both cases, the network adapter transmits the data, checksumming and encrypting (if required). Interface interrupts host to inform that a TAR has been used to successfully transmit the data.

To receive data, the host software writes a control block, a *Receive Authorization Record (RAR)*, that "arms" the network interface to receive packets, specifying the maximum size to receive and providing a list of pointers to memory pages in host memory into which to deliver the data. The RAR can specify any one of: (1) a specific source, (2) a class of allowable sources or, (3) any source for the received data. where source is either a transport-level or network-level address.

The interface interrupts the host when the received packet(s) satisfies one of these RARs, returning the RAR, along with the packet header, to the appropriate host via the control register. The returned RAR itself may contain small amounts of data in addition to the corresponding packet header. When the RAR is returned, the data has been already stored in the host memory at the location pointer(s) contained in it, unless the data is contained in the RAR. Incoming packets are discarded if they cannot be matched to an outstanding RAR.

Type, a subfield in the control block, distinguishes the records passed via the control register. Four major types of records, used in transmission and reception of data, are an RAR with small amounts of data, an RAR with data descriptors, a TAR with small amounts of data and a TAR with data descriptors. To optimize host-network adapter interactions, host is allowed to combine issuing of a TAR and RAR, using the same buffers to send and to receive data. Additional types of records are used by a host for various purposes such as to add or delete acceptable destinations, to restrict traffic from a source, to provide decryption/encryption keys to the NAB, to get status information, and to reset the NAB.

The RARs and TARs include a packet header including a network-level header, either small amounts of data or a list of data descriptors, and various control information. The buffer descriptors in a TAR point to locations in the physical memory space where data to transmit is available. The buffer descriptors in an RAR point to locations in the physical memory space where the data is to be received. The returned RAR or TAR contain in addition to the buffer descriptors the number of data words actually received or transmitted. The control information, used by NAB, includes a link field, type of RAR matching to be used, transport-level source and destination addresses, interrupt control, timeout control, a local host number, and a local process identifier. The buffer descriptors are omitted for TARs and RARs containing small amounts of data. A link field, used by the adaptor, allows chaining of these records as necessary. The type of RAR matching to be done indicates if the RAR can be used for receiving traffic only from the specified source or any source.

Interrupt control is used to determine when to interrupt the host identified in the record. On reception, the host, indicated by the host number, is interrupted either when the data of the first packet is stored in the system memory or when the data is completely received or both. The completed RAR is returned via the control register with the length of data received before the host indicated in RAR is interrupted. On transmission, one may have the host interrupted either when the NAB begins processing a TAR, or when the last of the data segment is transmitted. The TAR is written into the control register before a host is interrupted.

In the following, we discuss how this interface efficiently handles small amounts of data, large amounts of data, and also allows the interface to act as a firewall, protecting the host from the network.

2.1 Short Message Handling

The latency with short messages is minimized because the short message is written to the interface as part of the TAR and read as part of the returned RAR on reception. The operating system interrupt handlers for the network adapter can directly copy the message data between the control register and the operating system data structures, moving the higher-layer data to its intended destination with minimal cost. For multiprocessor hosts with per processor cache this procedure also avoids additional cache and bus activity, as we expect the small amount of data to be already available in cache before being sent with TAR or used immediately after having received in an RAR. Thus, the delay introduced in transmitting and receiving a packet with a small amount of data is no more than that incurred with a host directly handling the packet and using the interface as a staging area to send and receive packets.

Note that including the packet header in the TAR means that the processor writes a small amount of data to the interface for transmission yet the network adapter has minimal processing on the data to prepare packets for transmission. In particular, for small data appended to header, the network adapter need not do anything before starting network transmission, given that checksumming and encryption occur as part of transmission.

2.2 Long Message Handling

Host overhead is minimized for the transmission and reception of large amounts of data, typically in the range of 4-16 kilobytes, by passing descriptors rather than actual data. On transmission, the host writes one TAR and receives one completion interrupt, with the network adapter transferring the data from host memory with minimal bus overhead.[1] The network adapter handles the per-packet overhead of packetizing, checksumming, encryption and per-packet coordination. On reception, the host receives a single interrupt for each RAR returned after the data has been transferred into global memory. Again, the per-packet interrupt overhead is handled by the network adapter.

Latency in transfer of large amounts of data is reduced by ensuring minimal bus and memory references. Moving byte-based processing functions such as checksumming and encryption to network adapter reduces memory references to one per data word. Passing buffer descriptors to network adapter which retrieves data from host memory ensures that only one bus transfer per word transferred is required. For multiprocessors with per-processor cache, the buffer-passing model helps reduce number of cache misses one would otherwise incur, as a cache is neither likely to have most of the data transferred nor going to use it in the near future. The cache pollution, resulting from using cache for network data transfers, would also increase cache miss ratio for other applications. An additional factor reducing latency is the use of burst-mode transfer on host bus, which decreases transfer time of data on the bus by a factor of 4.

In sending and receiving groups of packets, the interface can afford to introduce some latency for the first packet of the group as long as the whole transmission and reception has less delay as compared to a host processor handling per packet processing. That is, for a small number of packets K, it should be the case that

$$K * P_{host} > D + K * P_{interface}$$

where writing the control record to the interface introduces a delay D in transmission over the host processor writing the data directly to the network, K is the number of packets to be transmitted, P_{host} is the time for the host to packetize and send one full-sized packet and $P_{interface}$ is the time for the interface to transmit one full-sized packet. The value of K for which this is true should be as small as possible, ideally 1 but certainly less than the common size of a multi-packet packet group. In this interface, the value is close to 1 primarily because $P_{interface}$ is much smaller than P_{host}.

[1] The VMP memory supports block transfer using the VME serial bus protocol, thereby minimizing bus occupancy and arbitration overhead.

2.3 Network Firewall

The interface architecture is designed to allow the network adapter to function as a *firewall*, protecting the host from network packet pollution, both accidental and malicious. In essence, a host incurs no overhead for network packets whose reception it has not authorized; the interface discards all packets that are not compatible with an RAR provided by the host and are not directed to an end-point registered with the adaptor. Some examples of its use follows. If the host does not provide an RAR for broadcast packets, then garbage broadcast packets incur zero overhead on the host processor(s). Multiple responses generated by a multicast request can be limited to only those that fit into the buffer area provided; the rest are discarded without incurring any host overhead. By providing RARs for only those sources it wants to listen to, a process could avoid host overhead required otherwise for pruning out the traffic from unauthorized sources. In general, the authorization model of packet reception plus the speed of the network adapter insulates the host from packet pollution on the network.

3 Network Adapter Internal Architecture

The network adapter internal architecture is designed to provide maximal performance between the host interface and the network architecture. The internal architecture is structured as five major components, interconnected as shown in Figure 1. These components serve the following functions:

Network access controller (NAC) : implements the network access protocol and transfers data between the network and the packet pipeline.

Packet Pipeline : generates and checks transport-level checksums and performs encryption and decryption of data in secure communication.

Buffer memory : a staging and speed-matching area for data in transit between the packet pipeline and the host memory. Its specialized buffer memory permits fast block data transfers between the network and host and provides the on-board processor with contention-free memory access to the packet data.

Host block copier : moves data between the buffer memory and the host memory using a burst-transfer bus protocol, minimizing the latency as well as the bus and memory overhead for transfers.

On-board processor : a general-purpose processor that manages the packet processing pipeline and various bookkeeping functions associated with the protocol.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.