10Gb/s Ethernet Performance and Retrospective

Technical Report: SF-100602-TC (2) *

Steven Pope and David Riddoch
Solarflare Communications Inc.

7 December 2006

1 Abstract

This paper presents both a retrospective of the development of network interface architecture, and performance and conformance data from a range of contemporary devices sporting various performance enhancing technologies. The data shows that 10Gb/s networking is now possible without statefull offload and while consuming less than one CPU core on a contemporary commodity server.

2 Introduction

From ARPANET [6] and Ethernet [31] in the mid-70s, through Local Area Networking [42], distributed computing [2], and TCP/IP in the early-80s, significant maturity in the design and philosophy of Internetworking [9] was reached by 1988. Implementations of TCP/IP on mainframe and supercomputers in the mid-80s were reported together with some of the architectural tradeoffs of the time. For example, Kline describes [27] the implementation of a library-level protocol stack and Brandriff describes [4] a host based (rather than front-end processor) implementation.

The advent of workstation class computers [37] in the late-80s led to significant developments in the design of network interface hardware, particularly following early experiences with ATM networks [28] and the intent to handle the challenging QoS requirements of multimedia applications. Early ATM network interfaces made use of Programmed IO (PIO)for data transfer. This movement of data proved difficult for the RISC CPUs of the time, and DMA techniques came to be used. For example, Davie describes [12] the host interface used in the AURORA ATM testbed, and Smith and Traw describe [40] DMA double buffering techniques and interrupt moderation. The per-byte overheads of networking were understood and mitigations such as checksum offload were in use [34, 22]. These developments to the mid-90s have come to represent the roots of today's main-stream LAN interface designs. However, while is the case that offloads are now regarded as commodity items, the desirability and utility of even the most simple offload should remain under debate. For example Stone and Partridge [36] describe a study of the root cause of network errors which escape the Ethernet FCS check and find many systematic errors in hardware and software.

One architecture which received attention from the late-80s was that of executing the transport protocol on the host network interface. Implementations include the XTP Protocol Engine [8], the Nectar communications processor [14], and the VMP adaptor [24]. Follow on work [23, 34] based on the Protocol Engine architecture implemented TCP/IP on the host interface for a 622Mb/s ATM network. This TCP/IP offload architecture was rejected [10, 25] and has not subsequently been taken up to any significant degree by the academic community except recently as a means to an end for the support of Remote Direct Memory Access (RDMA)protocols [32].

Another architectural choice which is periodically revisited is whether to perform protocol processing in user or kernel space. Druschel and Davie implemented [13] an interface which allowed user-space programs direct access to an ATM adaptor and Thekkath describes [38] an implementation of a user level TCP/IP stack over Mach. As well as performance, this work was concerned with the issues of multi-protocol coexistence and efficient operation in a micro-kernel envi-

* Issue (1) of this paper has been submitted for publication

ronment. By contrast, the Jetstream/Afterburner adaptor [16] was used to implement TCP/IP in user space [15] over a monolithic kernel, as did Pratt using a firmware modified Gigabit Ethernet NIC [33]. These were essentially ports of their respective kernel protocol stacks to user level over a protected hardware interface. Hybrid models have also been proposed, see [29] for a survey and perspective. However achieving good all-round performance has proven to be elusive without a protocol stack implementation which has been expressly designed for user level operation.

Over the same late-80s to mid-90s period, there was considerable parallel activity with multicomputer network interface architecture. The ATOMIC project[11] utilised components from the Mosaic multicomputer to construct a Gb/s LAN (later commercialised [3]). The multicomputer environment was somewhat less constrained than that of the ATM or distributed systems environments, application behaviour suited low-overhead user-level abstractions of communication, and these abstractions were used by host adaptors in conjunction with high-performance network techniques such as cut-through [26] and source based [7] routing. Portability for scientific application codes running on this architecture was largely resolved by the MPI specification [30]. Multicomputer and LAN convergence was proposed in the early-90s [21, 20] and reports of large-scale deployments [39] of multicomputer interconnects as a LAN were made by 1997. However the availability of commodity 100Mb/s and 1Gb/s Ethernet meant that a bigger movement formed around the use of LAN interconnects in a multicomputer environment [35, 41].

Continual software, protocol, and chipset performance improvements over the 90s meant that achievable throughputs on commodity hardware grew from around 130Mb/s in 1996 to Gb/s by 2001 [18]. This impressive performance increase contributed to the resistance met by industry as it attempted to convert mid-90s work on user-accessible network interfaces [17, 5] into the Infiniband general-purpose converged interconnect. This performance trend has continued over the course of the introduction of 10Gb/s Ethernet. In 2003 Feng reported [19] unidirectional TCP/IP/Ethernet throughputs of 4Gb/s on commodity and 7Gb/s on next-generation hardware, and by 2006 10Gb/s line rate has become possible on a single core of commodity server chipsets.

The remainder of this paper offers a set of comparative micro-benchmark data for some generally available contemporary 10Gb/s Ethernet NICs. It is hoped that this data will be a useful calibration point for the community, particularly at a time when industry is again debating network interface architecture.

3 Offload Taxonomy

The basic act of transmitting and receiving data from a network interface to an operating system is defined here as regular networking. A *regular network interface* performs no processing of the packets above the link layer protocol. For example, an Ethernet interface may process the Ethernet FCS, or perform multicast filtering, but it does not process the IP headers within the frame. Nevertheless, a regular adaptor can be a highly tuned device which is capable of efficient DMA to and from a host, tracking large numbers of transmit and receive descriptors (and their associated buffers), and balancing the tradeoffs associated with interrupting the main CPU in the system.

If the adaptor is capable of providing optimisations based on the local state contained within the upper layer protocols embedded within a single frame, then an adaptor is defined to be a *stateless offload adaptor*. There are a number of stateless offloads which can be performed based on higher level protocols. For example TCP/IP checksum calculation and verification, and TCP Segmentation Offload (TSO)[1][1].

An adaptor which performs optimisations based on the state contained within upper layer protocols within a sequence of frames is a *stateful offload adaptor*. Where TCP is the higher level protocol, then a statefull offload adaptor is also known as a TCP Offload Engine (TOE). RDMA optimisations when run over TCP are also therefore termed statefull offloads.

4 Methodology

For the purposes of this study, the offload features as defined in the Taxonomy are grouped into two sets: `NET={regular networking, stateless offload}` and `TOE={stateful offloads}`.

A number of generally available 10Gb/s Ethernet NICs were used for benchmarking. Each NIC vendor is anonymised, we group them as: `{A,B,C,D,E,F,G}`. All NICs are capable of operation in NET configuration

[1]The Linux kernel maintainers hold that TSO can be efficiently performed in software and that contemporary hardware implementations are really only solving layering issues.

and {E,F,G} are also capable of TOE operation. Vendor E has two firmware/driver combinations, E1 which supports NET only operation and E2 which supports both NET and TOE operation.

For each experiment, the vendor supplied tuning parameters were applied for each vendor's NIC. In practice we found that these varied little and are essentially those described by Feng [19]. Interrupt moderation was disabled (where possible) for the latency experiment. A result shown as X indicates that a measurement was not possible in the given configuration, and _ indicates that a measurement was not taken. All experiments were performed back-to-back, using CX4 cable, and using latest software from each vendor in Q3 2006. Windows benchmarking was not undertaken because the available TOE drivers were too unstable for measurement.

For driver availability over all the NICs, testing was first performed on a pair of Intel E7520 dual 2.8Ghz Xeon (EM64T 32bit-mode) / 2GB RAM machines, running Linux 2.6.9. The TOE measurements require the installation of a vendor supplied operating system patch which creates a fast-path from the socket interface to the driver. No operating system by-pass middleware was used. Further experiments designed to investigate the relative performance of the devices over chipset generations were then made where possible using a pair of more recent Nvidia NForce Pro2200/2500 dual AMD285 (dual core 2.8Ghz 32bit-mode) / 4GB RAM machines, running Linux 2.6.17.

5 Results

5.1 Latency

NetPIPE was used to measure the half-round trip latency ($L_{RTT/2}$), results are shown in Table 1. It was not possible to disable interrupt moderation for the NET drivers of vendors E and G.

5.2 Bandwidth and CPU Efficiency

NetPerf was used to measure the CPU efficiency (E) for a single uni-directional stream at peak bandwidth (typically around 32KB message size) and 9KB MTU. Efficiency is expressed as % of a single CPU core per Gb/s. Table 2 shows results on the E7520 platform. Table 3 shows the same experiment on the Nvidia platform where possible.

Vendor		Latency (us)	
		E7520	Nvidia
A	NET	13.7	x
B	NET	17.8	14.9
C	NET	15.4	8.9
D	NET	18.9	13.2
E1	NET	55.5	x
E2	NET	77.7	76.8
E2	TOE	130.0	129.0
F	NET	18.7	_
F	TOE	15.8	_
G	NET	54.4	x
G	TOE	56.3	x

Table 1: Latency

Vendor		Bandwidth (Gb/s)	E(TX)	E(RX)
A	NET	6.2	8.9	7.9
B	NET	6.2	x	x
C	NET	7.2	10.0	8.0
D	NET	5.9	11.9	10.9
E1	NET	7.3	8.9	10.2
E2	NET	3.1	13.0	10.3
E2	TOE	2.4	7.5	6.4
F	NET	4.9	9.2	9.2
F	TOE	4.9	10.0	7.0
G	NET	1.5	33.7	27.0
G	TOE	6.1	13.9	9.0

Table 2: Peak Bandwidth and Efficiency (E7520)

Vendor		Bandwidth (Gb/s)	E(TX)	E(RX)
A	NET	x	x	x
B	NET	6.4	10.2	9.6
C	NET	9.8	6.7	7.6
D	NET	6.8	7.2	8.2
E1	NET	x	x	x
E2	NET	3.4	8.7	7.6
E2	TOE	2.4	6.3	5.5
F	NET	6.8	8.9	10.3
F	TOE	3.6	7.6	7.1
G	NET	x	x	x
G	TOE	x	x	x

Table 3: Peak Bandwidth and Efficiency (Nvidia)

5.3 Multi-Stream Bandwidth

The *HighPerf* script from the Chariot test bench was used to generate 5 simultaneous uni-directional streams on the Intel E7520 platform. Results are shown in Table 4. Bandwidth is as reported by the Chariot test bench. MTU is 9KB. The same experiment was then performed on the Nvidia platform using the *same* NIC (Vendor F) in both its TOE and NET configurations. Results are shown in Table 5. Of particular note is the relative improvement for the NET configuration compared with the TOE.

Vendor		Bandwidth (Gb/s)
A	NET	6.1
B	NET	5.2
C	NET	5.7
D	NET	5.9
E1	NET	5.5
E2	NET	3.4
E2	TOE	6.4
F	NET	5.7
F	TOE	6.5
G	NET	1.3
G	TOE	6.0

Table 4: Chariot HighPerf (E7520)

Platform		Bandwidth (Gb/s)	
		1500mtu	9000mtu
E7520	NET	3.4	5.7
Nvidia	NET	6.1	6.7
E7530	TOE	6.3	6.5
Nvidia	TOE	6.8	6.5

Table 5: Chariot HighPerf (Chipset Comparison)

5.4 RFC Conformance

RFC conformance expressed as a % of ANVL tests passed for each RFC section (in the tool's suite) is given in Table 6 for Linux (2.6.9) and NIC vendors F and G in their TOE configuration. Vendor F does not appear to implement SACK, therefore micro-benchmark data for this device will not extrapolate to good performance in a network which is experiencing loss. Vendor G responds to most ANVL tests by resetting the connection. (A root-cause analysis of this problem would probably quickly improve conformance levels.)

RFC	Conformance (% Pass)		
	Linux	F	G
793	76	71	11
1122	86	77	3
1191	65	6	0
1323	89	84	0
2001	83	67	0
2018	100	18	0
2581	83	67	0

Table 6: RFC Conformance

5.5 Conclusions

The data presented represents a snapshot of 10 Gb/s Ethernet NIC performance on commodity hardware and confirms that line-rate and sub-10 us latency is achievable with or without a full offload implementation. Our opinion is that this situation is very similar to that of 1 Gb/s Ethernet in 2001, and that the current crop of 10 Gb/s offload devices therefore again represents a point solution.

Of note is that the performance improvements from the more recent chipset platform were shown to significantly errode the benefits of a TOE device[2]. It is recommended that experimenters be aware that their measurements and hence conclusions could be very different depending on the test platform.

The RFC conformance data we took should warn users that any statefull offload devices may not be operating at the same level of conformance than that of the operating system being offloaded.

References

[1] H. Bilic, Y. Birk, I. Chirashnya, and Z. Machulsky. Deferred segmentation for wire-speed transmission of large TCP frames over standard GbE networks. In *Proc. 9th Symp. on High Performance Interconnects*. IEEE Computer Society, 2001.

[2] Anecdotally similar results have been obtained on an Intel Woodcrest server.

[2] A. Birrell, R. Levin, M. Schroeder, and R. Needham. Grapevine: an exercise in distributed computing. *Commun. ACM*, 25(4):260–274, 1982.

[3] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W.-K. Su. Myrinet: A gigabit-per-second local area network. *IEEE Micro*, 15(1):29–36, 1995.

[4] R. Brandriff, C. Lynch, and M. Needleman. Development of a TCP/IP for the IBM/370. 15(4), 1985.

[5] P. Buonadonna, A. Geweke, and D. Culler. Implementation and analysis of the virtual interface architecture. In *Proc. ACM/IEEE conf. on Supercomputing*, 1998.

[6] V. Cerf and R. Kahn. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, 22(5), May 1974.

[7] D. Cheriton. Sirpent: A high-performance internetworking approach. In *Proc. ACM SIGCOMM*, 1989.

[8] G. Chesson. Protocol engine design. In *Proc. USENIX*, June 1987.

[9] D. Clark. The design philosophy of the DARPA internet protocols. In *Proc. ACM SIGCOMM*, 1988.

[10] D. Clark, V. Jacobson, J. Romkeym, and H. Salwen. An analysis of TCP processing overhead. *IEEE Communications Magazine*, June 1989.

[11] D. Cohen and G. Finn. The use of message-based multicomputer components to construct gigabit networks. *ACM Computer Communication Review*, 23(3), July 1993.

[12] B. Davie. A host-network interface architecture for ATM. In *Proc. ACM SIGCOMM*, 1991.

[13] P. Druschel, L. Peterson, and B. Davie. Experiences with a high-speed network adaptor: a software perspective. *ACM Computer Communication Review*, 24(4), 1994.

[14] R. S. E Cooper, P Steenkiste and B. Zill. Protocol implementation on the Nectar communication processor. In *Proc. ACM SIGCOMM Symposium on Computer Architectures and Protocols*, 1990.

[15] A. Edwards and S. Muir. Experiences implementing a high performance TCP in user-space. In *Proc. ACM SIGCOMM*, 1995.

[16] A. Edwards, G. Watson, J. Lumley, D. Banks, C. Calamvokis, and C. Dalton. User-space protocols deliver high performance to applications on a low-cost Gb/s LAN. In *Proc. ACM SIGCOMM*, 1994.

[17] T. V. Eicken, A. Basu, V. Buch, and W. Vogels. U-Net: a user-level network interface for parallel and distributed computing. In *Proc. 15th ACM symp. on Operating systems principles*, 1995.

[18] J. Evans and T. Buller. The end of history. In *IEEE TCGN Gigabit Networking Workshop*, April 2001.

[19] W. Feng, J. Hurwitz, H. Newman, S. Ravot, R. Cottrell, O. Martin, F. Coccetti, C. Jin, X. Wei, and S. Low. Optimizing 10-gigabit ethernet for networks of workstations, clusters, and grids: A case study. In *Proc. ACM/IEEE conf. on Supercomputing*, 2003.

[20] G. Finn. An integration of network communication with workstation architecture. *ACM Computer Communication Review*, 21(5), 1991.

[21] M. Hayter and D. McAuley. The desk area network. *ACM Operating Systems Review*, 25(4), October 1991.

[22] P. S. K Kleinpaste and B. Zill. Software support for outboard buffering and checksum. In *Proc. ACM SIGCOMM*, 1995.

[23] M. Kaiserswerth. The Parallel Protocol Engine. *IEEE/ACM Transactions on Networking*, 1(6), 1993.

[24] H. Kanakia and D. Cheriton. The VMP network adaptor board: High performance network communication for multiprocessors. In *Proc. ACM SIGCOMM*, 1988.

[25] J. Kay and J. Pasquale. The importance of non-data touching processing overheads in TCP/IP. In *Proc. ACM SIGCOMM*, 1993.

[26] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3(4), September 1979.

[27] C. Kline. Supercomputers on the internet: A case study. *ACM Computer Communication Review*, 17(5):27–33, 1987.

[28] I. Leslie and D. McAuley. Fairisle: an ATM network for the local area. In *Proc. ACM SIGCOMM*, 1991.

[29] D. McAuley and R. Neugebauer. A case for virtual channel processors. In *Proc. Workshop on Network-I/O Convergence (NICELI).*, 2003.

[30] Message Passing Forum. MPI: A message-passing interface standard. Technical report UT-CS-94-230, 1994.

[31] R. Metcalfe and D. Boggs. Ethernet: distributed packet switching for local computer networks. *Commun. ACM*, 19(7), 1976.

[32] J. Mogul. TCP offload is a dumb idea whose time has come. In *Proc. USENIX HotOS IX Workshop*, 2003.

[33] I. Pratt and K. Fraser. Arsenic: A user-accessible gigabit ethernet interface. In *Proc. IEEE INFOCOM*, 2001.

[34] E. Rutsche. The architecture of a Gb/s multimedia protocol adapter. *ACM Computer Communication Review*, 23(3), 1993.

[35] T. Sterling, D. Savarese, D. Becker, J. Dorband, U. Ranawake, and C. Packer. Beowulf: A parallel workstation for scientific computation. In *Proc. 24th Int. Conf. on Parallel Processing*, 1995.

[36] J. Stone and C. Partridge. When the CRC and TCP checksum disagree. In *Proc. ACM SIGCOMM*, 2000.

[37] C. P. Thacker and L. C. Stewart. Firefly: a multiprocessor workstation. In *ASPLOS-II: Proc. 2nd int. conf. on Architectual support for programming languages and operating systems*, 1987.

[38] C. A. Thekkath, T. D. Nguyen, E. Moy, and E. D. Lazowska. Implementing network protocols at user level. In *Proc. ACM SIGCOMM*, 1993.

[39] J. Touch, T. Faber, and D. Jani. Experience with a production gigabit LAN. In *Gigabit Networking Workshop*. IEEE ComSoc Tech. Comm. on Gigabit Networking, 1997.

[40] C. Traw and J. Smith. Hardware/software organization of a high performance ATM host interface. *IEEE Journal on Selected Areas In Communications*, February 1993.

[41] M. S. Warren, T. C. Germann, P. S. Lomdahl, D. M. Beazley, and J. K. Salmon. Avalon: an Alpha/Linux cluster achieves 10 Gflops for $15k. In *Proc. ACM/IEEE conf. on Supercomputing*, 1998.

[42] M. Wilkes and D. Wheeler. The cambridge digital communication ring. In *Local Area Communications Network Symposium*, Boston, May 1979.

6 Biography

Steven Pope *is a CTO at Solarflare Communications. Previously he co-founded Level 5 Networks and prior to that was a post-doctorate researcher in the field of high-speed networks and operating systems at Olivetti Research Labs, which later became AT&T Laboratories Cambridge. He holds a PhD in Computer Science from the University of Cambridge.*

David Riddoch *is Chief Software Architect at Solarflare Communications. David joined Solarflare with the merger of Solarflare with Level 5 Networks in April 2006. David co-founded Level 5 Networks in July 2002. Previously, David was the architect and lead developer of the software for the CLAN high performance network project at AT&T Laboratories Cambridge. David holds a first class degree in computer science and a Ph.D. in high performance networking from the University of Cambridge.*