An Overview of RDMA over IP

Allyn Romanow
Cisco Systems
San Jose, CA 95134 USA
allyn@cisco.com

Stephen Bailey
Sandburst Corporation
Andover, MA 01810 USA
steph@sandburst.com

Abstract

This paper gives an overview of Remote Direct Memory Access (RDMA) over IP. The first part of the paper is taken from an internet draft [RMTB02] that describes the problem of high system costs due to network I/O copying in end-hosts at high speeds. A review of experience and research over the last ten or so years shows that the problem is due to limits on available memory bandwidth, and the prohibitive cost of additional memory bandwidth, and that it can be substantially improved using copy avoidance. The second part of the paper considers an RDMA over IP solution, giving background and current status. An architecture is described that is that currently being adopted by the RDMA Consortium and the RDDP WG in the IETF. Outstanding issues, some of a research nature, are considered.

1 Introduction

This paper considers RDMA over IP. Much of the paper is taken from the internet draft, draft-rddp-problem-statement-00.txt [RMTB02], by the authors, and Jeff Mogul and Tom Talpey. While the internet draft describes the problem and reviews the literature, it also motivates why the topic should be in the IETF, and the draft is constrained as a document or an IETF working group. This workshop paper covers the same ground with respect to describing the problem and reviewing relevant literature- in fact we have changed the IETF draft text very little. However, this paper also covers related topics that would be inappropriate for a working group document, but are interesting in a research workshop.

The first section describes what the problem is – high costs of end-host network I/O processing under high data rates - and then reviews literature on different approaches to solving the problem that have been investigated. The next section introduces RDMA over IP and briefly covers some background and current status in industry and standards. Section 4 gives an overview of the architecture currently being used within the RDMA consortium (RDMAC) [RDMAC] and the IETF Working Group, Remote Direct Data Placement (RDDP) [RDDP]. The architecture overview section includes key

concepts, the rationale for architectural choices, and security considerations. The last section describes some of the outstanding issues, some of which might benefit from research efforts.

2 The I/O bottleneck

The high overhead of host processing associated with network I/O that occurs under high speed conditions limits the bandwidth that can be sent between machines. This problem is often referred to as the "I/O bottleneck" [CT90]. More specifically, the source of high overhead of interest here is data movement operations - copying. This issue is not 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 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 in 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 Mbits/s FDDI and Fast Ethernet, 155 Mbits/s 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 [DAFS, IB, VI, CGZ01, 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 allows 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, Infiniband [IB], Compaq Servernet [SRVNET], 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.

From publicly available work, we substantiate the claim that in network I/O processing, high overhead results from data movement operations, specifically copying; and that copy avoidance significantly decreases the 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.

In addition, this document introduces an architectural approach to solving the problem, which is developed in detail in [BT02]. It also discusses how the proposed technology may introduce security concerns and how they should be addressed.

2.1 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 to the costs associated with processing individual bytes, specifically computing the checksum and moving data in memory. They found 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 460 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 - 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, check summing 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 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, B99, 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.)

2.2 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.)

2.3 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.

A common question is whether 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 - 1 for the network interface to deposit the data in memory, and 2 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 back-end 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.

2.4 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 - 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 SAN 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 are a large and growing number of these application servers distributed throughout the Internet. In 1999 approximately 3.4 million server units were shipped, in 2000, 3.9 million units, and the estimated annual growth rate for 2000-2004 was 17 percent [Ne00, PA01].

There is high motivation to maximize the processing capacity of each CPU, as scaling by adding CPUs one way or another has drawbacks. For example, adding CPUs to a multiprocessor will not necessarily help, as 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 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.

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 an optimized server called Flash, and the third 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 1 shows, Flash-Lite has better throughput, especially as the number of clients increases.

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

Table 1: Throughput by Number of Clients for Apache, Flash, and Flash-Lite

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 SANs or multi-tier systems will suffer more from copy overheads than in the example above.

2.5 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], hosts communicate via information that 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 [BP96, 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 NICs [KSZ95, DWB+93]. In software, any 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 changes.

In early work, one goal of the software approaches was to show that TCP could go faster with appropriate OS support [CJR89, 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 re-mapping, 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 for each variation in architecture optimizations must be re-crafted [Ch96].

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 of applications has already emerged which takes advantage of such capabilities, including all the major databases, as well as file systems such as DAFS [DAFS] and network protocols such as Sockets Direct [SDP].

3 An RDMA over IP solution

The desire to put RDMA over IP gained momentum recently due to several business incentives that developed. Servers in data centers often use multiple special-purpose interfaces, such as fibre channel for storage, Virtual Interface Architecture (VIA) [VI] or Infiniband (IB) [IB] for database or other cluster interconnections. By using IP as a high speed interconnect, server management and expense can be simplified and reduced. Another reason is the arrival of 10 Gigabit Ethernet. Without some mechanism to overcome the I/O bottleneck, less than a gigabit per second of network data can comfortably be handled at an end-station. The existence of a general standardized solution over the most commonly used transport means that hardware can be developed for NICs at relatively low cost.

From a feasibility perspective, many individuals and companies have experience building RDMA networking technologies. There is not a recognized feasible alternative approach to solving the I/O bottleneck – special purpose kernel changes are too fragile and need to change too often; there is not a large enough demand to warrant specialized hardware and software systems.

3.1 Background and current status of RDMA over IP

The current effort to use RDMA over IP started with an internet-draft, “TCP RDMA Option”, by David Cheriton and Costas Sapuntzakis in February 2000. (The document is no longer available in the IETF database.) Their draft was partly motivated by concerns about high volume data movement in the context of iSCSI, SCSI over IP, work that was being done in the IP Storage (IPS) Working Group in the IETF [IPS]. They suggested a TCP option to signal the use of RDMA, with support for framing upper level protocol data units in the TCP byte stream. Reception on IETF TCP mailing lists (end-to-end mailing list [E2E]) was largely opposition to the proposal- arguments against included the lack of need for any performance improvements, the recommendation to hack the kernel to improve performance, recollections of the abysmal failure of past efforts to put TCP in hardware, and many other objections.

However, a number of companies in the server business, and their NIC vendors, who were already using RDMA in Fibre Channel, in VIA, and developing it for Infiniband, thought it would be a great idea to put RDMA over IP. The NIC suppliers said they could add support for RDMA in hardware on NICs at low cost if there was an IP standard. Data centers could scale more efficiently, and in the longer term the capability of high speed machine transfers would be more widespread by virtue of cheap RDMA NICs.

Momentum grew amongst companies in storage applications, server and storage machines, and NIC vendors. There was a significant amount of industry experience and enthusiasm with RDMA protocols, such as Servernet, VIA, IB, Scheduled Transfers (STP). A public RDMA mailing list was started on Yahoo Groups, and a wide constituency was solicited. A number of new players joined, many from startup companies. There were frequent open group design meetings.

At the same time, efforts to start a working group within the IETF met considerable opposition, especially from several recognized experts in the IETF transport area. Some of the concerns included: RDMA involves a layer violation; it requires dangerous changes to TCP; it is a special purpose hack for a particular community; all that is required is optimizing the kernel; it is a non-issue because TCP performance has always scaled.

After about a year, responding to commercial concerns to put RDMA protocols in hardware in NICs, several of the larger companies started an industry consortium. The RDMA Consortium (RDMAC) [RDMAC] charter specifies that the organization be short-lived; discussions are closed to nonmembers, but anyone can join; the organization has both a voting and a contributing membership, etc. The purpose of the RDMAC was to produce RDMA specs for possible early implementation experience, and to contribute the specs to an IETF WG, if one were started.

The discussion in the IETF was clarified by separating RDMA functionality from the issues involved with using RDMA over TCP in particular. The notion of RDMA over IP was accepted, while the issue of TCP framing is still very contentious and undecided. The IESG approved a working group in the IETF, called Remote Direct Data Placement (RDDP) [RDDP] in roughly June 2002. The charter of the working group is to define RDMA protocols that are transport independent, with mappings to specific Internet transports, in particular SCTP and TCP. The issue of framing upper level protocol data units in TCP is not in the charter of the RDDP WG but dealt with in the Transport Area Working Group (TSVWG).

The RDMAC has submitted drafts on DDP [SPRC02], and RDMA [RCGH02] to the RDDP WG. The architecture proposed in [BT02] and described here was followed in the current drafts. The drafts were adopted as WG documents at the last IETF meeting in November 2002. As yet there have not been alternative approaches proposed to DDP or RDMAP.

On the other hand, there is a lot more disagreement about the TCP framing issue, which is owned by the TSVWG. The RDMAC submitted a draft for framing called MPA [CER+02]. The draft uses markers for resynchronization. There is not agreement on whether markers are a good approach, or whether resynchronization is needed. An alternative strategy was suggested in [Wi02] and several other alternatives have been proposed in various Internet drafts over the past two years. The entire framing issue is under discussion and the outcome is unknown at this time.

4 RDDP architecture overview

This section gives an informal overview and discussion of some of the key choices of the Remote Direct Data Placement on Internet Protocols (RDDP) architecture. A detailed description of the RDDP architecture can be found in the RDDP Architecture Internet Draft [BAIL02].

RDDP provides a transfer model that allows protocol peers to transfer data to and from abstract, octet-addressable *buffers*. The RDDP buffer abstraction is described in more depth below.

RDDP data transfers can remotely target arbitrary octets in arbitrary buffers. For example, a remote peer could send a message containing 1000 octets of data to buffer `B' starting at address 5000 and

then send a subsequent message with another 1000 bytes, also to buffer `B', starting at address 2000. This *random-access* data transfer capability is the central feature of RDDP. All the mechanics of the RDDP architecture are to provide and support random-access data transfer. However, not all data transferred by RDDP has this random-access character. RDDP distinguishes two types of messages:

Tagged messages: A tagged message updates the contents of a target octet range in a receiver's buffer with the contents of the tagged message. The *tag* is simply a small piece of control information that identifies the buffer and starting octet address.

Untagged messages: An untagged message is delivered directly to the client protocol without any interpretation or modification. Untagged messages are analogous to traditional transport protocol (e.g., SCTP or UDP) messages.

An RDDP *notification* is the other concept used to define the RDDP architecture in addition to the RDDP *buffer*. A notification is an event delivered to the client protocol that signals completion of an RDDP protocol operation. For example, after a tagged message is completely processed by an RDDP receiver, and the target buffer is updated, a notification may be delivered to the client protocol signaling this completion. The RDDP notification abstraction is described in more depth below.

The RDDP architecture is defined in terms of four protocol layers:

- A message-oriented transport protocol. For example, SCTP, UDP, DCCP or TCP with an additional, message delimiting (e.g. message length) layer.
- A Direct Data Placement (DDP) protocol that provides:
  - message segmentation,
  - random-access data transmission
- A Remote Direct Memory Access (RDMA) protocol that provides remote memory access transfer behavior including:
  - memory reading,
  - memory writing,
  - flow control,
  - other memory access primitives, such as atomic memory access.
- A client protocol (of RDMA or DDP).

The context of the DDP and RDMA protocols is shown in Figure 1 and Figure 2.

The primary contribution of the RDMA protocol to the RDDP architecture is the memory reading operation. An RDMA memory writing operation is equivalent to sending a tagged DDP message. On the other hand, RDMA memory reading operation permits a peer to *pull* data from a registered buffer, rather than requiring the other peer to *push* the data. When the RDMA reading operation is implemented in hardware, a pulling transfer requires the same host CPU overhead as a pushing transfer. Push and pull transfers are tools client protocols can use to efficiently implement different flow control and transfer initiative models.

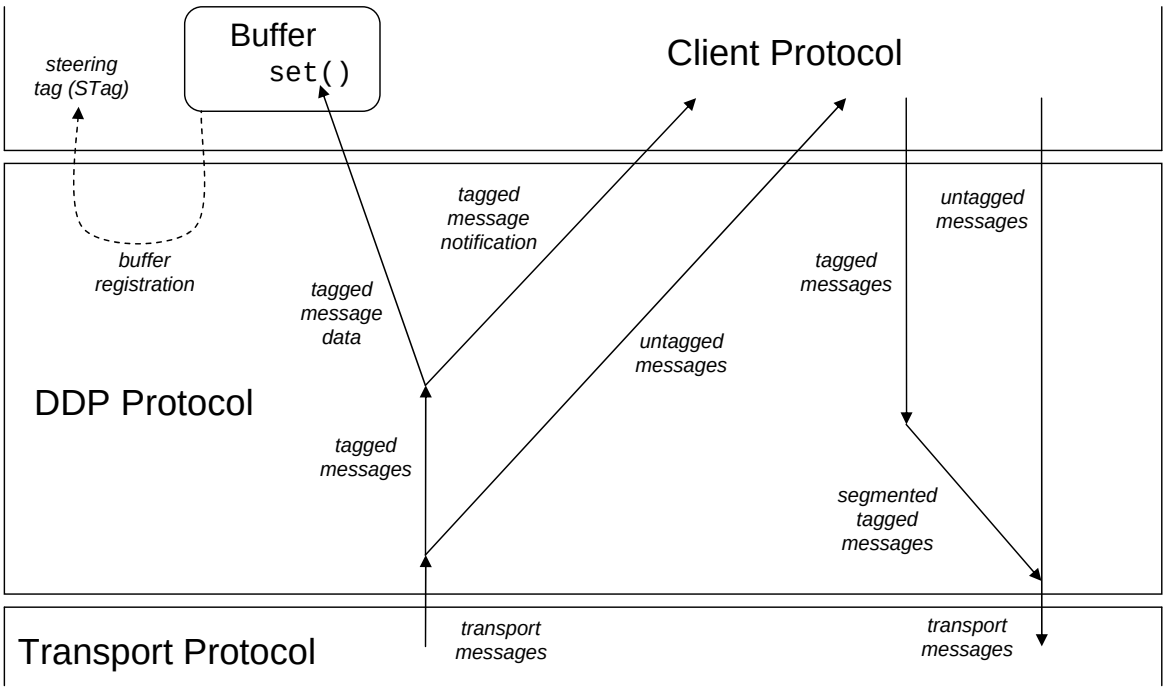

Figure 1 - DDP Protocol Context

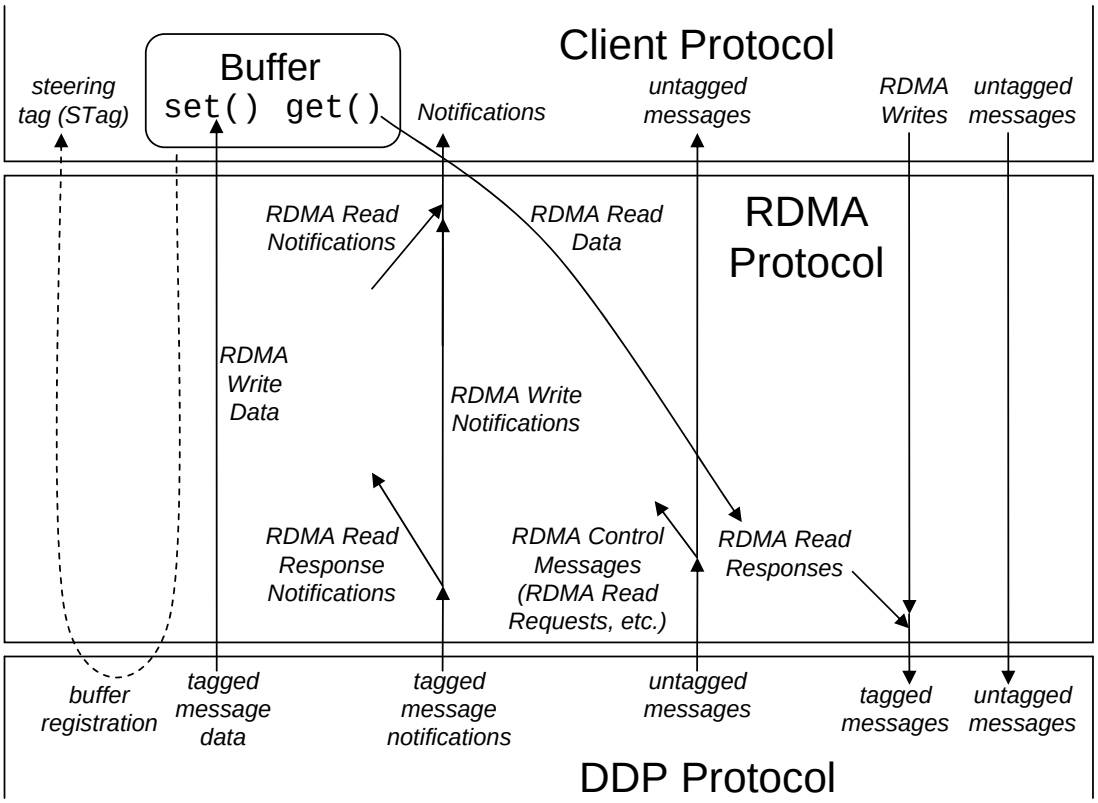

Figure 2 - RDMA Protocol Context

Traditionally, RDMA protocols run directly on the media layer and encompass most or all RDMA, DDP and transport protocol functions in a single protocol. Providing RDMA behavior in the Internet protocol architecture motivates separation of these functions for various reasons, outlined below.

4.1 Buffers

An RDDP buffer is an abstract object that defines a contiguous range of *addresses* from `start` to `end`, and provides two methods for accessing its contents:

`set(address, data)` sets the octet at address to `data`,

`get(address)` returns the value of the octet at `address`.

Client protocols *register* buffers with RDDP protocols. The result of registering a buffer are a *steering tag (STag)* and address range values used by the remote peer to perform tagged data operations on target buffers. After the local peer registers a buffer, it is available for access by the remote peer. However, the local peer must still communicate the buffer parameters to the remote peer to permit access. The RDDP architecture does not define mechanisms for exchanging buffer parameters. Buffer

parameters may be exchanged using any available means, including in-band, or out-of-band communication.

Tagged data operations, including DDP tagged messages, and RDMA Reads and Writes are used to transfer data to and from registered buffers. Untagged data messages are passed directly to the client protocol and do not affect the contents of registered buffers. A remote peer can freely, and randomly access the contents of registered buffers through tagged data operations in any way required by the application.

Many buffers may be registered with RDDP protocols simultaneously. A single buffer may be registered more than once. For example, multiple registrations can be used to make the same buffer accessible simultaneously to more than one remote peer when the transport protocol is point-to-point (e.g., TCP or SCTP).

4.2 Notifications

Notifications to the client protocol include:

- Delivery of an untagged message. The notification includes the contents of the untagged message.
- Delivery of a tagged message (or RDMA Write). The notification includes relevant identification of the tagged message. The sender of a tagged message can control whether a notification is delivered for the tagged message or not.
- Completion of an RDMA Read operation.
- A protocol error report.

RDDP notifications are the `anchors' client protocols use to ensure sequencing and completion of tagged (buffer accessing) operations. The DDP and RDMA protocol definitions, including their transport protocol mappings, define relationships among buffer accessing operations and notifications. These definitions are designed to provide:

- Efficient protocol implementation, and
- Useful client protocol services.

Specifying the relationships among buffer accessing operations and notifications is probably the most substantial component of RDDP protocol definitions. There is a tension between less constrained relationships, which permits more efficient implementation, and more constrained relationships, which may provide client protocol services that are easier to use. For example, if buffer `set()`operations were defined to have no relationship to notifications, an implementation would have complete freedom in performing `set()`s, but the client protocol could never be sure the contents of a buffer reflected the contents of tagged messages from the remote peer.

4.3 RDDP and transport protocols

It is a fundamental choice of the RDDP architecture to delegate concerns of network data transport to traditional Internet data transport protocols. RDDP is defined to strictly layer on top of traditional Internet data transport protocols.

Another important related choice of the RDDP architecture is that some characteristics of the underlying transport protocol are exposed through the DDP and RDMA protocols to the client protocol. In other words, rather than attempting to provide equivalent service to the client protocol no matter what the underlying transport protocol, the unique character of each transport protocol `shines through' in a manner defined by RDDP in the transport protocol mappings. For example:

- RDDP on UDP provides unreliable, unsequenced service, including point to multipoint (multicast) and multipoint to point (unconnected) modes,
- RDDP on SCTP provides reliable unsequenced and sequenced point-to-point service with multiple sequencing domains (SCTP streams).

This approach has two advantages:

- client protocols can exploit characteristics of different transport protocols. For example, latency-sensitive applications can use RDDP on an unreliable transport layer to avoid potentially long pauses associated with retransmission.
- RDDP specification and implementation is much simpler. Providing equivalent behavior across different transport protocols would require emulating a particular generic transport protocol on top of all transport protocols.

Although the RDDP architecture anticipates a variety of transport protocols, most attention has initially been concentrated on reliable, sequenced transport protocols such as SCTP and TCP. This is because most high performance Internet data transport applications *and* existing non-IP RDMA applications are designed for sequenced, reliable data transport.

4.4 Separation of DDP and RDMA

DDP provides the ability to *send* tagged data to modify remote, registered buffers. In other words DDP provides a random-access buffer writing operation. RDMA combines DDP's random-access buffer writing operation with untagged control messages, to create a complete, bi-directional random-access remote memory protocol that includes memory reading, writing, possibly atomic read-and-update, and request flow control.

There are two primary motivations for separating DDP and RDMA in the RDDP architecture:

- Existing RDMA protocols have very similar random-access data transfer mechanisms, but substantially different control models. For example, the Virtual Interface Architecture (VIA) does not define any request flow control mechanism, requiring the client protocol to implement it, if necessary, whereas Infiniband does define a request flow control mechanism. Separating DDP and

RDMA allows development of the control model while ensuring that the core random-access data transmission capability can be safely implemented in hardware.
- Some existing Internet application protocols define their own control models, and, thus, may be more efficiently mapped directly to DDP than to RDMA.

A few applications will be run directly over DDP. Most applications will use RDMA rather than DDP directly, because of the more comprehensive services provided by RDMA. And in some cases applications will run on an additional intermediate protocol that sits on RDMA, such as Sockets Direct Protocol (SDP) [SDP].

4.5 Security considerations

The security work has not been done at the time of this writing. A threat analysis is being undertaken for the protocols.

There is obvious concern that opening up the memory on one machine to a remote machine by passing information about buffer location creates specific security vulnerabilities. What if the STags are compromised? Buffer overwrites are of particular concern. The protection of STags must be considered in detail.

[BSW02] brings up 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, and the granularity of transfer.

Since RDMA/DDP is on top of an IP transport, SCTP or TCP, IPSec and TLS may be used. The details of using IPSec for these protocols need to be investigated.

5 Unresolved issues in RDDP

These issues are currently the subject of disagreement, and/or have the potential for research:

- **TCP Out-of-Order-Processing** - The importance of allowing and facilitating out-of-order RDDP processing of TCP segments. The primary argument is whether the message layer placed on top of TCP should allow an RDDP receiver implementation to place the data that arrives in tagged messages even when TCP segments are received out of order.

  Very simple message layers, such as preceding each RDDP message with a length value, would not provide this capability. Many different, more complex message framing alternatives have been proposed, including trying to ensure that RDDP messages are completely contained within a TCP segment, or inserting periodic resynchronization indications in the TCP stream.

  However, both the effectiveness and necessity of doing this are debatable, as are the performance characteristics of some of the proposals. It has also been argued that some proposals entail a change to the TCP protocol.

- **Checksum** - The adequacy of the existing TCP checksum for ensuring data integrity when transferring large quantities of data typical of RDDP applications is questionable.

- **Flow Control** - The interaction of transport protocol and RDMA flow control mechanisms is not completely understood.

- **Flow Control** - The interaction of RDMA and various client protocol flow control mechanisms is not completely understood.

- **Atomic Update** - The utility and viability of atomic memory update operations in RDDP. Atomic memory update operations have often been provided by RDMA protocols. It is unclear whether the characteristics of RDDP, especially the network-layer and reliable transport layer performance characteristics, reduce the usefulness of atomic memory update operations for traditional RDMA applications (e.g., clustered, shared memory). If the behavior of atomic memory operations in RDDP is not adequate for traditional RDMA applications, are there other compelling uses of atomic operations?

- **Ordering Constraints** - What are the performance and implementation implications of ordering constraints on operations (such as, buffer access, message and notification delivery)? On one hand, client protocols need useful guarantees on memory coherency properties of buffer access. On the other side, ordering constraints, such as strong coherency may cause poor performance under some (or all) circumstances.

- **RDDP with other Transports** - The utility and viability of RDDP on unsequenced , unreliable or multipoint transport protocols. Is it possible to develop useful, consistent semantics for RDDP (or some subset) on unreliable, unsequenced, or multipoint data transports, with or without update of the current RDDP model? Are there compelling applications for any of these RDDP/transport layer combinations?

- **SCTP or TCP?** - What transport should RDMA be over - TCP or SCTP? The argument for SCTP is that RDMA requires frames and SCTP has frames, whereas TCP is byte oriented. So, clearly SCTP is a better architectural choice. The only issue is that SCTP is not widely deployed as is TCP. The NIC manufacturers can add RDMA protocols to TOE NICs relatively inexpensively. Adding support for RDMA/SCTP would be much more expensive, as there are not yet plans to implement SCTP in hardware.

- **Network Behavior** - The impact of high speed, (hardware-driven) endpoint implementations on overall network behavior. RDDP implementations and applications are capable of creating extremely high network traffic loads consistently, and it is unclear whether this will create unanticipated behavior in the network or between peers.

- **Protocol Behavior** - There are protocol implications of end hosts keeping up with fast line rate. How do TCP and other transports perform? This might make changes like the congestion window change proposed by Sally Floyd for High Speed TCP [Fl02] more critical.

- **Other Bottlenecks** -What if the RDMA over IP effort is successful? Does that mean that multi-gigs can be moved at close to line rate? Or are there other bottlenecks? What is the next bottleneck?

- **New Applications** - What new applications would be enabled by widespread host-to-host high bandwidth capability? One possibility is that thin client, server-based computing, where all the computing is done on server and output sent to remote host, might be significantly more attractive.

- **Jumbo Frames –** What is the interaction with Ethernet jumbo frames? Are jumbo frames still needed? Needed more than ever?

6 Acknowledgements

Credit goes to Tom Talpey and Jeff Mogul who substantially contributed to the material in the sections taken from the internet-draft. Jeff Chase generously provided many useful insights and information. Thanks especially to Jim Pinkerton for many helpful discussions.

7 References

[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

[CER+02] P. Culley, U. Elzer, R. Recio, S. Bailey et. Al, “Marker PDU Aligned Framing for TCP Specification”, http://www.ietf.org/internet-drafts/draft-culley-iwarp-mpa-01.pdf, Work in Progress, November 2002

[BSW02] D. Black, M. Speer, J. Wroclawski, "DDP and RDMA Concerns", http://www.ietf.org/internet-drafts/draft-rddp-concerns-00.txt, Work in Progress, December 2002

[BT02] S. Bailey, T. Talpey, "The Architecture of Direct Data Placement (DDP) And Remote Direct Memory Access (RDMA) On Internet Protocols", Work in Progress, http://www.ietf.org/internet-drafts/draft-rddp-architecture-00.txt, December 2002

[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.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

[DAFS] Direct Access File System http://www.dafscollaborative.org/, http://www.ietf.org/internet-drafts/draft-wittle-dafs-00.txt

[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

[E2E] End-to-end-interest mailing list archive http://www.postel.org/mailman/listinfo/end2end-interest

[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

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

[FIBRE] Fibre Channel Standard, http://www.fibrechannel.com/technology/index.master.html

[Fl02] S. Floyd, “HighSpeed TCP for Large Congestion Windows”, http://www.ietf.org/internet-drafts/draft-floyd-tcp-highspeed-01.txt, Internet Draft, August 2002, (see also http://www.icir.org/floyd/hstcp.html).

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

[IB] InfiniBand Architecture Specification, Volumes 1 and 2, Release 1.0.a. http://www.infinibandta.org

[IPS] IP Storage (IPS) IETF Working Group, http://ietf.org/html.charters/ips-charter.html

[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)", accepted for publication at 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

[MYR] Myrinet, http://www.myricom.com/

[Ne00] A. Newman, "IDC report paints conflicted picture of server market circa 2004", ServerWatch, July 24, 2000 http://serverwatch.internet.com/news/2000_07_24_a.html

[Pa01] M. Pastore, "Server shipments for 2000 surpass those in 1999", ServerWatch, February 7, 2001 http://serverwatch.internet.com/news/2001_02_07_a.html

[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". http://www.microsoft.com/windows2000/techinfo/howitworks/communications/winsock.asp

[Po81] J. Postel, "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, September 1981

[QUAD] Quadrics Ltd., http://www.quadrics.com/

[RCGH02] R. Recio, P. Culley, D. Garcia, J. Hilland, “An RDMA Protocol Specification”, http://www.ietf.org/internet-drafts/draft-recio-iwarp-rdma-01.txt, Work in Progress, November 2002.

[RDDP] Remote Direct Data Placement (RDDP) IETF Working Group, http://ietf.org/html.charters/rddp-charter.html

[RDMAC] RDMA Consortium http://www.rdmaconsortium.org/home

[RMTB02] A. Romanow, J. Mogul, T. Talpey, S. Bailey, "RDMA Over IP Problem Statement", http://www.ietf.org/internet-drafts/draft-rddp-problem-statement-00.txt, Work in Progress, December 2002

[SDP] Sockets Direct Protocol v1.0

[SPRC02] H. Shah, J. Pinkerton, R. Recio, P. Culley, “Direct Data Placement over Reliable Transports”, http://www.ietf.org/internet-drafts/draft-shah-iwarp-ddp-01.pdf, Work in Progress, November 2002

[SRVNET] Compaq Servernet, http://nonstop.compaq.com/view.asp?PAGE=ServerNet

[STREAM] 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

[VI] Virtual Interface Architecture Specification Version 1.0, http://www.vidf.org/

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

[Wi02] J. Williams, “iWARP Framing for TCP”, http://www.ietf.org/internet-drafts/draft-williams-iwarp-ift-00.txt, Work in Progress, October, 2002

Copyright Statement