Isolating the Performance Impacts of Network Interface Cards through Microbenchmarks

Technical Report #EE0401

Vijay S. Pai, Scott Rixner, and Hyong-youb Kim

Rice University
Houston, TX 77005
{vijaypai, rixner, hykim}@rice.edu

Abstract

This paper studies the impact of network interface cards (NICs) on network server performance, testing six different Gigabit Ethernet NICs. Even with all other hardware and software configurations unchanged, a network service running on a PC-based server can achieve up to 150% more throughput when using the most effective NIC instead of the least effective one. This paper proposes a microbenchmark suite that isolates the micro-level behaviors of each NIC that shape these performance effects and relates these behaviors back to application performance. Unlike previous networking microbenchmark suites, the new suite focuses only on performance rather than aiming to achieve portability. This choice allows tight integration with the operating system, eliminating nearly all operating system overheads outside of the device driver for the network interface. The results show that the throughputs achieved by both a web server application and a software router have an evident relationship with the microbenchmarks related to handling bidirectional streams and small frames, but not with sends or receives of large frames.

1 Introduction

Rapid developments in application and operating system software have reduced the CPU load of network servers, the amount of main memory used for networking, and the bandwidth requirements of data transfers between the CPU and its main memory [2, 11, 12, 13]. However, these techniques do not address the overheads of processing data at the server's network interface card (NIC), potentially leaving the NIC as a performance bottleneck as network link speeds increase. Modern network servers include one or more Gigabit Ethernet network interface cards (NICs), but many factors limit their ability to achieve the theoretical maximum throughput of those interfaces. Simply replacing the Gigabit NIC in an otherwise identical server can increase the network throughput achieved by an application up to 150%.

This work is supported in part by a donation from Advanced Micro Devices, by the Department of Energy under Contract Nos. 03891-001-99-4G, 74837-001-0349 and/or 86192-001-0449 from the Los Alamos National Laboratory, and by the National Science Foundation under Grant Nos. CCR-0209174 and CCR-0238187.

Although microbenchmarks often provide insight into low-level system behaviors, the recent operating system improvements deployed in servers (such as zero-copy I/O) have not been incorporated into existing application-level networking microbenchmarks such as netperf, LMbench, or the hbench:OS extensions to LMbench [4, 5, 9]. Therefore, these microbenchmarks are too heavyweight either to isolate performance characteristics of NICs from the underlying operating systems or to send and receive data at rates high enough to expose the limitations of NICs. At the other extreme, hardware-based network performance analyzers such as the Spirent Smartbits have often been used to test networking devices, including NICs, but these must act as both the source and sink of data in order to verify their traffic. Thus, such systems cannot isolate specific behaviors of a single NIC, such as send or receive performance alone. Finally, these microbenchmarks (whether performed through applications or hardware) do not directly relate their results to the performance of real network services.

This paper proposes a lightweight microbenchmark suite to effectively isolate the behaviors of modern Gigabit Ethernet NICs and then relates those micro-level behaviors back to the performance of real network services. These microbenchmarks show that most of the six NICs studied are able to achieve near the maximum theoretical bandwidth of Gigabit Ethernet when sending or receiving a continuous unidirectional stream of maximum-sized Ethernet frames. However, performance varies widely when sending or receiving a continuous stream of minimum-sized frames, with the best interface achieving five times the throughput of the worst (and even then achieving less than half of the theoretical limit). The ability of each network interface to handle bidirectional traffic also varies greatly, with up to 73% throughput difference for maximum-sized frames and a factor of seven throughput difference for minimum-sized frames.

While testing six network interfaces is not enough to generate a strong statistical correlation, the behavior of each network interface on certain microbenchmarks has evident relationships with the performance achieved by both a web server and a software router. For example, the tests related to sending or receiving maximum-sized UDP datagrams have little or no correlation with the system performance of these services. In contrast, the tests related to processing bidirectional streams (of either large or small frames) or unidirectional streams of small frames (either send or receive) have a much clearer relationship to system performance. These results indicate that resource contention between send and receive traffic and the fixed overheads of processing Ethernet frames have greater impact on throughput than the size-dependent overheads of a single unidirectional stream. These results also give insight into the shortcomings of particular interfaces and suggest areas of a network interface in which careful design is important.

The rest of this paper is organized as follows. Section 2 describes the behavior and performance of

a high-performance web server and software router using different network interfaces. Section 3 presents the proposed microbenchmarks and explains their interaction with modern network interfaces. Section 4 presents the performance of various network interfaces on the microbenchmarks, and Section 5 discusses the relationship between those results and the measured application performance. Section 6 concludes the paper.

2 Motivation

Web servers and software routers are two representative network services that exercise the network in very different ways. A web server responds to incoming requests by producing and returning the appropriate response, whereas a software router simply forwards incoming data to a different network with little or no modification. Despite placing different demands on the Ethernet network interfaces, the performance of each service changes dramatically with the capabilities of the network interface in the system.

2.1 Web Servers

A web server interacts with the network in two primary ways. The first is by receiving HTTP requests sent by clients. These requests are typically quite small, on the order of 200 bytes of ASCII text. Web clients and servers communicate using TCP, so the server responds to the requests with acknowledgments, leading to minimum-sized (64 byte) Ethernet frames. The second way in which the web server interacts with the network is to send responses to the clients' HTTP requests. The web server either sends static files or dynamically generated content, depending on the type of request. Both types of responses can vary greatly in size—anywhere from empty files to several hundred megabytes. Again, these responses are sent using the TCP protocol over the Ethernet, so the data must first be segmented and encapsulated in Ethernet frames. The maximum size of an Ethernet frame is 1518 bytes, which allows 1460 bytes of TCP content (after subtracting 14 bytes for Ethernet headers, 20 bytes for IP headers, 20 bytes for TCP headers, and 4 bytes for Ethernet CRC). The server sends the data using TCP flow control policies based upon the receipt of acknowledgments.

The network traffic of a web server is largely bimodal in size. Incoming requests and acknowledgments, as well as outgoing acknowledgments, are very small frames, whereas response content includes many large objects which result in maximum-sized frames. Therefore, to achieve high system performance, the network interface in a web server must be able to support traffic volumes dominated by sends of large frames while also receiving and sending small frames.

2.2 Software Routers

A software IP router behaves quite differently from a web server. First, a software router is connected to more than one network. Second, incoming IP packets are not destined for the router itself (except for occasional router control messages); rather, the router's task is to determine the network to which it is connected that will bring the packet closest to its final destination and to forward the packet onto that network. The router typically does not change the contents of the packet, although it will change the Ethernet headers appropriately.

As the traffic that flows through a router is of all types, the size of incoming and outgoing frames varies between the Ethernet minimum (64 bytes) and maximum (1518 bytes). Upon receiving a frame, the router must either forward it over one of the other networks or generate an error and typically send that error back on the receiving network. Therefore, to achieve high system performance, each network interface in a software router must be able to support bidirectional traffic with arbitrary frame sizes.

2.3 Performance

The throughput of a PC-based server running either the thttpd web server [14] or the Click software router [8] depends on the Gigabit Ethernet network interface that is installed. These differences can be seen on a server with an AMD Athlon 2600+ XP processor running the FreeBSD 4.7 operating system, 2 GB of DDR SDRAM, a 64-bit/66 MHz PCI bus, and a single 40 GB IDE disk (none of the workloads are disk intensive). For the web workload, the server is accessed by a set of synthetic clients replaying traces from real web sites using an infinite-demand model. The server and clients are connected using a Gigabit Ethernet switch. For the router workload, the router is accessed by two clients which replay IP packet traces from real routers as fast as the router can handle them without dropping packets. Each host is connected to the router over a different network interface, and the two interfaces of the router are on separate Gigabit Ethernet switches. Each client machine includes an AMD Athlon 2000+ XP processor, 512 MB of DDR SDRAM, and an Intel Pro 1000 MT Desktop network interface.

The six copper-based Gigabit Ethernet adapters under test are the Intel PRO 1000 MT server and desktop NICs, a programmable 3Com NIC based on the Alteon ACEnic with distributed (ACE) and optimized (ACE Optimized) firmware, a Netgear NIC, and an additional 3Com NIC. These NICs will be presented in more detail in Table 1 in Section 3. The comparisons in this paper are not intended to recommend any of these network interfaces over the others, since the different NICs vary in age, expense, and operating system support. Further, this study is far from comprehensive given the large number of NICs available. Rather,

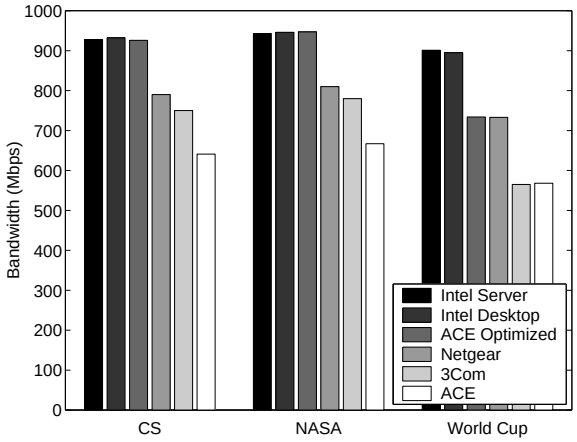

Figure 1: Throughput in Mbps achieved by the thttpd web server running on a PC-based server with various network interfaces on three workloads.

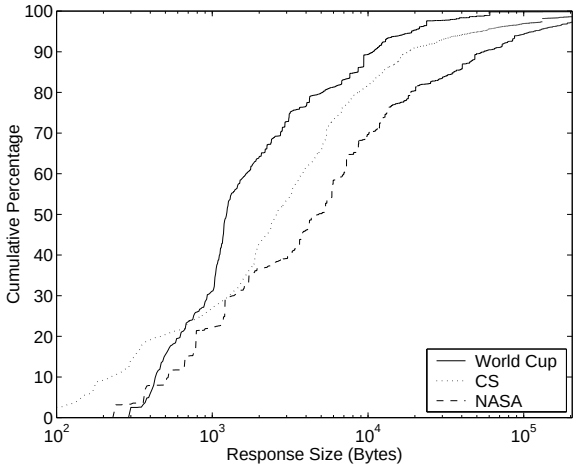

Figure 2: Cumulative distribution of the response sizes for three web workloads.

the goals of this paper are solely to identify concrete performance differences when running representative network services on servers equipped with various network interfaces and to relate those effects back to the micro-level behaviors that shape them.

Figure 1 shows the performance achieved by the thttpd web server for client traces extracted from a university computer science department (CS), the 1998 soccer World Cup tournament (World Cup), and a NASA web site (NASA). The first trace is obscured for anonymous review, while the latter two are available from the Internet Traffic Archive (`http://ita.ee.lbl.gov/`). Figure 2 shows the cumulative distribution of response sizes in these traces. Responses larger than 200 KB, which are less than 3% of the responses in all of the workloads, are not shown in the figure. The largest responses are over 17 MB in CS, almost 7 MB in NASA, and almost 3 MB in World Cup. In Figure 1, the network interfaces are ordered such that performance decreases from left to right. All other figures in this paper will maintain this ordering, making it easy to remember the order of application-level performance of the NICs. The web environment shows substantial performance differences across the NICs, with the fastest NIC consistently achieving 40–60% more throughput than the least effective.

Figure 3 shows the performance achieved by the Click software router for traces from Advanced Network Services (ADV), NASA Ames to MAE-West (AIX), and the University of Memphis (MEM). These traces are from October 30, 2002, and were made available by the National Laboratory for Applied Network Research. Figure 4 shows the cumulative distribution of packet sizes in these traces. This distribution is largely bimodal between small and full-sized packets, with a much smaller fraction of medium-sized pack-

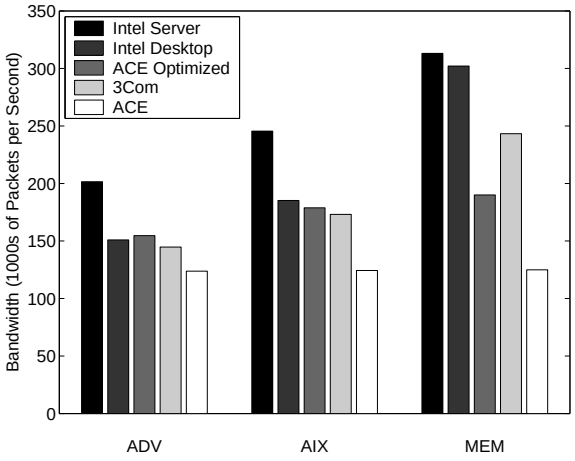

Figure 3: Throughput in thousands of packets per second achieved by the Click software router running on a PC-based server with various network interfaces on three workloads.

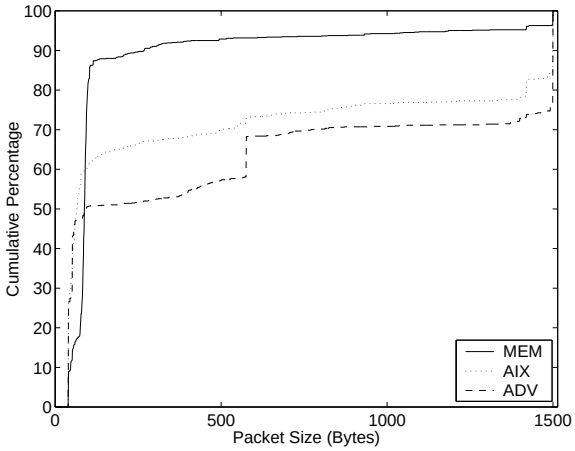

Figure 4: Cumulative distribution of packet sizes in three router traces.

ets. As in the web environment, there is a substantial performance difference across NICs, with the fastest NIC achieving 60–150% higher throughput than the least effective NIC. The driver for the Netgear NIC has some incompatibilities with the Click software, which prevented them from working together.

The figures show that the choice of network interface can have a substantial impact on application-level performance. The following sections explore the micro-level behaviors of NICs that shape system performance.

3 Microbenchmarking NICs

This section proposes a set of benchmarks to study the performance impacts of network interface hardware on real network services. Section 3.1 explains the shortcomings of current networking microbenchmarks. Section 3.2 describes the functionality of real network interfaces and the overheads associated with transferring data through a network interface. Section 3.3 then describes the new microbenchmark suite and how it aims to capture the overheads seen in network interfaces and real applications.

3.1 Current Networking Microbenchmarks

A variety of microbenchmarks exist to capture the behavior of various network protocols and implementations. Of these, some of the most popular are netperf and LMbench [5, 9]. LMbench includes both latency and bandwidth measurements, while netperf only includes bandwidth measurements. The netperf

NIC	Abbreviation	Year	Checksumming	Interrupt Coalescing
Intel Pro 1000 MT Server	Intel Server	2002	TCP/UDP Rx/Tx	131 $u$sec (Tx), 28.6 $u$sec (Rx)
Intel Pro 1000 MT Desktop	Intel Desktop	2002	TCP/UDP Rx/Tx	131 $u$sec (Tx), 28.6 $u$sec (Rx)
3Com 70024 Optimized	ACE Optimized	1997	Disabled (HW bug)	4 msec (Tx) 400 $u$sec (Rx)
Netgear GA622	Netgear	2001	IP/TCP/UDP Rx/Tx	None
3Com 3C996B	3Com	2002	IP/TCP/UDP Tx	150 $u$sec
3Com 70024	ACE	1997	Disabled (HW bug)	2 msec (Tx), 200 $u$sec (Rx)

Table 1: Characteristics of various network interfaces. (Rx and Tx mean receive and transmit, respectively.)

measurements, however, are more flexible. Both systems are portable across various operating systems and architectures; the descriptions below focus on netperf but also apply to LMbench.

The netperf microbenchmarks allow for testing the bandwidth of UDP and TCP streams from a sender to a receiver. The UDP stream test generates datagrams as fast as the operating system can produce them, while the TCP stream test sends data on a single connection that complies with TCP congestion control policies. However, these programs use standard networking APIs (e.g., `read` and `write`) for portability rather than more advanced system calls such as `sendfile` that speed up CPU performance by using zero-copy I/O [11]. Since the UDP test does not throttle its datagram production to any specific rate, the operating system can also encounter overload conditions in which it does extra work to create datagrams only to have them dropped by the device driver or the network interface. This extra wasted work can cause a reduction in network throughput relative to the peak value achieved when the driver and network interface are able to deliver all datagrams successfully. The TCP bandwidth test is limited by its use of only a single connection, causing latency and window-size limitations to influence achieved bandwidth. It is possible to run multiple copies of netperf to create multiple connections, but each such copy requires a separate heavyweight process. In contrast, modern servers use such techniques as fast event notification and non-blocking sockets to allow a single process to manage many connections without the overhead of context switching or process management [2, 12].

As a result of the above limitations, the netperf benchmarks incur more operating system overhead than necessary and do not reflect the various performance optimizations applied to real network server applications. Despite the extra computational work performed by these server applications, their use of such techniques as zero-copy I/O and fast event notification for connection management allow them to achieve higher network throughput than the values reported by netperf.

3.2 Network Interface Hardware

In order to properly benchmark network interfaces, their function must be well understood. Table 1 shows some of the characteristics of the six network adaptors considered in this study and the abbreviations that

will be used to identify them throughout the paper. As shown in Table 1, the network adapters considered in this study are the Intel Pro 1000 MT Server, Intel Pro 1000 MT Desktop, the 3Com 70024 based on the programmable Alteon ACEnic with optimized firmware, the Netgear GA622, the 3Com 3C996B, and the 3Com 70024 using the distributed firmware. The 3Com 70024 entry appears twice because it is a programmable NIC, and two different firmware versions were tested: the firmware included with the FreeBSD driver (ACE) and firmware that is parallelized across the two programmable processors on the NIC and makes better use of on-chip memory (ACE Optimized) [6]. Although these codes run on the same hardware, they are considered separate network interfaces since the firmware performance characteristics vary substantially.

All of the NICs studied have a PCI hardware interface to the host server, use a device driver to interact with the operating system, use direct memory access (DMA) to communicate data between the host memory and the network interface memory, maintain the Ethernet medium access control (MAC) policies (including 802.3x flow control), and have special signal processing hardware to interpret the physical (PHY) data layer of the network. The Intel Desktop NIC has a 32-bit/66 MHz PCI interface, whereas the rest of the NICs have a 64-bit/66 MHz PCI interface. Although the internals of the application-specific integrated circuits in the NICs cannot be easily determined, they all share certain characteristics.

To initiate a send, the operating system invokes a device driver, which informs the network interface of a new Ethernet frame by writing to a memory-mapped NIC register. All the interfaces studied use that write to indicate to the NIC the presence of a DMA descriptor in a well-known location. The NIC first transfers this descriptor by DMA, and then uses the contents of the descriptor to issue a second DMA that transfers the actual frame data into its internal buffers. The frame remains in the NIC's internal buffers until the MAC layer is ready to transmit the frame. When the MAC is ready to transmit the frame, the NIC transfers the data from its internal buffers to the network.

Receives are handled differently from sends because they appear as unsolicited traffic from the network. Consequently, the operating system must have pre-allocated data storage in main memory for receives. The network interface must provide the operating system with a mechanism to inform it of this reserved storage. When data actually arrives, the interface first copies the data into its internal buffers. (The frame cannot be transferred to the host until the frame is known to have a valid Ethernet CRC and the host I/O bus is ready.) The NIC then initiates a DMA of the received data from its internal buffer to the pre-allocated operating system buffer in main memory.

Calculating checksums for the IP, TCP, and UDP layers involves numerous simple computations and

can consume a significant amount of CPU resources. To alleviate the load on the CPU, many modern NICs support checksum offloading for at least some packets [7]. Such checksum calculations are typically done while data is being transferred through DMA. As shown in Table 1, the Intel NICs support TCP and UDP checksumming for both transmitted and received packets, the Netgear NIC performs all checksumming functions, and the 3Com NIC only performs checksumming on transmitted packets. While the ACE NIC does have support for checksum offloading, it is disabled in this study to avoid a hardware bug reported by the manufacturer.

Traditionally, network interfaces would interrupt the host processor after completing a send (to indicate that the state information for that packet may be freed) or after a receive (to indicate that the driver should process the new data). However, most of the interfaces studied in this paper improve performance by interrupting the CPU only after a certain number of frames has arrived or been sent, a certain time has elapsed, or some resource has become exhausted (such as DMA descriptors). Table 1 shows the time thresholds for each of the NICs. As the table shows, these thresholds are set differently on the send and receive paths, since these paths have different needs. The Intel NICs include further optimizations, as they do not automatically interrupt the CPU when the timer expires on send-only workloads. In this case, the NICs wait until resources become close to exhaustion, since the additional latency of notifying the CPU that a packet has been sent has a minimal effect on server performance.

Both send and receive operations have some overheads that scale with the size of the frame (per-byte overheads), including the transfer of frame data over the I/O bus, the transfers of data into and out of the internal buffers of the network interface, and the TCP or UDP checksum computations on the packet data. However, there are also several overheads that are independent of the size of the frame (per-frame overheads), including the driver invocation, the memory-mapped I/O operations, the DMA descriptor transfer, the initiation of the DMA (including I/O bus addressing and request phases), IP checksumming, MAC management, and interrupting the host.

3.3 Proposed Microbenchmarks

Although full-fledged network services exercise the send and receive data paths described in Section 3.2 with various sizes and types of frames, the specific impacts of each data path and type of overhead (per-frame or per-byte) are difficult to isolate because of the large number of data flows with distinct characteristics, the complex interactions of the service code with the operating system, and the overhead of the operating system. This paper proposes and utilizes a new microbenchmark suite specifically aimed at characterizing

the micro-level behaviors of network interfaces that impact system-level performance. The goal of this suite is to isolate the primary micro-level behaviors exhibited by the network services and determine what, if any, relationship exists between that specific behavior and overall system performance. The suite consists of the following tests:

1. UDP send, 1472 byte datagrams: a continuous stream of maximum-sized UDP datagrams is sent from a sender to a receiver. This test isolates the NIC's ability to send large frames. UDP datagrams of 1472 byte results in maximum-sized Ethernet frames of 1518 bytes (including 14 bytes for Ethernet headers, 20 bytes for IP headers, 8 bytes for UDP headers, and 4 bytes for the Ethernet CRC). For every Ethernet frame, there is also an 8 byte preamble and a 12 byte inter-frame gap. Therefore, the maximum theoretical UDP throughput on this test is 957 Mbps ($\frac{1472}{1538} \times 1000$ Mbps).
2. UDP receive, 1472 byte datagrams: this test is the opposite of the UDP send test, and isolates the NIC's ability to receive large frames.
3. UDP 3-way, 1472 byte datagrams: the NIC under test simultaneously sends a stream of maximum-sized UDP datagrams to one host while receiving a stream of maximum-sized UDP datagrams from another host. This test isolates the NIC's ability to exercise its send and receive paths simultaneously. As Gigabit Ethernet is full duplex, the maximum theoretical UDP throughput on this test is 1914 Mbps.
4. UDP send, 18 byte datagrams: a continuous stream of minimum-sized UDP datagrams is sent from a sender to a receiver. This test isolates the NIC's ability to send small frames, which will expose the impact of per-frame overheads. UDP datagrams of 18 bytes result in minimum-sized Ethernet frames of 64 bytes (including Ethernet/IP/UDP headers and Ethernet CRC). Including the preamble and inter-frame gap, the maximum theoretical UDP throughput on this test is 214 Mbps ($\frac{18}{84} \times 1000$ Mbps).
5. UDP receive, 18 byte datagrams: this test is the opposite of the UDP send test, and isolates the NIC's ability to receive small frames.
6. UDP 3-way, 18 byte datagrams: as in the large frame 3-way test, the NIC under test simultaneously sends and receives streams of minimum-sized UDP datagrams. This test isolates the ability of the NIC to exercise its send and receive paths simultaneously when per-frame overheads are dominant. The maximum theoretical UDP throughput of this test is 428 Mbps.
7. TCP send, 1460 byte segments: a continuous stream of maximum-sized TCP segments is sent from a sender to a receiver over several TCP connections subject to TCP's acknowledgment and flow control policies. This test monitors the NIC's ability to handle sending maximum-sized data frames while

receiving minimum-sized acknowledgment frames. This test is not the same as simply combining tests 1 and 5 because the frame rate in the simultaneous streams will be paced according to TCP flow control rules and balanced at one acknowledgment for every second data segment [1]. TCP segments of 1460 bytes result in maximum-sized Ethernet frames of 1518 bytes (including 14 bytes for Ethernet headers, 20 bytes for IP headers, 20 bytes for TCP headers, and 4 bytes for the Ethernet CRC). Therefore, the maximum theoretical TCP throughput on this test is 949 Mbps ($\frac{1460}{1538} \times 1000$ Mbps).

8. TCP receive, 1460 byte segments: this test is the opposite of the TCP send test, and isolates the NIC's ability to receive large frames subject to TCP flow control while also sending acknowledgments.
9. TCP connection acceptance: a continuous stream of requests to open and close TCP connections is sent to the NIC under test. This test isolates the NIC's impact on the system's ability to accept connections.
10. TCP connection initiation: this test is the opposite of the TCP connection acceptance test, and isolates the NIC's impact on the system's ability to initiate connections.

All of the UDP and connection microbenchmarks can be throttled using a `rate` parameter. This eliminates unnecessary overhead in the operating system resulting from unsuccessful transmissions or connection attempts. The TCP send and receive tests support a configurable number of independent connections. Furthermore, all transmissions of UDP and TCP data are performed by a special system call into FreeBSD that bypasses `read` and `write`, instead continually replaying pre-built packets of the appropriate size to the device driver. This system call enables the microbenchmarks to isolate the performance of the NIC from other operating system effects as much as possible. However, the performance of the device driver can still impact the results, since the NIC cannot function without its driver. Each test reads statistics about the network interface from the device driver to determine if all relevant data was actually sent on the Ethernet. The throttling and connection parameters for the maximum achievable throughput are selected through iterative search.

Obvious omissions from the suite include latency tests, TCP tests with small data segments and TCP 3-way tests. Latencies are not tested because the latency of the network interface only impacts the number of TCP connections required to achieve a given level of bandwidth in a network server. A TCP test with small data segments (6 bytes) would be highly subject to TCP implementation details, because TCP congestion control policies only specify that an acknowledgment for a segment should be sent no later than 500 ms afterward or upon receiving two maximum-sized segments worth of data, whichever comes first [1]. An aggressive TCP implementation could thus delay an acknowledgment until receiving $\lceil \frac{2 \times 1460}{6} \rceil = 487$ data

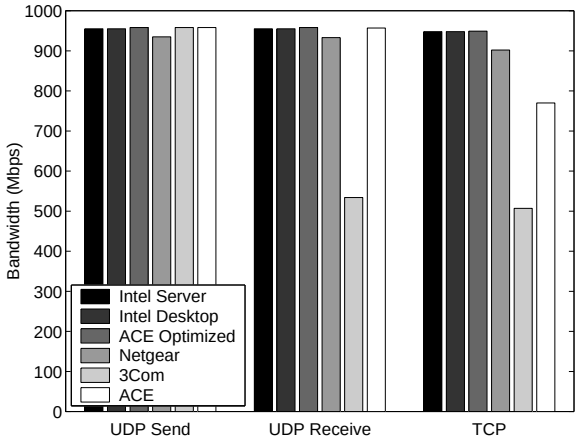

Figure 5: Throughput in Mbps achieved using the microbenchmarks of Section 3.3 with maximum-sized frames (1472 byte UDP datagrams or 1460 byte TCP segments to form 1518 byte Ethernet frames). In each of these tests, the sending and receiving NICs are the same.

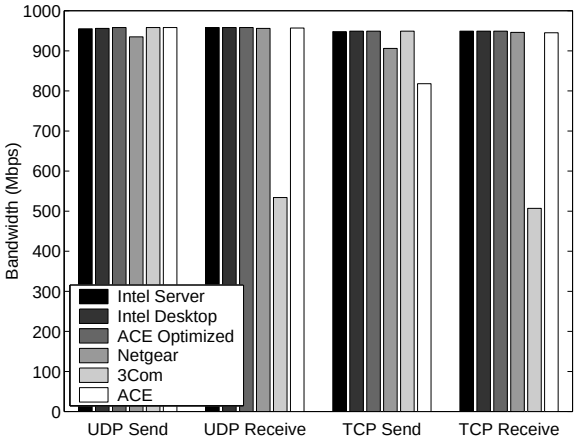

Figure 6: Throughput in Mbps achieved using the microbenchmarks of Section 3.3 with maximum-sized frames (1472 byte UDP datagrams or 1460 byte TCP segments to form 1518 byte Ethernet frames). In all of the send tests, the best receiving NIC, Intel Server, is used. In all of the receive tests, the best sending NIC, 3Com, is used.

segments. In practice, most TCP implementations do not wait so long to acknowledge even small segments. A TCP 3-way test simultaneously sends and receives streams of both large and small segments subject to TCP flow control, making it difficult to isolate any particular performance characteristic.

4 Microbenchmark Results

This section details the results of the proposed microbenchmark suite in gauging the performance of the network interfaces under test. Each microbenchmark from Section 3.3 is run using each of the six network interfaces that were considered in Section 2.3. In most cases, the microbenchmark is first run with the NIC under test being used both to send and receive frames. Then, each NIC is tested as a sender by sending frames to the best receiver and as a receiver by receiving frames from the best sender. This enables send and receive performance to be isolated for each NIC.

4.1 Maximum-sized Frames

Figure 5 shows the performance of various network interfaces in the data transmission tests which send and receive maximum-sized Ethernet frames. Each test involves two machines, and both systems are configured identically to the client machines described in Section 2.3, with the exception that in each test the sender and

receiver both use the NIC being tested. For example, the leftmost bar shows the result of running the UDP Send microbenchmark, where the sending host uses an Intel Server NIC and the receiving host also uses an Intel Server NIC. The three groups of bars show the achieved bandwidth of each of the six network interfaces on the UDP Send (test 1), UDP Receive (test 2), and TCP (tests 7 and 8) microbenchmarks, respectively, from Section 3.3. These tests show that all of the network interfaces can achieve near maximum throughput when sending large UDP datagrams. However, the throughput of the 3Com network interface drops substantially on the UDP receive path. Detailed analysis shows that this drop comes from a workaround in the driver, which limits the maximum length of a DMA transfer in order to avoid tripping a bug in the TCP checksumming features of this NIC. TCP throughput is near the maximum for both Intel network interfaces and the optimized ACE, slightly lower for the Netgear, and substantially lower for 3Com and the unoptimized ACE. The 3Com suffers from the poor receive path of the peer because TCP is a reliable protocol. The unoptimized ACE suffers because its firmware shares a single programmable processor between send and receive (e.g., ACK) processing; this is in contrast to the parallelized firmware of the optimized ACE.

Figure 6 shows the send and receive performance of each NIC when paired with the best receiving and sending NICs, respectively. For the UDP and TCP send test, the Intel Server NIC is used as a receiver. Despite UDP's unreliability, this configuration can impact UDP send because of the 802.3x flow control mechanisms supported at the MAC layer by these network interfaces. However, our results show little impact on these interfaces. The TCP performance improves for 3Com (since this interface had a fine send path and a poor receive path) and for ACE (by reducing the latency of the receiver and consequently of ACKs). For the UDP and TCP receive test, the 3Com NIC is used as a sender. The Netgear and unoptimized ACE now approach the theoretical maximum throughput, indicating that their results in Figure 5 were hindered by their send-sides and that these NICs may be more heavily optimized for receive processing.

4.2 Minimum-sized Frames

Figure 7 shows the UDP data throughput for minimum-sized frames (18 byte datagrams) between identical NICs (tests 4 and 5). The achieved throughput on these benchmarks are not only dramatically lower than in Figure 5, but they are also significantly lower than the maximum possible utilization for Ethernet (214 Mbps). These results indicate that these NICs are not designed for high performance on small frames and that per-frame overheads substantially limit performance. The two Intel NICs and the 3Com NIC perform best on small frames for the six NICs tested, with the Intel Server NIC most efficient at sending and the Intel Desktop NIC most efficient at receiving.

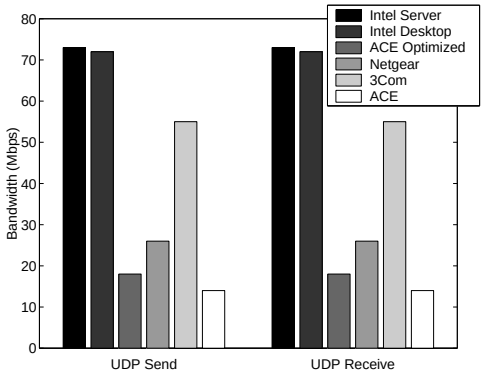

Figure 7: UDP throughput in Mbps achieved using microbenchmarks of Section 3.3 with minimum-sized frames (18 byte UDP datagrams to form 64 byte Ethernet frames). In each of these tests, the sending and receiving NICs are the same.

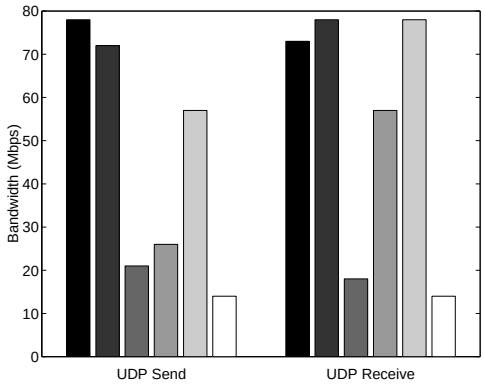

Figure 8: UDP throughput in Mbps achieved using the microbenchmarks of Section 3.3 with minimum-sized frames (18 byte UDP datagrams to form 64 byte Ethernet frames). In all of the send tests, the best receiving NIC, Intel Desktop, is used. In all of the receive tests, the best sending NIC, Intel Server, is used.

Figure 8 shows the send and receive performance of each NIC when paired with the most efficient receiving and sending NICs, respectively. When paired with the Intel Desktop as receiver, only the Intel Server NIC improves performance, implying that it is slightly more optimized for sending small frames than receiving them. When paired with the Intel Server NIC as sender, the Intel Desktop, Netgear, and 3Com NICs all achieve higher receive throughput, suggesting that all of those NICs are better suited for receiving small frames than sending them. The ACE and ACE Optimized NICs show negligible improvement when paired with the best receiving or sending NICs, and they are also the worst performers at sending and receiving small frames.

4.3 Simultaneous Send and Receive Traffic

Figures 9 and 10 show the bidirectional throughput of each NIC using the 3-way UDP microbenchmarks (tests 3 and 6). Figure 9 shows the performance of each NIC when maximum-sized UDP datagrams are being sent to it by the best sender, the 3Com NIC, and it is sending maximum-sized UDP datagrams to the best receiver, the Intel Server NIC. These results show a clear trend among the NICs, as performance degrades from the left to the right in the figure. Recall that the NICs are organized from left to right in order of their application-level performance. Only the Intel Server NIC is able to achieve nearly the sum of its individual send and receive bandwidth. The other network interfaces most likely have some limited resource shared between the send and receive paths; possible limitations include PCI bandwidth (a likely problem

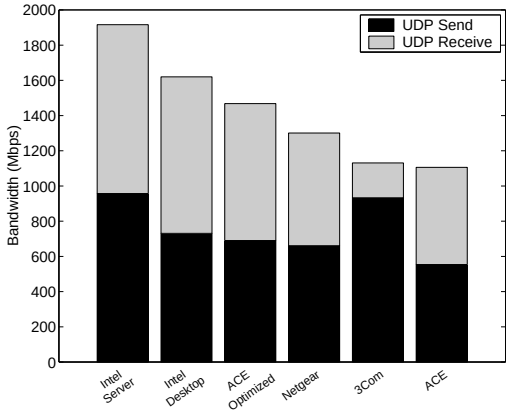

Figure 9: UDP Throughput in Mbps achieved for the UDP 3-way test for maximum-sized frames. In all cases the best sending NIC, 3Com, sends frames to the NIC being tested at the same time that the NIC being tested sends frames to the best receiving NIC, Intel Server.

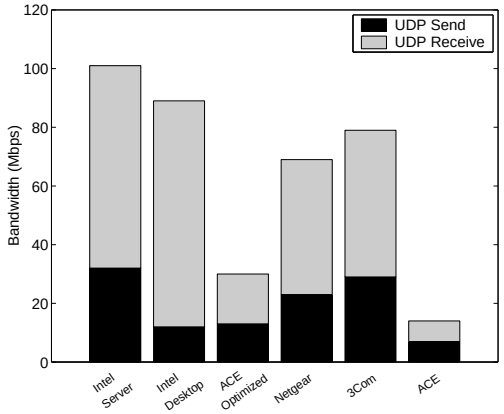

Figure 10: UDP Throughput in Mbps achieved for the UDP 3-way test for minimum-sized frames. In all cases the best sending NIC, Intel Server, sends frames to the NIC being tested at the same time that the NIC being tested sends frames to the best receiving NIC, Intel Desktop.

for the Intel Desktop NIC, which only has a 32-bit PCI interface with a theoretical maximum of 2 Gbps), a shared programmable processor (in unoptimized ACE), or on-board memory bandwidth (since each bit of network traffic is touched twice as described in Section 3.2, the card must provide at least twice as much memory bandwidth as full-duplex network bandwidth).

Figure 10 shows the performance of each NIC when minimum-sized UDP datagrams are being sent to it by the best sender, the Intel Server NIC, and it is sending minimum-sized UDP datagrams to the best receiver, the Intel Desktop NIC. In this test, no NIC is able to approach the sum of its individual send and receive bandwidth for minimum-sized datagrams. Each NIC achieves a slight improvement over its individual send or receive throughput except for ACE. ACE achieves the same throughput as in either send or receive because both paths share the same processor, which is saturated because of per-frame overheads.

4.4 Connection Establishment

Figure 11 shows the rate of TCP connection establishment when connecting between two NICs of the same type, when accepting connections (test 9) from the best NIC for connection initiation (3Com), and when initiating connections (test 10) to the the best NIC for connection acceptance (Intel Desktop). TCP connection establishment generates bidirectional traffic of small frames. The trends in Figure 11 are therefore similar to the trends in Figure 10. The Intel NICs are the best connection acceptors and the Netgear and 3Com NICs are the best connection initiators. The two ACE NICs perform poorly at both connection initiation and

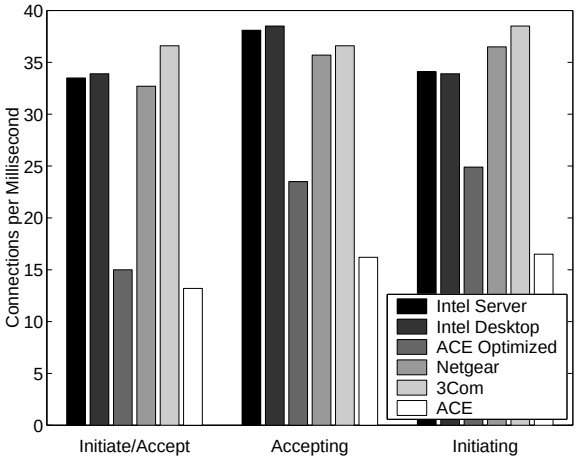

Figure 11: Number of TCP connections per millisecond achieved. In the Initiate/Accept tests both NICs are the same. In the Accepting tests, the initiating NIC is the best initiator, the 3Com NIC. In the Initiating tests, the accepting NIC is the best accepter, the Intel Desktop NIC.

acceptance, as is to be expected from their poor performance when dealing with small frames.

4.5 Summary

The results of the microbenchmark tests on the six network interfaces show three significant behaviors. First, all of the NICs achieve near wire-speed on the large frame UDP send test, and all but one achieve near wire-speed on the large frame UDP receive test. These NICs are thus able to handle per-byte overheads at wire-speed, and maximum-sized Ethernet frames are sufficient to amortize the per-frame overheads.

Second, none of the NICs tested can achieve even 50% of the theoretical peak on the small frame send and receive tests. While these NICs can handle the per-byte overheads of even higher data rates, the per-frame overheads become overwhelming at the increased frame rate demanded by small frames. The ability of a NIC to handle small frames is a potential differentiator, as the fastest NIC achieves five times the throughput of the slowest.

The third key behavior is the bidirectional performance of the six network interfaces. With maximum-sized frames, only the Intel Server NIC can achieve a throughput that is close to the sum of its individual send and receive throughput; this NIC also achieves 73% more bidirectional throughput than the lowest performing interface. This result indicates that either this NIC has completely separate resources for the send and receive paths or that its shared resources are provisioned to handle wire-speed in both directions, whereas the rest of the NICs suffer from some sort of resource contention. Finally, none of the network interfaces are able to achieve a throughput that is close to the sum of its individual send and receive throughputs

Correlation Coefficient		UDP, max			UDP, min			TCP			
		Send	Recv	3-way	Send	Recv	3-way	Send	Recv	Conn. Acc.	Conn. Init.
Web	CS	0.07	0.32	0.85	0.56	0.35	0.53	0.82	0.32	0.52	0.38
	NASA	0.08	0.29	0.84	0.55	0.35	0.52	0.83	0.30	0.52	0.38
	WC	-0.12	0.56	0.94	0.63	0.44	0.63	0.55	0.56	0.57	0.34
Router	ADV	-0.81	0.20	0.90	0.68	0.48	0.70	0.61	0.21	0.61	0.51
	AIX	-0.82	0.11	0.89	0.78	0.61	0.80	0.73	0.11	0.74	0.65
	MEM	-0.83	-0.06	0.76	0.97	0.89	0.97	0.78	-0.05	0.96	0.85

Table 2: Correlation coefficients between the throughputs reported by the microbenchmarks used for testing network interfaces and the high-level network services of Section 2. The "max" and "min" indicate the use of maximum-sized or minimum-sized frames.

on minimum-sized frames. This behavior stems from both resource contention and per-frame overheads. Furthermore, this test shows that the Intel Server NIC does indeed share resources between the send and receive paths, and these shared resources become saturated with the per-frame overheads of small frames. Even so, the Intel Server NIC achieves a bidirectional throughput seven times higher than the slowest NIC. The connection establishment tests are similar to the bidirectional tests with small frames, since these tests have the same types of network traffic.

5 Discussion

As described in Section 3, this microbenchmark suite aims to capture networking behaviors that appear in real network services and that have performance impacts that vary with the network interface. Some microbenchmarks clearly have no impact on system-level performance. For example, for UDP send of maximum-sized frames, there is less than 3% difference between the slowest and fastest network interface, so this behavior cannot possibly account for the differences of up to 150% in system performance. Since this study only considers six different network interfaces, there is not sufficient data to establish statistically strong correlations between the microbenchmarks and system-level performance for the higher-level services. Nevertheless, statistical methods are useful for providing insight into the relationship between the microbenchmarks and system-level throughput. Table 2 reports the correlation coefficient between each microbenchmark and the system-level throughput for each workload, calculated using MATLAB's `corrcoef`. Entries near 0 suggest that the microbenchmark listed in that row of the table is of little value as a performance predictor for the service and workload specified in that column, while entries near ± 1 suggest that the microbenchmark is a valuable performance predictor.

As the table shows, the performance of the web server has a close relationship to the performance of the NIC on the TCP send test and on the UDP 3-way test with maximum-sized frames. The similarity

to the TCP send test is intuitive, as the major function of a web server is to send responses over TCP connections. Figure 2 shows that NASA and CS have over 70% of their HTTP responses greater than the size of a single maximum-sized segment, while less than 50% of the World Cup data responses exceed a single segment. As a result, the World Cup workload has fewer sequences of maximum-sized frames between HTTP requests and thus has less correlation to TCP send. The similarity to the UDP 3-way test with maximum-sized frames follows the observation that the benchmark results of Figure 9 degrade from left to right, just as the NICs are arranged by application-level performance. This relationship indicates that the resource contention of simultaneous send and receive impacts performance even for web servers that have their data bitrates dominated by send. Furthermore, the performance of the NIC on the UDP send and receive tests with minimum-sized frames corresponds more closely to application performance than the performance of the NIC on the UDP send and receive tests with maximum-sized frames. This indicates that the performance of unidirectional streams of small frames, such as acknowledgments, is an important element of application performance. Finally, the TCP connection tests are not nearly as matched to web server performance as would be expected. Ultimately, this is because the NICs are able to handle almost ten times the connection rate required for the web application. Only the World Cup web workload sees a noticeable correlation with the UDP receive of maximal frames; this workload has smaller response sizes than the other tests and thus sees a greater fraction of data traffic from HTTP requests. This result suggests that HTTP requests may behave more similarly to large frames than small frames.

Table 2 also shows that the performance of the software router is most closely related to the performance of the NIC on the UDP 3-way send tests with both minimum-sized and maximum-sized frames. Again, the similarity of these tests to the application is intuitive, as the router must be able to handle bidirectional streams of all frame sizes. The UDP send test with maximum-sized frames appears to have a strong negative correlation with the software router, but this is simply a numerical fluke; all of the NICs are within 0.5% of each other and their variations coincidentally correspond negatively to the differences in the software router. The software router's performance is also reasonably matched with the NIC's performance on the UDP send and receive tests with minimum-sized frames. This reiterates the point that the per-frame overhead of minimum-sized frames is difficult to handle at high frame rates and can have a profound impact on performance. Finally, the TCP connection tests are surprisingly similar to the performance of the software router, despite the fact that the router does not initiate or receive any connections. However, these tests consist of small connection-establishment and shutdown frames, similar to the bidirectional small frame traffic of the software router.

Overall, the results of the microbenchmark tests show that the micro-level behaviors that are most relevant to the performance of both web servers and software routers are the 3-way tests, with both maximum-sized and minimum-sized frames having significant impact. These results indicate that resource contention between different flows in a router or among HTTP responses, HTTP requests, and acknowledgments in a web server has a first-order impact on system-level performance. Additionally, the ability of the network interface to send or receive small frames is of much greater significance in shaping its comparative system-level performance than its ability to send or receive large frames.

6 Conclusions

The network interface card used in a network server can dramatically impact its application-level performance. Simply changing the Gigabit Ethernet NIC in a network server can increase the throughput of a web server by as much as 60% and increase the throughput of a software router by as much as 150%. This paper promotes microbenchmarking of network interfaces as a tool to isolate the low-level behaviors that shape their impact on application-level performance.

A suite of ten lightweight microbenchmarks isolates key aspects of network interface performance that relate well to application-level performance. These benchmarks have minimal dependence on the operating system, allowing the device driver and NIC itself to be tested. The benchmarks isolate the ability of a network interface to send and receive small and large UDP datagrams, to handle bidirectional UDP traffic, and to handle TCP flows and connections. The microbenchmark results show that all NICs tested can achieve near wire-speed in sending large frames, but that the performance of NICs varies greatly when processing bidirectional streams of large frames (up to 73% performance difference between NICs), bidirectional streams of small frames (up to a factor of seven performance difference), or unidirectional streams of small frames (up to a factor of five performance difference). Although each test has too few data points (only six interfaces) to provide a strong statistical correlation, the throughputs achieved by both the web server and the software router correspond closely to the microbenchmarks related to handling bidirectional flows and small frames.

The microbenchmark results indicate directions for profitable future investigation. Some studies and experimental implementations have proposed offloading TCP segmentation of large data regions to the network interface to reduce CPU utilization [3, 10]. The results of Section 4.1 indicate that given sufficient CPU resources, the existing TCP stack can already saturate the Gigabit Ethernet links of most network interfaces for large frames. Since CPU speeds continue to increase faster than all other system components, it does

not seem profitable to focus solely on reducing the CPU utilization required to send large frames. Instead, new techniques should focus on offloading processing that will enable more efficient management of small frames and simultaneous send and receive streams. Techniques to reduce the per-frame overheads of communication between the host and the network interface (e.g., driver invocation, interrupts, memory-mapped I/O, and DMA initiation) will likely yield greater benefits for small frames. Techniques to reduce resource contention such as parallelizing the firmware of a programmable network interface (as in ACE Optimized) or carefully provisioning memory bandwidth on the NIC will likely yield greater benefits for bidirectional flows.

Although this work correlates NIC performance back to applications, it does not directly integrate NIC performance into a formal system-performance vector akin to those previously described by Saavedra and Smith or Seltzer et al. [15, 16]. These vector-based methodologies have been used to analyze scientific or operating-system-intensive codes in terms of microbenchmark-like segments. Although complicated by interactions with operating system and network stack implementation details, integration of NIC performance into such a vector-based methodology should enable faster analysis and performance correlation for other network-based workloads.

References

[1] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. IETF RFC 2581, April 1999.

[2] G. Banga, J. C. Mogul, and P. Druschel. A scalable and explicit event delivery mechanism for UNIX. In *Proceedings of the USENIX 1999 Annual Technical Conference*, pages 253–265, June 1999.

[3] H. Bilic, Y. Birk, I. Chirashnya, and Z. Machulsky. Deferred Segmentation for Wire-Speed Transmission of Large TCP Frames over Standard GbE Networks. In *Hot Interconnects IX*, pages 81–85, August 2001.

[4] A. B. Brown and M. I. Seltzer. Operating System Benchmarking in the Wake of *lmbench*: A Case Study of the Performance of NetBSD on the Intel x86 Architecture. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 214–224, June 1997.

[5] Information Networks Division, Hewlett-Packard Company. *Netperf: A Network Performance Benchmark*, February 1995. Revision 2.0.

[6] H. Kim, V. S. Pai, and S. Rixner. Exploiting Task-level Concurrency in a Programmable Network Interface. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 61–72, June 2003.

[7] K. Kleinpaste, P. Steenkiste, and B. Zill. Software Support for Outboard Buffering and Checksumming. In *Proceedings of the ACM SIGCOMM '95 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 87–98, August 1995.

[8] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[9] L. McVoy and C. Staelin. lmbench: Portable Tools for Performance Analysis. In *Proceedings of the 1996 USENIX Technical Conference*, pages 279–295, January 1996.

[10] Microsoft Corporation. Windows Network Task Offload. In *Microsoft Windows Platform Development*, December 2001.

[11] E. M. Nahum, T. Barzilai, and D. Kandlur. Performance Issues in WWW Servers. *IEEE/ACM Transactions on Networking*, 10(2):2–11, February 2002.

[12] V. S. Pai, P. Druschel, and W. Zwaenepoel. Flash: An Efficient and Portable Web Server. In *Proceedings of the USENIX 1999 Annual Technical Conference*, pages 199–212, June 1999.

[13] V. S. Pai, P. Druschel, and W. Zwaenepoel. I/O-Lite: A Unified I/O Buffering and Caching System. In *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation*, pages 15–28, February 1999.

[14] J. Poskanzer. *thttpd - tiny/turbo/throttling HTTP server*. Acme Labs, February 2000. Unix manual page.

[15] R. H. Saavedra and A. J. Smith. Analysis of Benchmark Characteristics and Benchmark Performance Prediction. *ACM Transactions on Computer Systems*, 14(4):344–384, November 1996.

[16] M. Seltzer, D. Krinsky, K. Smith, and X. Zhang. The Case for Application-Specific Benchmarking. In *Proceedings of the 1999 Workshop on Hot Topics in Operating Systems (HotOS VII)*, pages 102–107, March 1999.