An Evaluation of an Attempt at Offloading TCP/IP Protocol Processing onto an i960RN-based iNIC

Boon S. Ang
Computer Systems and Technology Laboratory
HP Laboratories Palo Alto

HPL-2001-8
January 9th , 2001*

TCP/IP networking, Intelligent Network Interface

This report presents an evaluation of a TCP/IP offload implementation that utilizes a 100BaseT intelligent Network Interface Card (iNIC) equipped with a 100 MHz i960RN processor. The entire FreeBSD-derived networking stack from socket downward is implemented on the iNIC with the goal of reducing host processor workload. For large messages that result in MTU packets, the offload implementation can sustain wire-speed on receive but only about 80% of wire-speed on transmit. Utilizing hardware-based profiling of TTCP benchmark runs, our evaluation pieced together a comprehensive picture of transmit behavior on the iNIC. Our first surprise was the number of i960RN processor cycles consumed in transmitting large messages--around 17 thousand processor cycles per 1.5kbyte (Ethernet MTU) packet. Further investigation reveals that this high cost is due to a combination of i960RN architectural shortcomings, poor buffering strategy in the TCP/IP code running on the iNIC, and limitations imposed by the I20-based host-iNIC interface. We also found room for improvements in the implementation of the socket buffer data-structure. This report presents profiling statistics, as well as code-path analysis that back up these conclusions. Our results call into question the hypothesis that a specialized networking software environment coupled with cheap embedded processors is a cost effective way of improving system performance. At least in the case of the offload implementation on the i960RN-based iNIC, neither was the performance adequate nor the system cheap. This conclusion, however, does not imply that offload is a bad idea. In fact, measurements we made with Alacritech's SLIC NIC, which partially offloads TCP/IP protocol processing to an ASIC, suggests that offloading can confer advantages in a cost effective way. Taking the right implementation approach is critical.

1 Introduction 3
1.1 Summary of Results 3
1.2 Organization of this Report 5
2 TCP/IP Offload Implementation Overview 6
2.1 Cyclone PCI-981 iNIC hardware 6
2.2 Offload Software 7
2.2.1 Execution Environment 7
2.2.2 Networking code 8
3 iNIC Side Behavior 11
3.1 General Performance Statistics 12
3.2 Hardware bottleneck 14
3.3 tcp_output cost breakdown 17
3.4 Buffering… and the resulting copy 18
3.4.1 Buffer usage for large message transmit 20
3.4.2 Buffer usage for small message transmit 21
3.5 Host-iNIC interface behavior 21
3.6 Miscellaneous Inefficiencies 23
4 Host-side Behavior 24
5 Related Work 26
5.1 Studies on TCP/IP implementation issues 26
5.2 Other networking protocol offload work 29
6 Conclusion 30
6.1 Limitations of this Study 30
6.2 Summary of Results from this Evalution 30
6.3 Implications of our Findings 31
6.4 Recommendation for Future Work 32
Bibliography 33

1 Introduction

This report presents an evaluation of a TCP/IP implementation that performs network protocol stack processing on a 100BaseT intelligent network interface card (iNIC) equiped with an i960RN embedded processor. Offloading TCP/IP protocol processing from the host processors to a specialized environment was proposed as a means to reduce the workload on the host processors. The initial arguments profered were that network protocol processing is consuming an increasingly larger portion of processor cycles and that a specialized software envrionment on an iNIC can perform the same task more efficiently using cheaper processors.

Since the inception of the project, alternate motivations for offloading protocol stack processing to an iNIC have been proposed. One was the iNIC offers a point of network traffic control independent of the host -- a useful capability in the Distributed Service Utility (DSU) architecture [9] for Internet data-centers, where the host system is not necessarily trusted. More generally, the iNIC is viewed as a point where additional specialized functions, such as firewall and web caching, can be added in a way that scales performance with the number of iNIC's in a system.

The primary motivation of this evaluation is to understand the behavior of a specific TCP/IP offload design implemented in the Platform System Software department of HP Laboratories' Computer Systems and Technology Laboratory. Despite initial optimism, this implementation using Cyclone's PCI-981 iNIC, while able to reduce host processor cycles spent on networking, is unable to deliver the same networking performance as Windows NT's native protocol stack for 100BaseT Ethernet. Furthermore, transmit performance lags behind receive performance for reasons that were not well understood.[1]

Another goal of this work is to arrive at a good understanding of the processing requirements, implementation issues and hardware and software architectural needs of TCP/IP processing. This understanding will feed into future iNIC projects targetting very high bandwidth networking in highly distributed data center architectures. At a higher level, information from this project provides concrete data-points for understanding the merits, if any, of offloading TCP/IP processing from the host processors to an iNIC.

1.1 Summary of Results

Utilizing hardware-assisted profiling of TTCP benchmark runs, our evaluation pieced together a comprehensive picture of transmit behavior on the iNIC. All our measurements assume that checksum computation, an expensive operation on generic microprocessors, is done by specialized hardware in Ethernet MAC/Phy devices, as is the case with commodity devices appearing in late 2000.[2]

[1] The team that implemented the TCP/IP offload recently informed us that they found some software problem that was causing the transmit and receive performance disparity and had worked around it. Unfortunately, we were unable to obtain further detail in time for inclusion in this report.

[2] The Cyclone PCI-981 iNIC uses Ethernet devices that do not have checksum support. To factor out the cost of checksum computation, our offload implementation simply does not compute checksum on both transmit and receive during our benchmark runs. To accommodate this, machines receiving packets from

Our first surprise was the number of i960RN processor cycles consumed in transmitting large messages over TCP/IP -- around 17 thousand processor cycles per 1.5kbyte (Ethernet MTU) packet. This cost increases very significantly for smaller messages because aggregation (Nagle Algorithm) is done on the iNIC, thus incurring the overhead of handshake between host processor and iNIC for every message. In an extreme case, with host software sending 1-byte messages that are aggregated into approximately 940byte packets[3], each packet consumes 4.5 million i960RN processor cycles. Even transmitting a pure acknowledgement packet costs over 10 thousand processor cycles.

Further investigation reveals that this high cost is due to a combination of i960RN architectural shortcomings, poor buffering strategy in the TCP/IP code running on the iNIC, and limitations imposed by the I2O-based host-iNIC interface. We also found room for improvements in the implementation of the socket buffer data-structure and some inefficiency due to the gcc960 compiler.

Our study of the host-side behavior is done at a coarser level. Using NT's Performance Monitor tool, we quantified the processor utilization and the number of interrupts during each TTCP benchmark run. We compare these metrics for our offload implementation with those of NT's native TCP/IP networking code and another partially offloaded iNIC implementation from Alacritech. To deal with the fact that different implementations achieve different networking bandwidth, the metrics are accumulated over the course of complete runs transferring the same amount of data.

The measurements show that compared against native NT implementation, our offload implementation achieves significantly lower processor utilization for large messages, but much higher processor utilization for small messages, with crossover point at around 1000byte messages. The interrupt statistics shows similar trend, though with crossover at a smaller message size. Furthermore, the number of interrupts is reduced by a much more significant percentage than the reduction in host processor utilization, suggesting that costs other than interrupt processing contributes quite significantly to the remaining host-side cost in our offload implementation. Based on other researcher's results [1], we believe that host processor copying data between user and system buffer is the major remaining cost.

The Alacritech NIC is an interesting comparison because it represents a very lean and low cost approach. Whereas the Cyclone board is a full-length PCI card that is essentially a complete computer system decked with a processor, memory and supporting logic chips, the Alacritech NIC looks just like another normal NIC card, except that its MAC/Phy ASIC has additional logic to process TCP/IP protocol for "fast path" cases. A

---

our offload implementation run specially doctored TCP/IP stacks that do not verify checksum. Error rates on today's networking hardware in a local switched network are low enough that running TTCP benchmark is not a problem.

[3] The aggregation is not controlled by a fixed size threshold, and thus the actual packet size varies dynamically, subjected to an MTU of 1460 data bytes.

limitation of this approach is it does not allow any modification or additions to the offloaded functions once the hardware is designed.

Our measurement shows that an Alacritech NIC is able to sustain network bandwidth comparable to that of Native NT for large messages, which is close to wire-speed. Its accumulated host processor utilization, while lower than native NT's, is higher than that with our offload implementation. Its performance degrades when messages are smaller than 2k bytes because it has no means of aggregating out-going messages (i.e. no Nagel Algorithm).

This study calls into question the hypothesis that a specialized software environment together with a cheap embedded processor can effectively offload TCP/IP protocol stack processing. The i960RN-based implementation studied in this work is unable to adequately handle traffic even at the 100Mbit/s level, much less at 1 Gigabit/s or 10 Gigabit/s levels that are the bandwidths of interest in the near future. While bad buffering strategy is partly responsible for the less than satisfactory performance on our offload implementation, its BSD-derived TCP/IP protocol stack is in fact better than NT's when executed on the same host platform. Clearly, the "better" stack and the advantage from the specialized interrupt-handling environment are insufficient to make up for the loss of a good processor and the additional overhead of interactions between the host and the iNIC. Ultimately, this approach is at best moving work from one place to another without conferring any advantage in efficiency, performance or price, and at worse, a performance limiter.

This conclusion does not imply that offload is a bad idea. In fact, the performance of the Alacritech NIC suggests that it can confer advantages. What is critical is taking the right implementation approach. We believe there is still unfinished research in this area, that a fixed hardware implementation, such as that from Alacritech, is not the solution. From a functional perspective, having a flexible and extensible iNIC implementation not only enables tracking changes to protocol standards, but also allows additional functions to be added over time. The key is a cost effective, programmable iNIC micro-architecture that pays attention to the interface between iNIC and host. There are a number of promising alternate micro-architecture components, ranging from specialized queue management hardware, to field programmable hardware, to multi-threaded and/or multi-core, possibly Systolic-like, processors. This is the research topic of our next project.

1.2 Organization of this Report

The next section gives an overview of our i960RN-based TCP/IP offload implementation. We briefly cover both the hardware and the software aspects of this implementation to pave the background for the rest of this report. Section 3 examines the behavior on the iNIC. Detailed profiling statistics giving breakdowns for various steps of the processing, and utilization of the hardware resources is presented. This is followed in Section 4 with an examination of the host side statistics. Section 5 presents some related work covering both previous studies of TCP/IP implementations and other offload implementations. Finally, we conclude in Section 6 with what we learned from this study and areas for future work.

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.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.