Towards a Virtualized Data Center Transport Protocol

K. Kant
Intel Corporation
krishna.kant@intel.com

***Abstract*—This paper proposes a new transport protocol designed for large virtualized data centers of the future. The underlying premise of this work is that it is possible to achieve significant added functionality with a new transport layer w/o materially affecting the underlying IP and MAC layers. The paper motivates the need for such a protocol, discusses the required features and shows how they can be implemented by starting with an existing and popular transport, namely SCTP.**

Keywords: TCP, SCTP, Infiniband, RDMA, virtualization, QoS, Congestion control.

I. Introduction

The torrid growth of online services and hence of data centers has exposed many deficiencies in existing data center networks. The purpose of this paper is to explore a new transport protocol that can better fulfill the needs of emerging virtualized data centers.

As detailed in the next section, there are many reasons to force a rethink of network protocols for data centers, including very high data rates (e.g., 100 Gb/sec Ethernet), evolution in media types (wired, wireless, optical, etc.), multi-level virtualization, security/isolation issues, move from SMP (symmetric multiprocessor) to cluster enabled applications, etc. From a purely technical perspective, these considerations call for extensive changes to the entire network stack, e.g., MAC, IP and transport. However, our goals here are much more moderate: we want to investigate how well we can fulfill the data center needs via a more powerful and intelligent transport layer on top of the ubiquitous and slowly evolving IP/Ethernet infrastructure. There are, of course, several complete data center fabrics in existence such as infiniband (IBA), Quadrics, QsNet, Myrinet, etc. [13] and we do not wish to create another. However, the paper does attempt to demonstrate that in several ways, the emerging data center needs go beyond what is offered by these bonafide data center fabrics.

The proposals here are based on extensive prior research into the issues addressed here, including some of our own work [7]-[11]. Several of the features mentioned in later sections are even available in existing protocols and implementations, and are not claimed as new. However, the main contribution of the paper is in putting the requirements and features for a virtualized data center transport as a cohesive package. We note at the outset that this paper does not intend to showcase the results from an actual implementation of the new protocol, but merely discuss the relevant ideas and implementation approaches.

The rest of the paper is organized as follows. In section II, we discuss the emerging data center networking needs, the requirements they impose, and how the current protocols stack up against these requirements. In section IV, we discuss the design of the data center transport in view of these requirements. Finally, in section V we summarize the work and identify areas for future work.

II. Data Center Requirements and Transport Protocols

A. Transport Protocol Requirements

The most obvious reason to rethink transport layer is the rapid evolution in the underlying media. It is well recognized by now that the emerging high data rates require lightweight protocols/implementations in order to simply keep up with the line rate. For example, even at 10 Gb/sec, TCP processing requires both the hardware acceleration [7] and techniques such as queue-pairs, user-mode IO, interrupt batching, etc. [2], [5]. On the application side, high data rate MAC allows a single physical "pipe" (e.g., Ethernet cable) to be used to carry a wide variety of traffic streams simultaneously (e.g., storage and inter-process communication (IPC) traffic running over Ethernet), thereby requiring more sophisticated quality of service (QoS) than provided by current fabrics. Third, the media choices continue to multiply (e.g., Ethernet, Optical, WLAN, PCI-Express, UWB, 60GHz, etc.) each of which can be deployed in data centers. Clearly, futuristic transport protocols should run well on most such fabrics.

Let us start with the requirements for the most common situation: a large data center with all servers in one geographic location (usually one building) and using a wired network. This configuration has the following main characteristics:

1) Small round-trip times (RTTs) and yet rather high BW-delay product.
2) Almost no error related packet drops. Buffer overflow related losses are highly undesirable.
3) Very high data rates (e.g., 10 Gb/sec now and going to 100 Gb/sec in the future).

These characteristics have several consequences. First, low CPU overhead and latency are far more important than the throughput under error conditions. In particular, procedures such as SACK (selective acknowledgements) are undesirable since they add significant complexity and buffering requirements to handle rare events as shown in [8]. Also, to prevent

packet losses, a delay based throughput control (as in TCP-Vegas) is much more desirable than a loss based control [14]. Finally, scalability to 100 Gb/sec requires such attributes such as proper field layout, small transmission control blocks, zero-copy, user-level network access, Kernel transition avoidance, and maximization of parallelism in protocol processing. Since many of these aspects have been well researched over the last decade [5], we shall not discuss these further.

Although Ethernet is expected to remain the dominant MAC layer, other MACs (e.g., WLAN, UWB, Optical) are becoming attractive in the entire data center. For example, wireless technologies eliminate the wire management problem and allow for ad-hoc addition/deletion to the infrastructure. Similarly, an all optical network could be very attractive for data centers (or portions therefore) requiring very high BW. In order to accommodate multiple MAC layers, it is necessary to deploy congestion/flow control mechanism appropriate for each MAC. The adaptable congestion control discussed in section IV-C addresses this issue.

Since the transport layer often carries several higher level protocols with varying characteristics, it is necessary for it to include a user-level protocol (ULP) indicator. Such an indicator can be exploited for appropriately handling the processing of sends and receives w/o having to peek-into and understand the precise ULP. Some examples of popular ULPs include RDMA [15], iSCSI, HTTP/SOAP, etc. Respecting and retaining message boundaries of ULP is a highly desirable capability (that, for example, TCP doesn't have).

From an on-line services perspective, there is a strong trend towards a computing model often referred to as "utility", "cloud" or "scaleout" computing, and this has several implications for the transport layer. The major drivers for cloud computing include:

1) Low server utilization. Most data center servers show a very low CPU utilization (5-10% range). This is not so much driven by unpredictable Internet traffic, but by the need to isolate various applications and activities (development vs. live traffic). Virtualization of computes, storage and networking can address this problem while maintaining the desired isolation between VMs.
2) High cost of large SMPs and storage systems which drives the trend of application "clusterisation". Cluster enabled applications can run efficiently on inexpensive uni- or dual-processor systems with distributed local storage, provided that that the inter-process communication (IPC) and the networked storage traffic can be transported with appropriate QoS.
3) High management costs. With management and operations costs in a data center becoming dominant, it is significantly cheaper to purchase data center services from a consolidator rather than running one's own data center. Virtualization again plays a key role in enabling such "outsourced data centers".

Given a virtualized scaleout environment, each clusterised application can be thought of running on an appropriate "virtual cluster" (VC), i.e., a set of virtual machines (or virtual nodes) connected via virtual communication links. These virtual nodes and links, in turn, need to be mapped to real nodes (servers) and communication pathways. From a networking perspective, we need to tag communications with a cluster tag and enforce QoS so that the application can run well. In other words, *the QoS is to be thought of at the level of virtual clusters rather than an individual flow between two endpoints.* This is the main distinction between the type of QoS we discuss here and the traditional QoS notions. In particular, enforcing application or virtual cluster level QoS requires coordination among all virtual nodes as discussed in IV-B. In fact, given the importance of supporting clustered applications well over a common fabic such as Ethernet, we name the proposed transport protocol as Unified Scaleout Transport Protocol (USTP).

Power/thermal issues are becoming critical in data centers due to unsustainable power density, high power consumption, and high cooling costs. Although much of the focus in power management is on the end-nodes, we believe that the power needs to be considered as a first class resource for communication as well, as discussed in section IV-D.

In a distributed virtualized environment, it is important to make the transport layer resistant to denial of service (DoS) attacks, provide adequate data integrity checking, and support some level of challenge-response type of checking. It is also important to support basic high availability mechanisms such as multi-homing, connection migration, path diversity, and path control.

III. Requirements vs. Existing Transports

Feature	TCP	SCTP	IBA
Scalability to 100 Gb/s	difficult	difficult	Easy
Msg. based & ULP support	No	Yes	Yes
QoS friendly transport?	No	No	Yes
Virtual cluster support	No	No	limited
DC centric flow/cong. control	No	No	limited
Power aware transmission	Limited	limited	No
High availability features	Poor	Fair	Fair
Compatible w/ TCP/IP base	Yes	Yes	No
Protection against DoS attacks	Poor	Good	No

Table I
Comparison of Various Transports

In this section we evaluate existing transports in view of the requirements discussed above. Table I provides a eye-chart involving TCP/UDP, SCTP (Stream control transmission protocol) and IBA (Infiniband) protocols. We consider TCP/UDP as a group since the combination is necessary to cover all important data center applications; however, most of the comments are necessarily directed towards TCP. There are numerous variants of TCP for WAN use (e.g., TCP-Reno/NewReno, TCP-Vegas, etc.) and also for high speed

networks (e.g., HS-TCP, Fast-TCP, etc.) [12]. We allow for any of these variants in this comparison, since most variants are focussed on improving loss performance. From a data center perspective, a delay based TCP (e.g., Vegas) is most important, and simpler window control procedures are more desirable than complex ones. Numerous other TCP alternatives have also been offered in the past; for example, the Xpress transport protocol (XTP). However, we shall confine the comparison to the well-known protocols only. One important protocol that we do not list in Table I is DCCP (datagram congestion control protocol), since it is subsumed by SCTP. (DCCP provides unreliable datagram traffic over a connection).

SCTP [3] is an important protocol to consider because of its IP base, closeness to TCP, telecom origins, and recent IETF efforts to position it as a general purpose transport. Although, SCTP shares flow and congestion control algorithms with TCP, it provides a number of useful enhancements over TCP: (a) multiple parallel streams within a connection to carry different types of data effectively over one connection, (b) multihoming capability to allow redundant interfaces for a connection (or "association"), (c) significantly better robustness (32-bit CRC, peer verification during connection setup, etc.), (d) good extensibility due to its "chunk" structure, (e) message boundary preservation, and (f) flexible message delivery, ie., ordered or unordered, reliable or unreliable, etc. However, SCTP was not designed for data centers, shares many of the deficiencies with TCP, and has significantly higher complexity than TCP.

IBA is an important candidate from the perspective of data center features it supports. Other specialized protocols such as Myrinet, Giganet, QsNet, etc. are not considered here since IBA is the most up to date and commercially successful fabric in this class [13]. IBA would be a good starting point for developing a virtualized data center protocol, except that it is not IP/Ethernet based and requires an entirely new infrastructure: cables, connectors, NICs, switches, routers, etc.

IBA supports "virtual lanes" – a concept similar to SCTP's streams – and "service levels" that can be mapped to virtual lanes on a hop by hop basis. A credit based flow control coupled with direct support of service levels and virtual lanes in endpoints, switches and routers allows IBA to provide a good end to end QoS support [1]. In contrast, the greedy window flow control supported by TCP/SCTP generally makes QoS control very difficult in spite of extensive IP level QoS support. It is well known that congested TCP flows divide up the available bandwidth equally. While this is often regarded as a "fairness" property of TCP, it also makes it nearly impossible to grant bandwidth to applications in proportion to their requirements or designated shares.

A related point to note here is that data center infrastructure is dominated by (layer 2) switches, not routers. Since the QoS mechanisms at Ethernet level are currently rather weak, achieving good QoS at transport layer is made particularly difficult. The ongoing standard's work on data center ethernet provides for backward congestion notification at switches and differentiated treatment based on the type of traffic [17]

IV. Design of Data Center Transport

A. From SCTP towards USTP

Although USTP can be designed from scratch based on the requirements above, we believe that SCTP forms a good starting point since it is already a standard protocol that has many of the desired features. Unfortunately, as verified by our investigation in [8], SCTP is too heavy duty and would have to be thinned out. SCTP's extensibility can be exploited for adding new feature w/o changing the nature of the protocol substantially. These two aspects, namely simplification and enhancements to SCTP are discussed in this section.

SCTP connection setup procedure attempts to avoid the "SYN attack" problem of TCP by not creating a TCB when the "INIT" message is received. This can lead to many race conditions that make the connection setup quite complex. It also means that the same information must be resent in the "cookie-echo" message and re-examined on the other end. Building a cookie involves secure hashing and a later verification. Since the connections within a data center are expected to have a small RTT, the SYN attack problem can be handled more easily by keeping an orphaned TCB only for a short period of time. Similarly, secure hashing could be made a configurable feature.

A SCTP message can consist of multiple "chunks" which allows transmission of data from multiple streams into a single message. This "chunk bundling" involves quite a bit of overhead and involves an additional copy. It can be done away without much loss of functionality in most cases. With one chunk per MTU, the chunk header becomes an extension of the common header and can be compressed down. In the current SCTP implementations, multiple streams per connection don't scale very well as shown in [8]. Additional optimizations, such as more granular locking of TCB components can benefit SCTP streams; however, we believe that the streaming feature can be simply removed in USTP. SCTP congestion control can also be simplified for the traditional wired environment, as discussed in section IV-C.

SCTP can be enhanced in many directions by introducing special chunk types. In particular, additional chunks in the INIT message can be used to convey QoS, virtualization & power control information. Needless to say, subsequent messages such as INIT-ACK will also be affected in many cases. In the next several subsections we discuss the basic features in each case. Unfortunately, the space does not permit a discussion of implementation details at the message/field level.

The basic multihoming scheme in SCTP only supports primary/backup associations, but, there are already proposals for using multihoming for load balancing [4]. We shall assume that USTP can use these mechanisms and do not discuss this issue any further in this paper.

B. Transport Layer QoS in USTP

The transport layer QoS requires a consistent end-to-end treatment of QoS at all lower layers traversed. For example,

any prioritization at the transport layer must be mapped consistently to IP layer, e.g., appropriate DSCP's (diff-serv code points), to Ethernet QoS (802.3p/q), and to PCI-Express link QoS within the platform. However, we shall only discuss transport layer mechanisms at endpoints and how they work with existing IP/Ethernet mechanisms. In keeping with the requirement of no changes to the infrastructure, USTP will exploit any congestion notification features available at layers 3 or 2 (e.g., ECN at routers, or EBCN at switches), but does not mandate it. Clearly, if congestion notification is not available, a router/switch congestion can only be detected via a packet drop. This means that some sort of probing scheme is essential to find out how far the individual send windows can be increased. However, the proposed scheme differs from TCP's per connection AIMD control in the following ways:

1) Prioritized treatment at transport level. Packets belonging to flows that demand lower latency are processed ahead of others at both transmit and receive ends. The prioritized treatment is reflected at the IP level by the choice of appropriate DSCP and at ethernet level using appropriate CoS (class of service).
2) Admission control. A call to establish a new connection is rejected by the transport layer if it is oversubscribed. Oversubscription is determined based on automatic monitoring of BW usage of existing connections.
3) Pre-configuration: When a connection is established, it starts with a set of known parameters (e.g., starting window size, window limit, burst size, etc.) depending on the endpoint characteristics.
4) Collective resource control. Unlike TCP, connections do not control their rates (or windows, credits, etc. depending on the scheme) independently but collectively as discussed below.

Before going into the mechanism, we first discuss end-to-end QoS needs from both application and transport perspectives.

1) Application QoS Needs: In a data center environment, the primary role of QoS is to ensure that the applications can maintain a good performance under stress situations and are able to provide the service differentiation as mandated by the SLAs. This is an important distinction from much of previous work QoS where the primary goal to is achieve the desired QoS parameter (BW, latency, etc.) for one or more connections carrying the same type of traffic. An application may involve many traffic types with different characteristics. Yet, precise QoS requirements are rarely known and difficult to specify; therefore, schemes that try hard to meet elaborate requirements are unlikely to be useful. From an application perspective, QoS is generally relevant at the following two levels:

1) Inter-application, or comparative treatment of applications (e.g. BW subdivision) during periods of congestion, and
2) Intra-application or comparative treatment of "flows" belonging to an application. A "flow" here is used in the sense of type of traffic – for example, a DBMS application may be characterized as having three traffic flows: query/response (or networking), inter-process communication (IPC), and storage traffic.

At the inter-application level, we need a user specified indicator in terms of relative priority and the nature of application (e.g., streaming vs. transactional). However, the actual resource requirements are usually not known and are estimated dynamically using the following mechanism:

1) Estimate absolute resource requirements per application as the resources *used* during normal (non-congestion) situations.[1]
2) During a congestion period the resource allocation is done in proportion to the requirements obtained in the previous non-congestion period. Note that no enforcement occurs during non-congestion periods.

The intra-application QoS can also be handled in an almost identical manner, with the total available resources to the application divided up between its constituents during congestion periods. However, if the flows represent physically different traffic types (e.g., storage vs. IPC), it may be adequate (and perhaps even preferable) to simply enforce a user specified priority. Either of these policies can be implemented based on the tags that identify the flows and static configuration parameters (see below).

2) Dealing with Multiple Transport Layers: The intent of USTP is to create a single transport layer that provides features to carry multiple traffic types. For example, like SCTP and IBA, USTP is intended to support all four combinations of reliable/unreliable and ordered/unordered delivery.[2] Irrespective of whether various transport needs are satisfied by distinct protocols or different "modes" in the same protocol, it becomes necessary to allow for communication with varying transport characteristics. The major issues to consider are as follows:

1) Sharing between connection oriented and connectionless transports, e.g., USTP vs. UDP. This requires a coordination mechanism that goes across various flows. In particular, USTP can provide statistics that can be used by a higher layer to constrict UDP flows in a fair manner.
2) Sharing between multiple connection-oriented transports (e.g., USTP vs. TCP). This is a critical issue for gradual introduction of USTP and is addressed in section IV-E.

A cross-transport QoS control is tricky since the overall behavior is governed by the least controllable transport. For example, if USTP and TCP connections share the same link, USTP may put a collective limit over the BW used by all its connections, but since TCP does not have any collective limit, TCP may gain an unfair advantage. Similarly, if USTP was competing with un-policed UDP, UDP could squeeze USTP

[1] It is assumed that the congestion happens occasionally and not on a sustained basis. Sustained congestion is not a QoS issue and must be handled by other means such as application reconfiguration/migration or physical resource upgrades.

[2] For example, a reliable but unordered delivery is ideal for low-latency RDMA.

severely. These issues are best handled by a higher layer that has visibility into all competing transport flows.

3) Flow Tagging and QoS: In order to support inter- and intra-application QoS in USTP, we define the following two types of tags in USTP packet headers:

1) Connection set tag (CStag): Represents the "application type" in terms of QoS properties.
2) Virtual stream tag (VStag): Represents flows within a scope of application from QoS perspective.

CStag can be used in two ways: (a) to specify relative application priority, and (b) simply as a unique indicator on which to base the automatic resource subdivision discussed in section IV-B1. VStag can be used in an identical way for flows within an application. In addition, in cases where an application designer can provide a more precise indication of relative requirements of different flows, VStag can be used as a "code" to look up such information from a table.

We now consider how CStag can be exploited for the automatic BW control. Let us assume a connection oriented transport using AIMD control. If all the connections at an endpoint remain in congestion avoidance mode for some appropriately chosen time T, we can assume that we are in uncongested mode. Thus, if the connections stay in this mode, every T seconds the transport obtains an estimate of the BW usage by each of the flows. For the purposes of control, BW can be estimated as the window size of each connection and exponentially smoothed over successive interval [10]s.

If one or more connection experiences a congestion event, we need to aggregate this information to estimate the overall available BW and then divide it up in proportion to the BW needs. The details of this collective BW control mechanism [10] and are not included here for lack of space.

The intent of collective BW control is to do a "fair" subdivision of BW among competing applications and between flows within an application. We can also favor one application over another by treating CStag as a "code" that indicates the relative BW share between applications as demonstrated by the results in [10]. Unfortunately, a purely endpoint driven BW control is often inadequate. Ideally, one needs to implement QoS controlled paths at MAC layer with all switches participating in it (data centers mostly use layer2 switches, rather than routers). Such a scheme is presented in [11] which introduces the concept of *virtual links*. The current proposals for data center ethernet [17], [18] are a step in this direction, but only create a few "pipes" based on overall traffic characteristics. These lower layer features can be readily used by our tagging mechanism to set up connections properly depending on the type of the traffic.

C. Adaptable flow/congestion control

A proliferation of high-speed wired, wireless and optical media types implies that entire data centers or portions thereof may opt for different technologies. For example, small data centers or portions of a large data center that do not have huge BW requirements could benefit from wireless technologies as they eliminate wire management and allow for a more ad-hoc operation. Similarly, all optical networks could become cost effective in near future. From a transport perspective, the three media types have the following distinguishing properties:

1) *Wired:* Because of negligible error rate, a loss of packet almost always means congestion. TCP, in fact, assumes that all losses are congestion related.
2) *Wireless:* A packet could be lost either due to error or buffer overflow. Even though the errors are typically handled by MAC level mechanisms, the transport endpoint still must account for delays due to link-level retransmission and errors unmasked by error correction.
3) *Optical:* The near zero error rate and huge BW mean that the flow/congestion control can be extremely simple (e.g., credit based).

Most of these issues are well understood. The only interesting aspect is making the congestion/flow control scheme "pluggable" and negotiable between the peers during connection setup. This would require some support for the transport layer to discover the type of MAC layer (e.g., via the CIM database of the platform), so that it can choose the appropriate congestion control mechanism.

The window flow control protocol used in SCTP is similar to TCP-SACK. For wired networks, a delay based window control (e.g., TCP-Vegas) is very attractive. Even if the loss-based control is retained, the current SACK feature of SCTP is particularly undesirable and needs to be changed or eliminated. First, if SACK is to be issued for every 2 received packets, at most one gap can develop and no elaborate gap reporting mechanism is needed. Second, keeping later packets around to allow filling of gaps is extremely expensive and may require a lot of buffering and buffer management. For a Wireless MAC, TCP-like window control does not work and alternatives such as in [16] can used. In any case, making the congestion control pluggable allows significant freedom in choosing and refining the mechanism as the media needs evolve.

D. Power Aware Transport Protocols

Although the circuit and HW architecture are the best places to minimize power consumption, the way in which HW is used can also have a dramatic effect on power and thermal issues. There are essentially three power reduction "tricks" above the HW level that are relevant in the networking context:

1) Deliberate batching of traffic to enhance opportunities for a device to go into a low power mode.
2) Coordination among multiple devices for a more effective use of low power modes.
3) Minimize the footprint of an activity in terms of number of devices touched.

Both TCP and SCTP already have a rudimentary batching capability via the NODELAY option (on by default). Turning off this option means that the transport will wait until it has 1 MTU worth of data before building a packet and transmitting. This functionality can be easily generalized, e.g., delay the transmission of next burst by some amount τ. The appropriate value of τ obviously depends on the application; therefore,

we propose that the τ value be specifiable as a part of VStag introduced in section IV-B3. This can ensure that both sides of a connection and all related communications (e.g., all IPC messages) use the same τ value.

The batching of communications over a connection allows a more effective power control at the two communicating end points but it usually doesn't benefit the link interface itself. This is because the physical link interface generally will have many data streams going over it. A coordination among all such streams is necessary to achieve interface power savings. Towards this end, in USTP, the τ for each stream is merely considered as a recommendation for layer 2. The layer 2 does the batching by selecting the minimum τ over all data streams. Note that since this is a local implementation issue, there is no impact on standards. With layer 2 batching, there is no additional transport layer batching by the sender; instead, the transport level batching happens on a per stream basis on the receive end (to allow the receiver to enter low power mode for longer periods).

Network processing generally involves several HW devices in a system, e.g., the network interface, IO busses, chipset, memory and processor. While data sends & receives need this involvement, the routine maintenance activities do not have to. For example, SCTP has a heartbeat mechanism and HTTP on top of TCP will effectively do the same (exchange "keepalives"). The problem with these activities is that the entire system must wakeup periodically even if the link is not transmitting any data. This can eat a lot of power unnecessarily. One way to deal with this problem is to simply move the implementation of heartbeat down to the network interface. In USTP, we employ an additional scheme as well: keepalives are shared among all all connections between the same transport endpoints and hence the frequency of keepalives can be reduced substantially.

E. Interoperability with Existing Protocols

For a new protocol to thrive, it is crucial to be able to introduce it gradually. So, it should be possible to handle a situation where only a few servers support USTP while others use TCP/UDP. This requires the following features:

1) A USTP implementation must be able to recognize a TCP SYN message and transfer control to the local TCP module for further processing.
2) USTP servers must support a mechanism to find out which peers do not support USTP. A simple but flexible scheme to do this is to perform a UDP based message exchange before establishing the connection. The result could then be cached for rather long periods so that the net overhead of discovery remains very small.
3) When running in a mixed environment (i.e., TCP & USTP going over the same physical link), it is important to select a USTP congestion control scheme that is "friendly" to TCP [6]. Note that because of the notion of "pluggable congestion control", it is easy to pick a TCP-friendly mechanism only for those links that carry mixed traffic.

V. Discussion and Future Work

In this paper, we introduced a new transport protocol called unified scaleout transport protocol (USTP) for satisfying needs of future virtualized data centers. Although USTP is based on SCTP, it sports a number of optimizations, simplifications and new features to make it more suitable for emerging utility computing environments. The key new ideas in USTP are virtual cluster centric QoS, adaptable flow/congestion control, and power awareness. The protocol exploits SCTPs chunk structure to introduce new capabilities w/o a substantial reengineering of the protocol. Based on the design discussed here, the next step is to create a working prototype of USTP and deploy it for several high performance clustering applications.

References

[1] F.J. Alfaro, J.L. Sanchez, J. Duato, "QoS in Infiniband Subnetworks", IEEE trans. on parallel & distributed systems, vol 15, no 9, sept 2004.

[2] D. Dunning, G. Regnier, et. al., "The Virtual Interface Architecture", IEEE Micro, Vol 18, No 2, Mar-Apr 1998, pp 66 - 76.

[3] A.L. Caro, J.R. Iyengar, et. al., "SCTP : A proposed standard for robust Internet data transport", IEEE Computer, Nov 2003, pp20-27.

[4] A.L. Caro, J.R. Iyengar, et. al., "Using SCTP Multihoming for Fault Tolerance and Load Balancing", www.armandocaro.net/publications/2002.sigcomm.posterAbstract.pdf.

[5] P. Balaji, P. Shivam, P. Wyckoff, D. Panda, "High performance user level sockets over Gigabit Ethernet", Proceedings. 2002 IEEE International Conference on Cluster Computing, pp179- 186

[6] M. Handley, S. Floyd, et. al., "TCP friendly rate control (TFRC) protocol specification", IETF RFC 3448, Jan 2003.

[7] K. Kant, "TCP offload performance for front-end servers", Proc. of GLOBECOM 2003, Dec 2003, San Francisco, CA.

[8] K. Kant and N. Jani, "SCTP performance in Data Center Environments", Proc. of SPECTS, July 2005, Philadelphia, PA.

[9] K. Kant and A. Sahoo, "Clustered DBMS Scalability under Unified Ethernet Fabric", Proc. of ICPP, May 2005, Oslo, Norway

[10] K. Kant, "Application Centric Autonomic BW Control in Utility Computing", in Sixth IEEE/ACM workshop on Grid Computing, Seattle, WA, Nov 2005.

[11] K. Kant, "Virtual Link: An Enabler of Enterprise Utility Computing", to appear in the proceedings of International Symposium on Parallel and Distributed Processing and Applications (ISPA), Sorrento, Italy, Dec 2006.

[12] Y-T Li, D. Leith, R.N. Shorten, "Experimental Evaluation of TCP Protocols for High Speed Networks", IEEE/ACM trans on networking, Vol 15, No 5, Oc 2007, pp1109-1122.

[13] J. Liu, B. Chandrasekaran, et.al., "Microbenchmark performance comparison of high-speed cluster interconnects", IEEE Micro, Vol 24, No 1, Jan.-Feb. 2004 Page(s):42 - 51

[14] J. Martin, A. Nilsson and I. Rhee, "Delay based congestion avoidance in TCP", IEEE/ACM trans on networking, vol 11, no 3, pp356-369, June 2003.

[15] J. Pinkerton, available at www.rdmaconsortium.org/home/The_Case_for_RDMA020531.pdf.

[16] Y. Tien, K. Xu, N. Ansari, "TCP in Wireless Environments: Problems and solutions", IEEE Radio Communications, March 2005, pp27-32.

[17] M. Wadekar, G. Hegde, et. al., "Proposal for Traffic Differentiation in Ethernet Networks", See new-wadekar-virtual%20-links-0305.pdf at www.ieee802.org/1/files/public/docs2005

[18] D. Bergamasco, "Ethernet Congestion Manager (ECM)", See au-bergamasco-ethernet-congestion-manager-070313.pdf at www.ieee802.org/1/files/public/docs2007