Remote Direct Memory Access (RDMA) over IP Problem Statement

Status of This Memo

Abstract

   Overhead due to the movement of user data in the end-system network
   I/O processing path at high speeds is significant, and has limited
   the use of Internet protocols in interconnection networks, and the
   Internet itself -- especially where high bandwidth, low latency,
   and/or low overhead are required by the hosted application.

   This document examines this overhead, and addresses an architectural,
   IP-based "copy avoidance" solution for its elimination, by enabling
   Remote Direct Memory Access (RDMA).

Table of Contents

1. Introduction

   This document considers the problem of high host processing overhead
   associated with the movement of user data to and from the network
   interface under high speed conditions. This problem is often
   referred to as the "I/O bottleneck" [CT90]. More specifically, the
   source of high overhead that is of interest here is data movement
   operations, i.e., copying. The throughput of a system may therefore
   be limited by the overhead of this copying. This issue is not to be
   confused with TCP offload, which is not addressed here. High speed
   refers to conditions where the network link speed is high, relative
   to the bandwidths of the host CPU and memory. With today's computer
   systems, one Gigabit per second (Gbits/s) and over is considered high
   speed.

   High costs associated with copying are an issue primarily for large
   scale systems. Although smaller systems such as rack-mounted PCs and
   small workstations would benefit from a reduction in copying
   overhead, the benefit to smaller machines will be primarily in the
   next few years as they scale the amount of bandwidth they handle.
   Today, it is large system machines with high bandwidth feeds, usually
   multiprocessors and clusters, that are adversely affected by copying
   overhead. Examples of such machines include all varieties of
   servers: database servers, storage servers, application servers for
   transaction processing, for e-commerce, and web serving, content
   distribution, video distribution, backups, data mining and decision
   support, and scientific computing.

   Note that such servers almost exclusively service many concurrent
   sessions (transport connections), which, in aggregate, are
   responsible for > 1 Gbits/s of communication. Nonetheless, the cost

of copying overhead for a particular load is the same whether from few or many sessions.

The I/O bottleneck, and the role of data movement operations, have been widely studied in research and industry over the last approximately 14 years, and we draw freely on these results. Historically, the I/O bottleneck has received attention whenever new networking technology has substantially increased line rates: 100 Megabit per second (Mbits/s) Fast Ethernet and Fibre Distributed Data Interface [FDDI], 155 Mbits/s Asynchronous Transfer Mode [ATM], 1 Gbits/s Ethernet. In earlier speed transitions, the availability of memory bandwidth allowed the I/O bottleneck issue to be deferred. Now however, this is no longer the case. While the I/O problem is significant at 1 Gbits/s, it is the introduction of 10 Gbits/s Ethernet which is motivating an upsurge of activity in industry and research [IB, VI, CGY01, Ma02, MAF+02].

Because of high overhead of end-host processing in current implementations, the TCP/IP protocol stack is not used for high speed transfer. Instead, special purpose network fabrics, using a technology generally known as Remote Direct Memory Access (RDMA), have been developed and are widely used. RDMA is a set of mechanisms that allow the network adapter, under control of the application, to steer data directly into and out of application buffers. Examples of such interconnection fabrics include Fibre Channel [FIBRE] for block storage transfer, Virtual Interface Architecture [VI] for database clusters, and Infiniband [IB], Compaq Servernet [SRVNET], and Quadrics [QUAD] for System Area Networks. These link level technologies limit application scaling in both distance and size, meaning that the number of nodes cannot be arbitrarily large.

This problem statement substantiates the claim that in network I/O processing, high overhead results from data movement operations, specifically copying; and that copy avoidance significantly decreases this processing overhead. It describes when and why the high processing overheads occur, explains why the overhead is problematic, and points out which applications are most affected.

The document goes on to discuss why the problem is relevant to the Internet and to Internet-based applications. Applications that store, manage, and distribute the information of the Internet are well suited to applying the copy avoidance solution. They will benefit by avoiding high processing overheads, which removes limits to the available scaling of tiered end-systems. Copy avoidance also eliminates latency for these systems, which can further benefit effective distributed processing.

In addition, this document introduces an architectural approach to
solving the problem, which is developed in detail in [BT05]. It also
discusses how the proposed technology may introduce security concerns
and how they should be addressed.

Finally, this document includes a Terminology section to aid as a
reference for several new terms introduced by RDMA.

2. The High Cost of Data Movement Operations in Network I/O

A wealth of data from research and industry shows that copying is
responsible for substantial amounts of processing overhead. It
further shows that even in carefully implemented systems, eliminating
copies significantly reduces the overhead, as referenced below.

Clark et al. [CJRS89] in 1989 shows that TCP [Po81] overhead
processing is attributable to both operating system costs (such as
interrupts, context switches, process management, buffer management,
timer management) and the costs associated with processing individual
bytes (specifically, computing the checksum and moving data in
memory). They found that moving data in memory is the more important
of the costs, and their experiments show that memory bandwidth is the
greatest source of limitation. In the data presented [CJRS89], 64%
of the measured microsecond overhead was attributable to data
touching operations, and 48% was accounted for by copying. The
system measured Berkeley TCP on a Sun-3/60 using 1460 Byte Ethernet
packets.

In a well-implemented system, copying can occur between the network
interface and the kernel, and between the kernel and application
buffers; there are two copies, each of which are two memory bus
crossings, for read and write. Although in certain circumstances it
is possible to do better, usually two copies are required on receive.

Subsequent work has consistently shown the same phenomenon as the
earlier Clark study. A number of studies report results that data-
touching operations, checksumming and data movement, dominate the
processing costs for messages longer than 128 Bytes [BS96, CGY01,
Ch96, CJRS89, DAPP93, KP96]. For smaller sized messages, per-packet
overheads dominate [KP96, CGY01].

The percentage of overhead due to data-touching operations increases
with packet size, since time spent on per-byte operations scales
linearly with message size [KP96]. For example, Chu [Ch96] reported
substantial per-byte latency costs as a percentage of total
networking software costs for an MTU size packet on a SPARCstation/20

running memory-to-memory TCP tests over networks with 3 different MTU
sizes. The percentage of total software costs attributable to
per-byte operations were:

        1500 Byte Ethernet 18-25%
        4352 Byte FDDI   35-50%
        9180 Byte ATM   55-65%

Although many studies report results for data-touching operations,
including checksumming and data movement together, much work has
focused just on copying [BS96, Br99, Ch96, TK95]. For example,
[KP96] reports results that separate processing times for checksum
from data movement operations. For the 1500 Byte Ethernet size, 20%
of total processing overhead time is attributable to copying. The
study used 2 DECstations 5000/200 connected by an FDDI network. (In
this study, checksum accounts for 30% of the processing time.)

2.1. Copy avoidance improves processing overhead.

A number of studies show that eliminating copies substantially
reduces overhead. For example, results from copy-avoidance in the
IO-Lite system [PDZ99], which aimed at improving web server
performance, show a throughput increase of 43% over an optimized web
server, and 137% improvement over an Apache server. The system was
implemented in a 4.4BSD-derived UNIX kernel, and the experiments used
a server system based on a 333MHz Pentium II PC connected to a
switched 100 Mbits/s Fast Ethernet.

There are many other examples where elimination of copying using a
variety of different approaches showed significant improvement in
system performance [CFF+94, DP93, EBBV95, KSZ95, TK95, Wa97]. We
will discuss the results of one of these studies in detail in order
to clarify the significant degree of improvement produced by copy
avoidance [Ch02].

Recent work by Chase et al. [CGY01], measuring CPU utilization, shows
that avoiding copies reduces CPU time spent on data access from 24%
to 15% at 370 Mbits/s for a 32 KBytes MTU using an AlphaStation
XP1000 and a Myrinet adapter [BCF+95]. This is an absolute
improvement of 9% due to copy avoidance.

The total CPU utilization was 35%, with data access accounting for
24%. Thus, the relative importance of reducing copies is 26%. At
370 Mbits/s, the system is not very heavily loaded. The relative
improvement in achievable bandwidth is 34%. This is the improvement
we would see if copy avoidance were added when the machine was
saturated by network I/O.

Note that improvement from the optimization becomes more important if
the overhead it targets is a larger share of the total cost. This is
what happens if other sources of overhead, such as checksumming, are
eliminated. In [CGY01], after removing checksum overhead, copy
avoidance reduces CPU utilization from 26% to 10%. This is a 16%
absolute reduction, a 61% relative reduction, and a 160% relative
improvement in achievable bandwidth.

In fact, today's network interface hardware commonly offloads the
checksum, which removes the other source of per-byte overhead. They
also coalesce interrupts to reduce per-packet costs. Thus, today
copying costs account for a relatively larger part of CPU utilization
than previously, and therefore relatively more benefit is to be
gained in reducing them. (Of course this argument would be specious
if the amount of overhead were insignificant, but it has been shown
to be substantial. [BS96, Br99, Ch96, KP96, TK95])

3. Memory bandwidth is the root cause of the problem.

Data movement operations are expensive because memory bandwidth is
scarce relative to network bandwidth and CPU bandwidth [PAC+97].
This trend existed in the past and is expected to continue into the
future [HP97, STREAM], especially in large multiprocessor systems.

With copies crossing the bus twice per copy, network processing
overhead is high whenever network bandwidth is large in comparison to
CPU and memory bandwidths. Generally, with today's end-systems, the
effects are observable at network speeds over 1 Gbits/s. In fact,
with multiple bus crossings it is possible to see the bus bandwidth
being the limiting factor for throughput. This prevents such an
end-system from simultaneously achieving full network bandwidth and
full application performance.

A common question is whether an increase in CPU processing power
alleviates the problem of high processing costs of network I/O. The
answer is no, it is the memory bandwidth that is the issue. Faster
CPUs do not help if the CPU spends most of its time waiting for
memory [CGY01].

The widening gap between microprocessor performance and memory
performance has long been a widely recognized and well-understood
problem [PAC+97]. Hennessy [HP97] shows microprocessor performance
grew from 1980-1998 at 60% per year, while the access time to DRAM
improved at 10% per year, giving rise to an increasing "processor-
memory performance gap".

Another source of relevant data is the STREAM Benchmark Reference
Information website, which provides information on the STREAM
benchmark [STREAM]. The benchmark is a simple synthetic benchmark
program that measures sustainable memory bandwidth (in MBytes/s) and
the corresponding computation rate for simple vector kernels measured
in MFLOPS. The website tracks information on sustainable memory
bandwidth for hundreds of machines and all major vendors.

Results show measured system performance statistics. Processing
performance from 1985-2001 increased at 50% per year on average, and
sustainable memory bandwidth from 1975 to 2001 increased at 35% per
year, on average, over all the systems measured. A similar 15% per
year lead of processing bandwidth over memory bandwidth shows up in
another statistic, machine balance [Mc95], a measure of the relative
rate of CPU to memory bandwidth (FLOPS/cycle) / (sustained memory
ops/cycle) [STREAM].

Network bandwidth has been increasing about 10-fold roughly every 8
years, which is a 40% per year growth rate.

A typical example illustrates that the memory bandwidth compares
unfavorably with link speed. The STREAM benchmark shows that a
modern uniprocessor PC, for example the 1.2 GHz Athlon in 2001, will
move the data 3 times in doing a receive operation: once for the
network interface to deposit the data in memory, and twice for the
CPU to copy the data. With 1 GBytes/s of memory bandwidth, meaning
one read or one write, the machine could handle approximately 2.67
Gbits/s of network bandwidth, one third the copy bandwidth. But this
assumes 100% utilization, which is not possible, and more importantly
the machine would be totally consumed! (A rule of thumb for
databases is that 20% of the machine should be required to service
I/O, leaving 80% for the database application. And, the less, the
better.)

In 2001, 1 Gbits/s links were common. An application server may
typically have two 1 Gbits/s connections: one connection backend to a
storage server and one front-end, say for serving HTTP [FGM+99].
Thus, the communications could use 2 Gbits/s. In our typical
example, the machine could handle 2.7 Gbits/s at its theoretical
maximum while doing nothing else. This means that the machine
basically could not keep up with the communication demands in 2001;
with the relative growth trends, the situation only gets worse.

4. High copy overhead is problematic for many key Internet
    applications.

    If a significant portion of resources on an application machine is
    consumed in network I/O rather than in application processing, it
    makes it difficult for the application to scale, i.e., to handle more
    clients, to offer more services.

    Several years ago the most affected applications were streaming
    multimedia, parallel file systems, and supercomputing on clusters
    [BS96]. In addition, today the applications that suffer from copying
    overhead are more central in Internet computing -- they store,
    manage, and distribute the information of the Internet and the
    enterprise. They include database applications doing transaction
    processing, e-commerce, web serving, decision support, content
    distribution, video distribution, and backups. Clusters are
    typically used for this category of application, since they have
    advantages of availability and scalability.

    Today these applications, which provide and manage Internet and
    corporate information, are typically run in data centers that are
    organized into three logical tiers. One tier is typically a set of
    web servers connecting to the WAN. The second tier is a set of
    application servers that run the specific applications usually on
    more powerful machines, and the third tier is backend databases.
    Physically, the first two tiers -- web server and application server
    -- are usually combined [Pi01]. For example, an e-commerce server
    communicates with a database server and with a customer site, or a
    content distribution server connects to a server farm, or an OLTP
    server connects to a database and a customer site.

    When network I/O uses too much memory bandwidth, performance on
    network paths between tiers can suffer. (There might also be
    performance issues on Storage Area Network paths used either by the
    database tier or the application tier.) The high overhead from
    network-related memory copies diverts system resources from other
    application processing. It also can create bottlenecks that limit
    total system performance.

    There is high motivation to maximize the processing capacity of each
    CPU because scaling by adding CPUs, one way or another, has
    drawbacks. For example, adding CPUs to a multiprocessor will not
    necessarily help because a multiprocessor improves performance only
    when the memory bus has additional bandwidth to spare. Clustering
    can add additional complexity to handling the applications.

    In order to scale a cluster or multiprocessor system, one must
    proportionately scale the interconnect bandwidth. Interconnect

bandwidth governs the performance of communication-intensive parallel
applications; if this (often expressed in terms of "bisection
bandwidth") is too low, adding additional processors cannot improve
system throughput. Interconnect latency can also limit the
performance of applications that frequently share data between
processors.

So, excessive overheads on network paths in a "scalable" system both
can require the use of more processors than optimal, and can reduce
the marginal utility of those additional processors.

Copy avoidance scales a machine upwards by removing at least two-
thirds of the bus bandwidth load from the "very best" 1-copy (on
receive) implementations, and removes at least 80% of the bandwidth
overhead from the 2-copy implementations.

The removal of bus bandwidth requirements, in turn, removes
bottlenecks from the network processing path and increases the
throughput of the machine. On a machine with limited bus bandwidth,
the advantages of removing this load is immediately evident, as the
host can attain full network bandwidth. Even on a machine with bus
bandwidth adequate to sustain full network bandwidth, removal of bus
bandwidth load serves to increase the availability of the machine for
the processing of user applications, in some cases dramatically.

An example showing poor performance with copies and improved scaling
with copy avoidance is illustrative. The IO-Lite work [PDZ99] shows
higher server throughput servicing more clients using a zero-copy
system. In an experiment designed to mimic real world web conditions
by simulating the effect of TCP WAN connections on the server, the
performance of 3 servers was compared. One server was Apache,
another was an optimized server called Flash, and the third was the
Flash server running IO-Lite, called Flash-Lite with zero copy. The
measurement was of throughput in requests/second as a function of the
number of slow background clients that could be served. As the table
shows, Flash-Lite has better throughput, especially as the number of
clients increases.

#Clients	Apache ------ Throughput reqs/s	Flash ----- Throughput	Flash-Lite ---------- Throughput
0	520	610	890
16	390	490	890
32	360	490	850
64	360	490	890
128	310	450	880
256	310	440	820

Traditional Web servers (which mostly send data and can keep most of
their content in the file cache) are not the worst case for copy
overhead. Web proxies (which often receive as much data as they
send) and complex Web servers based on System Area Networks or
multi-tier systems will suffer more from copy overheads than in the
example above.

5. Copy Avoidance Techniques

There have been extensive research investigation and industry
experience with two main alternative approaches to eliminating data
movement overhead, often along with improving other Operating System
processing costs. In one approach, hardware and/or software changes
within a single host reduce processing costs. In another approach,
memory-to-memory networking [MAF+02], the exchange of explicit data
placement information between hosts allows them to reduce processing
costs.

The single host approaches range from new hardware and software
architectures [KSZ95, Wa97, DWB+93] to new or modified software
systems [BS96, Ch96, TK95, DP93, PDZ99]. In the approach based on
using a networking protocol to exchange information, the network
adapter, under control of the application, places data directly into
and out of application buffers, reducing the need for data movement.
Commonly this approach is called RDMA, Remote Direct Memory Access.

As discussed below, research and industry experience has shown that
copy avoidance techniques within the receiver processing path alone
have proven to be problematic. The research special purpose host
adapter systems had good performance and can be seen as precursors
for the commercial RDMA-based adapters [KSZ95, DWB+93]. In software,
many implementations have successfully achieved zero-copy transmit,
but few have accomplished zero-copy receive. And those that have
done so make strict alignment and no-touch requirements on the
application, greatly reducing the portability and usefulness of the
implementation.

In contrast, experience has proven satisfactory with memory-to-memory
systems that permit RDMA; performance has been good and there have
not been system or networking difficulties. RDMA is a single
solution. Once implemented, it can be used with any OS and machine
architecture, and it does not need to be revised when either of these
are changed.

In early work, one goal of the software approaches was to show that
TCP could go faster with appropriate OS support [CJRS89, CFF+94].
While this goal was achieved, further investigation and experience
showed that, though possible to craft software solutions, specific

system optimizations have been complex, fragile, extremely
interdependent with other system parameters in complex ways, and
often of only marginal improvement [CFF+94, CGY01, Ch96, DAPP93,
KSZ95, PDZ99]. The network I/O system interacts with other aspects
of the Operating System such as machine architecture and file I/O,
and disk I/O [Br99, Ch96, DP93].

For example, the Solaris Zero-Copy TCP work [Ch96], which relies on
page remapping, shows that the results are highly interdependent with
other systems, such as the file system, and that the particular
optimizations are specific for particular architectures, meaning that
for each variation in architecture, optimizations must be re-crafted
[Ch96].

With RDMA, application I/O buffers are mapped directly, and the
authorized peer may access it without incurring additional processing
overhead. When RDMA is implemented in hardware, arbitrary data
movement can be performed without involving the host CPU at all.

A number of research projects and industry products have been based
on the memory-to-memory approach to copy avoidance. These include
U-Net [EBBV95], SHRIMP [BLA+94], Hamlyn [BJM+96], Infiniband [IB],
Winsock Direct [Pi01]. Several memory-to-memory systems have been
widely used and have generally been found to be robust, to have good
performance, and to be relatively simple to implement. These include
VI [VI], Myrinet [BCF+95], Quadrics [QUAD], Compaq/Tandem Servernet
[SRVNET]. Networks based on these memory-to-memory architectures
have been used widely in scientific applications and in data centers
for block storage, file system access, and transaction processing.

By exporting direct memory access "across the wire", applications may
direct the network stack to manage all data directly from application
buffers. A large and growing class that takes advantage of such
capabilities of applications has already emerged. It includes all
the major databases, as well as network protocols such as Sockets
Direct [SDP].

5.1. A Conceptual Framework: DDP and RDMA

An RDMA solution can be usefully viewed as being comprised of two
distinct components: "direct data placement (DDP)" and "remote direct
memory access (RDMA) semantics". They are distinct in purpose and
also in practice -- they may be implemented as separate protocols.

The more fundamental of the two is the direct data placement
facility. This is the means by which memory is exposed to the remote
peer in an appropriate fashion, and the means by which the peer may
access it, for instance, reading and writing.

The RDMA control functions are semantically layered atop direct data placement. Included are operations that provide "control" features, such as connection and termination, and the ordering of operations and signaling their completions. A "send" facility is provided.

While the functions (and potentially protocols) are distinct, historically both aspects taken together have been referred to as "RDMA". The facilities of direct data placement are useful in and of themselves, and may be employed by other upper layer protocols to facilitate data transfer. Therefore, it is often useful to refer to DDP as the data placement functionality and RDMA as the control aspect.

[BT05] develops an architecture for DDP and RDMA atop the Internet Protocol Suite, and is a companion document to this problem statement.

6. Conclusions

   This Problem Statement concludes that an IP-based, general solution for reducing processing overhead in end-hosts is desirable.

   It has shown that high overhead of the processing of network data leads to end-host bottlenecks. These bottlenecks are in large part attributable to the copying of data. The bus bandwidth of machines has historically been limited, and the bandwidth of high-speed interconnects taxes it heavily.

   An architectural solution to alleviate these bottlenecks best satisfies the issue. Further, the high speed of today's interconnects and the deployment of these hosts on Internet Protocol-based networks leads to the desirability of layering such a solution on the Internet Protocol Suite. The architecture described in [BT05] is such a proposal.

7. Security Considerations

   Solutions to the problem of reducing copying overhead in high bandwidth transfers may introduce new security concerns. Any proposed solution must be analyzed for security vulnerabilities and any such vulnerabilities addressed. Potential security weaknesses -- due to resource issues that might lead to denial-of-service attacks, overwrites and other concurrent operations, the ordering of completions as required by the RDMA protocol, the granularity of transfer, and any other identified vulnerabilities -- need to be examined, described, and an adequate resolution to them found.

Layered atop Internet transport protocols, the RDMA protocols will
gain leverage from and must permit integration with Internet security
standards, such as IPsec and TLS [IPSEC, TLS]. However, there may be
implementation ramifications for certain security approaches with
respect to RDMA, due to its copy avoidance.

IPsec, operating to secure the connection on a packet-by-packet
basis, seems to be a natural fit to securing RDMA placement, which
operates in conjunction with transport. Because RDMA enables an
implementation to avoid buffering, it is preferable to perform all
applicable security protection prior to processing of each segment by
the transport and RDMA layers. Such a layering enables the most
efficient secure RDMA implementation.

The TLS record protocol, on the other hand, is layered on top of
reliable transports and cannot provide such security assurance until
an entire record is available, which may require the buffering and/or
assembly of several distinct messages prior to TLS processing. This
defers RDMA processing and introduces overheads that RDMA is designed
to avoid. Therefore, TLS is viewed as potentially a less natural fit
for protecting the RDMA protocols.

It is necessary to guarantee properties such as confidentiality,
integrity, and authentication on an RDMA communications channel.
However, these properties cannot defend against all attacks from
properly authenticated peers, which might be malicious, compromised,
or buggy. Therefore, the RDMA design must address protection against
such attacks. For example, an RDMA peer should not be able to read
or write memory regions without prior consent.

Further, it must not be possible to evade memory consistency checks
at the recipient. The RDMA design must allow the recipient to rely
on its consistent memory contents by explicitly controlling peer
access to memory regions at appropriate times.

Peer connections that do not pass authentication and authorization
checks by upper layers must not be permitted to begin processing in
RDMA mode with an inappropriate endpoint. Once associated, peer
accesses to memory regions must be authenticated and made subject to
authorization checks in the context of the association and connection
on which they are to be performed, prior to any transfer operation or
data being accessed.

The RDMA protocols must ensure that these region protections be under
strict application control. Remote access to local memory by a
network peer is particularly important in the Internet context, where
such access can be exported globally.

8. Terminology

   This section contains general terminology definitions for this
   document and for Remote Direct Memory Access in general.

   Remote Direct Memory Access (RDMA)
        A method of accessing memory on a remote system in which the
        local system specifies the location of the data to be
        transferred.

   RDMA Protocol
        A protocol that supports RDMA Operations to transfer data
        between systems.

   Fabric
        The collection of links, switches, and routers that connect a
        set of systems.

   Storage Area Network (SAN)
        A network where disks, tapes, and other storage devices are made
        available to one or more end-systems via a fabric.

   System Area Network
        A network where clustered systems share services, such as
        storage and interprocess communication, via a fabric.

   Fibre Channel (FC)
        An ANSI standard link layer with associated protocols, typically
        used to implement Storage Area Networks. [FIBRE]

   Virtual Interface Architecture (VI, VIA)
        An RDMA interface definition developed by an industry group and
        implemented with a variety of differing wire protocols. [VI]

   Infiniband (IB)
        An RDMA interface, protocol suite and link layer specification
        defined by an industry trade association. [IB]

9. Acknowledgements

   Jeff Chase generously provided many useful insights and information.
   Thanks to Jim Pinkerton for many helpful discussions.

10. Informative References

   [ATM]   The ATM Forum, "Asynchronous Transfer Mode Physical Layer
                Specification" af-phy-0015.000, etc. available from
                http://www.atmforum.com/standards/approved.html.

   [BCF+95]   N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C.
                L. Seitz, J. N. Seizovic, and W. Su. "Myrinet - A
                gigabit-per-second local-area network", IEEE Micro,
                February 1995.

   [BJM+96]   G. Buzzard, D. Jacobson, M. Mackey, S. Marovich, J.
                Wilkes, "An implementation of the Hamlyn send-managed
                interface architecture", in Proceedings of the Second
                Symposium on Operating Systems Design and Implementation,
                USENIX Assoc., October 1996.

   [BLA+94]   M. A. Blumrich, K. Li, R. Alpert, C. Dubnicki, E. W.
                Felten, "A virtual memory mapped network interface for the
                SHRIMP multicomputer", in Proceedings of the 21st Annual
                Symposium on Computer Architecture, April 1994, pp. 142-
                153.

   [Br99]   J. C. Brustoloni, "Interoperation of copy avoidance in
                network and file I/O", Proceedings of IEEE Infocom, 1999,
                pp. 534-542.

   [BS96]   J. C. Brustoloni, P. Steenkiste, "Effects of buffering
                semantics on I/O performance", Proceedings OSDI'96,
                USENIX, Seattle, WA October 1996, pp. 277-291.

   [BT05]   Bailey, S. and T. Talpey, "The Architecture of Direct Data
                Placement (DDP) And Remote Direct Memory Access (RDMA) On
                Internet Protocols", RFC 4296, December 2005.

   [CFF+94]   C-H Chang, D. Flower, J. Forecast, H. Gray, B. Hawe, A.
                Nadkarni, K. K. Ramakrishnan, U. Shikarpur, K. Wilde,
                "High-performance TCP/IP and UDP/IP networking in DEC
                OSF/1 for Alpha AXP", Proceedings of the 3rd IEEE
                Symposium on High Performance Distributed Computing,
                August 1994, pp. 36-42.

   [CGY01]   J. S. Chase, A. J. Gallatin, and K. G. Yocum, "End system
                optimizations for high-speed TCP", IEEE Communications
                Magazine, Volume: 39, Issue: 4 , April 2001, pp 68-74.
                http://www.cs.duke.edu/ari/publications/end-
                system.{ps,pdf}.

   [Ch96]   H.K. Chu, "Zero-copy TCP in Solaris", Proc. of the USENIX
                1996 Annual Technical Conference, San Diego, CA, January
                1996.

   [Ch02]   Jeffrey Chase, Personal communication.

   [CJRS89]   D. D. Clark, V. Jacobson, J. Romkey, H. Salwen, "An
                analysis of TCP processing overhead", IEEE Communications
                Magazine, volume: 27, Issue: 6, June 1989, pp 23-29.

   [CT90]   D. D. Clark, D. Tennenhouse, "Architectural considerations
                for a new generation of protocols", Proceedings of the ACM
                SIGCOMM Conference, 1990.

   [DAPP93]   P. Druschel, M. B. Abbott, M. A. Pagels, L. L. Peterson,
                "Network subsystem design", IEEE Network, July 1993, pp.
                8-17.

   [DP93]   P. Druschel, L. L. Peterson, "Fbufs: a high-bandwidth
                cross-domain transfer facility", Proceedings of the 14th
                ACM Symposium of Operating Systems Principles, December
                1993.

   [DWB+93]   C. Dalton, G. Watson, D. Banks, C. Calamvokis, A. Edwards,
                J. Lumley, "Afterburner: architectural support for high-
                performance protocols", Technical Report, HP Laboratories
                Bristol, HPL-93-46, July 1993.

   [EBBV95]   T. von Eicken, A. Basu, V. Buch, and W. Vogels, "U-Net: A
                user-level network interface for parallel and distributed
                computing", Proc. of the 15th ACM Symposium on Operating
                Systems Principles, Copper Mountain, Colorado, December
                3-6, 1995.

   [FDDI]   International Standards Organization, "Fibre Distributed
                Data Interface", ISO/IEC 9314, committee drafts available
                from http://www.iso.org.

   [FGM+99]   Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
                Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
                Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

   [FIBRE]   ANSI Technical Committee T10, "Fibre Channel Protocol
                (FCP)" (and as revised and updated), ANSI X3.269:1996
                [R2001], committee draft available from
                http://www.t10.org/drafts.htm#FibreChannel

   [HP97]   J. L. Hennessy, D. A. Patterson, Computer Organization and
               Design, 2nd Edition, San Francisco: Morgan Kaufmann
               Publishers, 1997.

   [IB]   InfiniBand Trade Association, "InfiniBand Architecture
               Specification, Volumes 1 and 2", Release 1.1, November
               2002, available from http://www.infinibandta.org/specs.

   [IPSEC]   Kent, S. and R. Atkinson, "Security Architecture for the
               Internet Protocol", RFC 2401, November 1998.

   [KP96]   J. Kay, J. Pasquale, "Profiling and reducing processing
               overheads in TCP/IP", IEEE/ACM Transactions on Networking,
               Vol 4, No. 6, pp.817-828, December 1996.

   [KSZ95]   K. Kleinpaste, P. Steenkiste, B. Zill, "Software support
               for outboard buffering and checksumming", SIGCOMM'95.

   [Ma02]   K. Magoutis, "Design and Implementation of a Direct Access
               File System (DAFS) Kernel Server for FreeBSD", in
               Proceedings of USENIX BSDCon 2002 Conference, San
               Francisco, CA, February 11-14, 2002.

   [MAF+02]   K. Magoutis, S. Addetia, A. Fedorova, M. I. Seltzer, J.
               S. Chase, D. Gallatin, R. Kisley, R. Wickremesinghe, E.
               Gabber, "Structure and Performance of the Direct Access
               File System (DAFS)", in Proceedings of the 2002 USENIX
               Annual Technical Conference, Monterey, CA, June 9-14,
               2002.

   [Mc95]   J. D. McCalpin, "A Survey of memory bandwidth and machine
               balance in current high performance computers", IEEE TCCA
               Newsletter, December 1995.

   [PAC+97]   D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K.
               Keeton, C. Kozyrakis, R. Thomas, K. Yelick , "A case for
               intelligient RAM: IRAM", IEEE Micro, April 1997.

   [PDZ99]   V. S. Pai, P. Druschel, W. Zwaenepoel, "IO-Lite: a unified
               I/O buffering and caching system", Proc. of the 3rd
               Symposium on Operating Systems Design and Implementation,
               New Orleans, LA, February 1999.

   [Pi01]   J. Pinkerton, "Winsock Direct: The Value of System Area
               Networks", May 2001, available from
               http://www.microsoft.com/windows2000/techinfo/
               howitworks/communications/winsock.asp.

   [Po81]   Postel, J., "Transmission Control Protocol", STD 7, RFC
               793, September 1981.

   [QUAD]   Quadrics Ltd., Quadrics QSNet product information,
               available from
               http://www.quadrics.com/website/pages/02qsn.html.

   [SDP]   InfiniBand Trade Association, "Sockets Direct Protocol
               v1.0", Annex A of InfiniBand Architecture Specification
               Volume 1, Release 1.1, November 2002, available from
               http://www.infinibandta.org/specs.

   [SRVNET]   R. Horst, "TNet: A reliable system area network", IEEE
               Micro, pp. 37-45, February 1995.

   [STREAM]   J. D. McAlpin, The STREAM Benchmark Reference Information,
               http://www.cs.virginia.edu/stream/.

   [TK95]   M. N. Thadani, Y. A. Khalidi, "An efficient zero-copy I/O
               framework for UNIX", Technical Report, SMLI TR-95-39, May
               1995.

   [TLS]   Dierks, T. and C. Allen, "The TLS Protocol Version 1.0",
               RFC 2246, January 1999.

   [VI]   D. Cameron and G. Regnier, "The Virtual Interface
               Architecture", ISBN 0971288704, Intel Press, April 2002,
               more info at http://www.intel.com/intelpress/via/.

   [Wa97]   J. R. Walsh, "DART: Fast application-level networking via
               data-copy avoidance", IEEE Network, July/August 1997, pp.
               28-38.

Authors' Addresses

   Stephen Bailey
   Sandburst Corporation
   600 Federal Street
   Andover, MA 01810 USA

   Phone: +1 978 689 1614
   EMail: steph@sandburst.com

   Jeffrey C. Mogul
   HP Labs
   Hewlett-Packard Company
   1501 Page Mill Road, MS 1117
   Palo Alto, CA 94304 USA

   Phone: +1 650 857 2206 (EMail preferred)
   EMail: JeffMogul@acm.org

   Allyn Romanow
   Cisco Systems, Inc.
   170 W. Tasman Drive
   San Jose, CA 95134 USA

   Phone: +1 408 525 8836
   EMail: allyn@cisco.com

   Tom Talpey
   Network Appliance
   1601 Trapelo Road
   Waltham, MA 02451 USA

   Phone: +1 781 768 5329
   EMail: thomas.talpey@netapp.com

Full Copyright Statement

   Copyright (C) The Internet Society (2005).

   This document is subject to the rights, licenses and restrictions
   contained in BCP 78, and except as set forth therein, the authors
   retain all their rights.

   This document and the information contained herein are provided on an
   "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS
   OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET
   ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED,
   INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
   INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
   WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

   The IETF takes no position regarding the validity or scope of any
   Intellectual Property Rights or other rights that might be claimed to
   pertain to the implementation or use of the technology described in
   this document or the extent to which any license under such rights
   might or might not be available; nor does it represent that it has
   made any independent effort to identify any such rights. Information
   on the procedures with respect to rights in RFC documents can be
   found in BCP 78 and BCP 79.

   Copies of IPR disclosures made to the IETF Secretariat and any
   assurances of licenses to be made available, or the result of an
   attempt made to obtain a general license or permission for the use of
   such proprietary rights by implementers or users of this
   specification can be obtained from the IETF on-line IPR repository at
   http://www.ietf.org/ipr.

   The IETF invites any interested party to bring to its attention any
   copyrights, patents or patent applications, or other proprietary
   rights that may cover technology that may be required to implement
   this standard. Please address the information to the IETF at ietf-
   ipr@ietf.org.

Acknowledgement

   Funding for the RFC Editor function is currently provided by the
   Internet Society.