Session Layer Burst Switching for High Performance Data Movement

Ezra Kissel and Martin Swany
Department of Computer & Information Sciences
University of Delaware, Newark, DE 19716
{*kissel, swany*}@*cis.udel.edu*

***Abstract*—High performance data movement remains a key problem despite continuing advances in the link speed of networks. The resurgent trend of dynamic networks, in which dedicated network resources can be requested and reserved on-demand, is contributing to an increasingly heterogeneous network landscape that is pushing the limits of existing transport layer protocols. We argue that the effective use of such networks will benefit from a data movement service that departs from traditional approaches by using a session layer protocol to manage the end-to-end connection.**

This paper introduces Session Layer Burst Switching (SLaBS) as an architecture for improving efficiency and utilization of dynamically provisioned network resources. Instead of reserving a dedicated "circuit" per application, SLaBS forms data bursts, at transport-layer gateways in the network, which are able to intelligently schedule and switch these bursts along allocated network paths as they become available. Using session-layer protocol data units, called "slabs", significant amounts of data can be buffered. This allows the burst size to be adapted to the characteristics of the network path and optimally transferred via explicit buffers negotiated between gateways. We have implemented a proof-of-concept prototype of SLaBS and used it to carry out some initial transfer tests. An analysis of slab management and the role of buffering in the network is also presented.

I. Introduction

Achieving reliable, high-speed data transfer performance remains a "holy grail" for many in the research and education (R&E) and e-Science communities, and commonly for those within the high performance network area itself. While available link and backbone capacity have rapidly increased, the achievable throughput for typical end-to-end (E2E) applications has failed to increase commensurately. In many cases, application throughput may be significantly less than what is theoretically achievable unless a considerable amount of effort is spent on host, application, and network "tuning" by users and network administrators alike. At the same time, processor capability is growing, and thus the performance gap, in terms of cycles per byte transferred, is continuing to widen.

The resurgent trend of dynamic networks, in which network resources can be requested and reserved, offers opportunities as well as challenges. Dynamic requests for some sort of differentiated network service have been possible, in various forms, for quite some time. The emergence of wave-division multiplexing in optical links, however, changes the environment in a way that is analogous to the shift toward multiple cores and away from frequency scaling in processors. Essentially, it is less expensive to purchase and operate ten 10GE links than a single 100GE link [1]. With current network architectures, it is arguably more difficult to effectively use these links.

Thus a leading theory for utilizing many parallel links involves dynamically allocating some of them to high-demand flows. This model effectively creates a "hybrid" network that involves both shared network segments and dedicated backbone links along an end-to-end (E2E) path. Traversing such hybrid paths is problematic for commonly used transport-layer protocols like the Transmission Control Protocol (TCP) where conflicting interactions between loss events at the edge, and the traversal of high-latency links, can lead to poor throughput. Backbone links are often engineered and provisioned to be virtually loss free, and temporarily dedicated links should always be loss free for traffic that does not exceed the bandwidth allocation. The key issue is that the lossless, long-haul networks interact poorly with "access layer" networks which, by their design and given the nature of statistical multiplexing, frequently experience some amount of loss.

In the Phoebus project, we introduced a data movement service based on a session-layer protocol that binds E2E communication to a "session", decoupling the context from a single transport-layer connection. This system is based on session-aware, transport-layer gateways. These gateways allow us to adapt or change transport protocols and effect tuning between network segments [1]. Phoebus demonstrates improved performance over a variety of edge and wide-area network scenarios. This work extends that model of decoupling into the time domain by assuming significant buffering in the network. This enables the asynchronous transfer of larger amounts of data for better utilization of lossless links and improved overall throughput.

This paper introduces Session-Layer Burst Switching, or "SLaBS", which improves utilization and performance of dedicated wide-area network resources. The key contribution of this work is the novel application of burst switching using explicit session-layer signaling. We argue that by using fast, plentiful buffer media (e.g. RAM, solid state disk (SSD), etc.) to stage large aggregations of data in slabs, we can enable optimal utilization of dedicated network paths via coordinating gateway-to-gateway bursts at channel capacity for short periods.

II. Overview

We are not alone in claiming that it is time to rethink, and potentially augment, the modes of operation of the Internet. Our session-based approach provides a framework for high throughput data movement in capable networks, but it also subsumes models such as that of Delay Tolerant Networking [2] (DTN), which generally targets less capable networks[2]. A key observation in our model is that bulk data transfer applications are not, in general, sensitive to the latency of a single segment of data, but rather to the overall time it takes to complete the transfer. Moreover, many applications only depend on a certain targeted completion time and thus can readily

[1] While the cost of 100GE interfaces will certainly drop, this is a general observation.

[2] Although the exploration of this case is beyond the scope of this paper.

make use of best-effort "scavenger" services or scheduled resources, as appropriate.

We have asserted that current end-to-end transport protocols are failing to serve the needs of today's networks. This issue has been observed and studied, but the performance gap continues to grow. Our hypothesis is that current end-to-end paradigms must be revisited and revised to provide scalable performance for terabit (and beyond) network infrastructures. One possible alternative to today's end-to-end transport paradigms is that of coarse-grained burst switching.

Burst switching garners its benefit from amortizing overheads over a large data transfer, including overheads from the protocol and from channel arbitration. In order to maintain uninterrupted data movement with our bursting model, relatively significant buffering is required to assemble the bursts being stored and forwarded. The role of buffering in network switches has been a key design consideration since networks became packet switched. Our efforts will attempt to revise current models and apply them to modern dynamically provisioned networks. Our view is that effective use of bursting in terabit speed networks will involve a radical rethinking of network protocols and the role of buffering in them.

We argue that, as data movement requirements, and network speeds, continue to increase, end-to-end closed-loop feedback control for ultra-high speed networks is increasingly strained. Perhaps more importantly, it is not always necessary. This is because closed-loop flow control becomes slower to respond as the feedback loop grows. A particular flow must infer network conditions and availability and thus may become either too conservative or too aggressive based on limited observations. The "length" of the feedback loop in end-to-end TCP is essentially the number of segments that need to be "in-flight" at any given time. This fact tends to degrade performance along long-distance, high BDP network paths [3] that are not engineered to be virtually loss-free. As the amount of data along the path increases, the performance of TCP over large RTT links suffers dramatically as the feedback loop becomes less responsive and cannot adapt quickly enough to congestion and loss.

In our SLaBS bursting model, we assume more control over core network links, and the flow control is negotiated out of band, on a session-based control channel. Congestion and congestive loss are no longer an issue in dynamic backbone networks with dedicated resources. This allows for the utilization of open-loop flow control over these long distance, high-bandwidth links. Flow control is managed at a much coarser granularity and traffic is no longer statistically multiplexed, but rather the capacity for a data burst over a dedicated channel is guaranteed for a certain time slot. An open loop model over the core allows us to determine ahead of time what resources are necessary and to pace burst transfers based on buffer and throughput capabilities at various points in the network.

We also consider the question of buffer management within the network. In most current cases, network buffers are limited in size and are easily overrun by aggressive senders. Part of TCP's success is in enforcing friendliness among competing flows to effectively work around these limitations. We propose a model that introduces the notion of explicit buffers that are immune to being overrun. In the core network, flow credits are issued by the receiving slab daemon that bound the total amount of transmitted data. At the edge, slabs are only filled as there is space available. If additional space cannot be allocated to a slab, then the TCP buffers will fill and the connection will block. Due to the relatively short RTT of these edge connections (given an edge's close proximity to a gateway), when TCP again offers a non-zero window it will rapidly be able to fill the available slab space. This solution is also possible in Layer 2 networks with pause frames, and IP networks with source quench ICMP, but it is difficult to scale at Ethernet frame resolution (1-8k). TCP gives us ideal back pressure in this case. An additional safeguard is to perform slab admission control, or policing at the edge, to keep the load from edge to core reasonable.

A host of new issues must be considered and addressed as we merge the traditional closed-loop flow control system over the IP network and the SLaBS model enabled by dynamic networks. Although we have asserted that an open loop control model can provide certain benefits in such networks, the details of how dynamic network technologies are reserved, provisioned, scheduled, and utilized must be analyzed and we must study appropriate buffer and burst parameters for various cases. In the SLaBS model the explicit buffers themselves, slabs, become the basic unit of data that is switched along the network core. Subsequently, slab formation, size, scheduling, and flow control are all subject to allocation time, the current state of the core network, and the policies enforced by the SLaBS nodes.

III. SLaBS: A Buffer and Burst Model

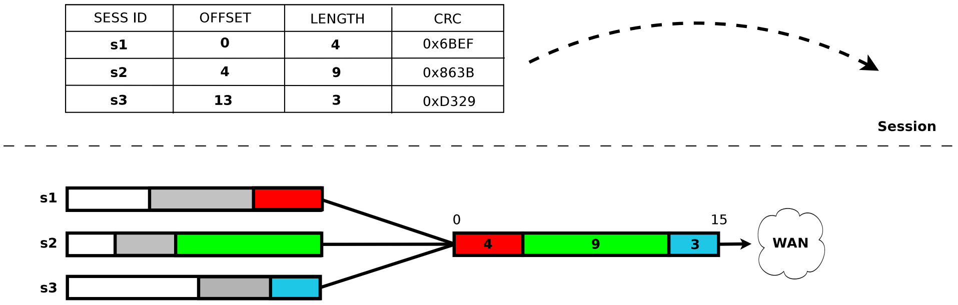
Our model of a slab being switched across a reserved network core is a natural extension of burst switching. The key idea of burst switching is that the overhead of processing (including lookup, forwarding and arbitration) can be mitigated by sending a burst of data, or in other words a relatively large PDU rather than the relatively small network and transport layer PDUs common today. This model, which is reasonably old [4] has seen a resurgence of interest. SLaBS applies this idea at very coarse level of granularity.

The SLaBS bursting model is predicated on three key network qualities. These, along with the level of burst granularity, differentiate it from previous bursting approaches. SLaBS assumes: (i) the existence of a session-layer protocol for IP networks, (ii) the existence of transport-layer gateways that speak this session-layer protocol and offer relatively significant amounts of buffering, and (iii) the availability of dedicated network resources (bandwidth), whether they be static or dynamic.

Our architecture utilizes a session protocol we call XSP for eXtensible Session Protocol. This protocol was developed for Phoebus but has been progressively generalized. XSP defines session-layer protocol data units (SPDUs). We call these SPDUs "slabs" in the context of an XSP session. In the SLaBS model, these SPDUs are marshaled and burst over backbone links. The details of XSP are beyond the scope of this paper.

The transport layer gateways in SLaBS are currently based on the Phoebus Gateway. Phoebus uses specialized PCs running a Unix variant, but versions that run directly on dedicated IP routers are possible (and in development.) Due to the different demands on the current systems (and as they are PCs now) they can have significant amounts of DRAM. This allows us to buffer large bursts that can still be streamed out at line rate on 10G interfaces.

The SLaBS model leverages dedicated resources, which essentially implies some amount of bandwidth dedication. This can take the form of MPLS tunnels, VLANs, SONET timeslots, or dedicated DWDM lambdas. A dedicated channel allows us to use efficient protocols, including application-controlled UDP bursts, rate-based transport protocols and even emerging standards such as RDMA over Converged Ethernet (RoCE). We can potentially utilize any of a family of transports and link layer protocols in order to minimize overhead and maximize channel utilization. The dedicated networks in question could be any sort of differentiated service tunnel, or optical link such as those in passive optical networks [5] (PONs), the knowledge



SESS ID	OFFSET	LENGTH	CRC
s1	**0**	**4**	0x6BEF
s2	**4**	**9**	0x863B
s3	**13**	**3**	0xD329

Fig. 1. Session layer formation of slabs where control information is signaled out of band. Multiple SPDUs are multiplexed into a single data burst for optimized transmission.

of which allows us to optimize the protocol and link usage. Even, in fact, a shared link with policy such that non-responsive bursts of data would not be disruptive to other service. SLaBS may be particularly well-suited within emerging long-reach PONs [6] (LR-PONs) where we are in a position to enable coarse-grained scheduling of multiplexed data bursts over passively switched, high-capacity optical paths.

Our architecture enables access to resources and services via XSP, which is based on the concept of a session. An XSP session contains the context for the end-to-end connection and provides for key protocol issues like E2E integrity assurance. The session context also manages explicit communication with, and among, the session gateways. An XSP-aware client (or proxy) communicates with the ingress session gateway and begins streaming data. This takes the form of a special-case SPDU that is of unspecified length, and which is essentially delimited by an eventual closing of the transport connection. This data is marshaled into slabs, or SPDUs, for that session. In Phoebus, gateways maintain a connection for each of the sessions, and forward or stream SPDUs. In the SLaBS model, complete SPDUs are multiplexed into a session connection dedicated for bursting. Again, these bursts may contain SPDUs from one or more E2E sessions. These bursts are formed explicitly by SLaBS gateways as opposed to reactive, opportunistic bursting, as found in most previous burst models.

There is an inherent relationship between the amount of buffering required and the provisioning latency of the network. To effectively saturate an optical network core with significant provisioning latency, there is a need for substantial buffering capability at the SLaBS node. However, there is no reason why a given SLaBS node needs to be a single physical host. Both for capacity and redundancy, we can imagine these entities as clusters of slab elements. In this model, we still envision the bandwidth of a dedicated link as being time-division multiplexed, as it is our assertion that the network-level utilization is easier to achieve with N elements sending at their maximum rate for 1/Nth of a time slot than with all elements trying to use 1/Nth of the bandwidth of a given channel. We will investigate a simple token-based protocol to enable this sort of cluster-based burst collusion.

IV. Implementation

The SLaBS model forms data bursts for three primary reasons: (i) to better schedule and optimize transmission of slabs over dedicated network resources, (ii) to reduce protocol overhead, and (iii) to hide provisioning latency incurred during dynamic network resource allocation. In order to realize these optimizations, a mechanism is required to coalesce multiple buffered SPDUs arriving at the SLaBS gateway into larger slab units of data, and efficiently burst them over the network core. We call this technique for forming session-layer PDUs “slabbing”, which is enabled via XSP and implemented within the SLaBS gateway. The following describes this mechanism and provides an overview of our prototype SLaBS implementation.

Our representative data transfer model is one of transparently maximizing network utilization for applications through a series of gateways within the network. Our previous work with Phoebus provides such an architecture with gateways speaking XSP. The SLaBS module described here extends Phoebus with adaptive SPDU buffering and slabbing capabilities along with a separate data channel for efficient slab transmission. Buffers are currently implemented by mapping the underlying buffer space to contiguous regions in virtual memory, requiring a gateway to have a reasonable amount of DRAM and memory bandwidth to be effective.

A SLaBS gateway accepts XSP session requests from end-host applications, or neighboring gateways, which manage one or more underlying transport connections. Each session is identified with an active session ID that associates the established end-to-end path. There is also an open session between gateway peers within the SLaBS network that allows for the exchange of per-flow session state and the signaling of slab transfer control information. Our prototype implementation uses a TCP connection to implement this control channel. Once the ingress SLaBS buffers fill and there is sufficient data, a burst may be formed and transferred to the next intermediate or egress gateway. Figure 1 illustrates this mechanism. As SPDUs are buffered at the SLaBS gateway, they are multiplexed into a larger SPDU during burst formation. Thus, each burst may contain one or more SPDUs and is itself an SPDU. For example, sessions *s1*, *s2*, and *s3* have SPDUs of length 4, 9, and 3, respectively, which are then multiplexed within the newly formed slab SPDU with a length of 16. We employ a simple strategy for selecting buffered SPDU lengths that fairly distributes the SPDUs within the slab based on their current buffer capacity. This selection may be further optimized based on knowledge of the network path and information about the rate at which the buffers fill.

Before the slab is transmitted, control information is sent to enable demultiplexing of the SPDUs at the egress SLaBS gateway. We have defined a slab control SPDU within XSP that contains information about the slab being transmitted, including the number and total length of all multiplexed SPDUs within the slab and an associated record for each. The slab record identifies the session ID of the SPDU and describes the SPDU slab offset, the length of the SPDU, and a CRC field for data error correction at the SPDU level of granularity. This control information is signaled out-of-band via XSP leaving the data channel free to transmit the slab without additional overhead. An acknowledgement is sent back over the control channel once a slab has been successfully received along with any information about slab errors that may have occurred during transmission.

One of our goals is to evaluate SLaBS performance when gateways are connected via high-bandwidth, high-latency dedicated links, requiring a data channel that can efficiently burst slabs from one gateway to another. We investigated and eliminated TCP as a possibility for this application due to the known performance issues over large BDP networks as previously described. User space protocols such as UDT[7] help in some cases but suffer from costly overhead with a level of reliability that may not be necessary for dedicated links. Our prototype SLaBS implementation uses standard UDP over the data channel in order to better approximate the performance of lower-layer approaches in a testbed environment. Our preliminary results indicate that RDMA-based protocols supported with XSP signaling will provide further performance improvements to our SLaBS bursting model.

V. Experimental Evaluation

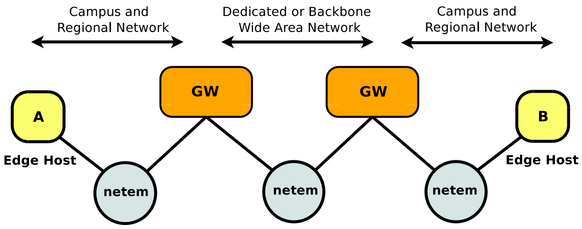
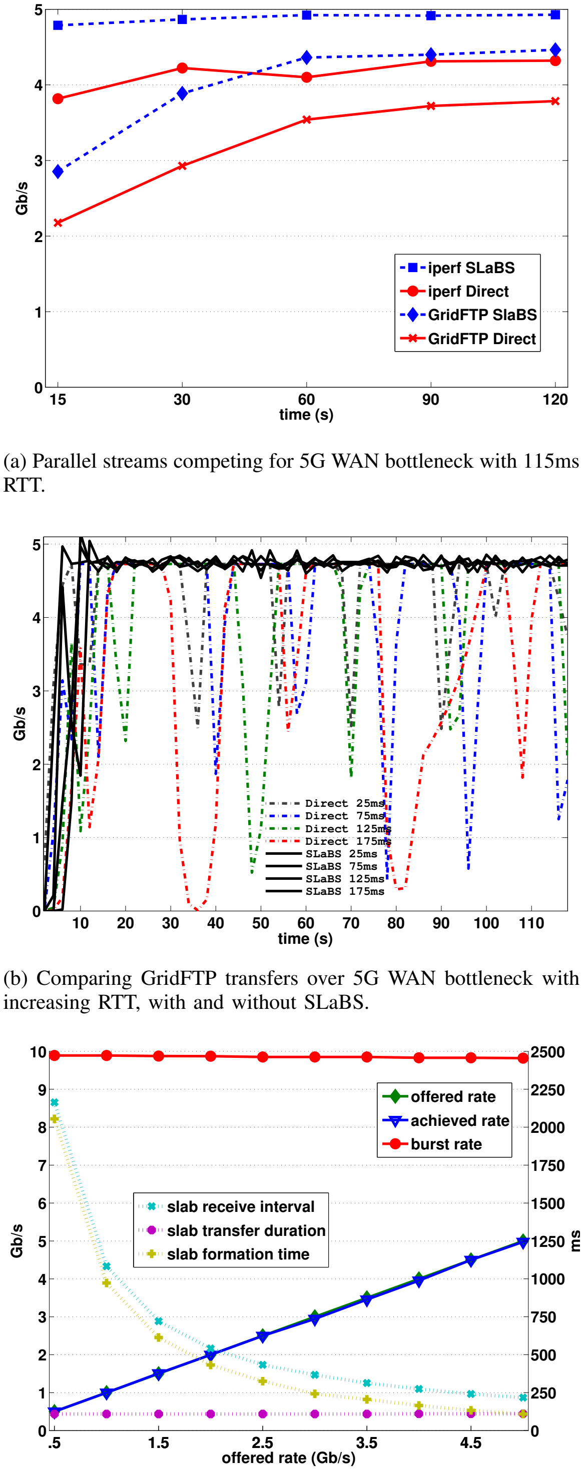


Fig. 2. 10G Experimental Testbed with two SLaBS Gateways.

Here we present some early experimental results of our SLaBS prototype. We show the performance of the popular file transfer tool GridFTP[8] and contrast this with benchmarks obtained using *iperf* [3]. All results were collected from a testbed environment consisting of 7 nodes connected with 10Gb/s Myricom Ethernet NICs, each node having two 10Gb/s interfaces. The testbed forms a linear network topology with client and server nodes at the edges (*A* and *B*), two SLaBS gateways in the middle (*GW*), and 3 delay and rate limiting nodes (*netem*) segmenting the network into representative LAN and WAN segments as shown in Figure 2. The edge host and netem nodes were Sun X2200 servers with quad-core AMD Opteron CPUs and 4GB of RAM, while the gateway systems contained AMD Phenom II X4 processors and included 8GB of high-speed DDR2 RAM.

In order to simulate realistic network environments, we required a way to introduce both delay and bottleneck (rate-limited) conditions along various segments in our testbed. We utilized the *netem* Linux kernel module [9] to induce delay on both the LAN and WAN segments. In order to create bottlenecks and control the edge sending rate, we took advantage of PSpacer [10], which enables configurable and precise network bandwidth control for 10Gb Ethernet. This testbed setup allowed us to evaluate direct end-to-end connections as well as connections over SLaBS using identical paths, guaranteeing the same network conditions.

We applied standard TCP tuning to each system to ensure that connections would not be buffer limited, and the default CUBIC TCP congestion control algorithm in Linux kernel 2.6.26 was used in each of these experiments. The default UDP recv buffer size was increased to 128MB to avoid dropped datagrams due to concurrent SLaBS processing on the gateway systems. Additionally, appropriate NIC driver tuning was performed to ensure that the network hardware itself did not impose any artificial limitations on performance. We also avoided local filesystem bottlenecks by performing memory-to-memory transfers for each experiment.



(a) Parallel streams competing for 5G WAN bottleneck with 115ms RTT.

(b) Comparing GridFTP transfers over 5G WAN bottleneck with increasing RTT, with and without SLaBS.

(c) Bursting performance with a 128MB SPDU "slab" size.

Fig. 3. SLaBS Performance Measurements

A. Preliminary Results

Our first experiment tested our hypothesis that SLaBS would improve performance for a number of streams competing for a

[3] Popular network measurement tool – see http://iperf.sourceforge.net

fixed amount of bandwidth over a shared network path. To that end, we created a 5Gb/s bottleneck with 115ms latency (RTT) on the WAN segment between SLaBS gateways and simulated typical regional network latency with 10ms RTT on the edge links. We ran both GridFTP and *iperf* tests over the direct path and enabled SLaBS transfers with the Phoebus XIO driver for GridFTP and the transparent XSP wrapper library for *iperf*, respectively. We used 4 parallel streams for each run to allow each stream to compete for the available bottleneck bandwidth. Figure 3a shows the final average rate over the total transfer duration for these 4 cases. Each data point is the average of 5 identical runs.

It is clear that transfers using SLaBS show substantial gains over the direct case. By virtue of having very little overhead, *iperf* nearly saturates the 5G bottleneck with the help of SLaBS bursting and achieves a 14% improvement over the direct *iperf* transfer after 120 seconds. The GridFTP transfers do not perform as aggressively, but we see an even larger improvement of 18% with SLaBS enabled after 120 seconds. The benefits with SLaBS are even greater when considering short transfers due to SLaBS avoiding the ramp-up of TCP connections over the high-latency WAN link. After 120 seconds, however, the direct transfer connections have leveled off and the average rate is limited purely by TCP dynamics. SLaBS avoids this through buffering and bursting, removing protocol overhead at the ingress gateway via slab formation, and by optimally transferring slabs with UDP at the WAN bottleneck rate. The slab SPDU size was set to 32MB in this experiment.

Our second experiment looks at SLaBS performance over various WAN latencies. Figure 3b compares 120 second GridFTP transfers with and without SLaBS for WAN latencies of 25, 75, 125, and 175ms RTT. The setup is identical to the 120 second tests above, but here we chart the “instantaneous rate” reported by the GridFTP client on a 2 second interval for a single transfer. Due to the UDP data channel, SLaBS is able to maintain consistent slab bursting performance over the WAN link while keeping the edge rate constant even as the RTT increases substantially, nearly fully utilizing the bottleneck link in each case. In contrast, TCP suffers from congestive loss as competing streams try to maximize their utilization of the 5G link. Here we see the direct transfer case experiencing increasing periods of reduced throughput and significantly longer recovery times as the latency increases. We also noticed apparent TCP timeouts for the highest latency direct cases. We believe this is due to the known limitations of the TCP SACK Linux implementation where the combination of large buffers (>20MB) and high BDP paths cause TCP timeouts when a SACKed packet cannot be located within the sender’s buffer in sufficient time.

Finally, we investigated slab bursting performance at 10Gb/s and analyzed bursting intervals for various offered loads at the ingress GW. Figure 3c shows the ability of SLaBS to consistently send 128MB slab SPDUs at 10Gb/s speeds while maintaining the maximum performance to the edge as we increase the sending rate in 500Mb/s increments up to 5Gb/s. This figure also charts the slab formation time and receive interval associated with each increase in the sending rate. At 10Gb/s, a 128MB slab takes approximately 109ms to transmit over the WAN, and the slab receiving interval is simply the sum of the transfer time and the time it takes to buffer and form a slab SPDU at the ingress SLaBS gateway. For relatively small offered load, the SLaBS model allows for a considerable time delay between slab transmission. As the formation time exceeds the transfer time, slabs are sent as soon as they are formed and must be pipelined over the WAN link to handle further increases in offered load. As we begin to support large-scale data movement with slab SPDU lenths of 1GB and beyond, we expect to take advantage of these longer slab burst intervals. A core network with multiple SLaBS gateways may maximize utilization by optimally scheduling slab transfers within these coarse-grained time slots.

VI. Related Work

While the *SLaBS* model described in this paper breaks new ground, the basic principles behind buffering and bursting are certainly not new concepts and have been proposed for a variety of network technologies. In the early 1980’s, Amstutz [4] takes advantage of the bursty nature of voice “talk spurts” and data messages in order to dedicate transmission channels only when needed, thereby improving efficiency in telecommunication switches. Protocol optimizations such as Nagle’s algorithm [11] are guided by similar ideas at the transport layer. The Delay Tolerant Networking Research Group (DTNRG) [2], with its associated Bundle Protocol [12], attempts to solve the problem of message delivery and routing over challenging network environments, including very large delay transmission and potentially frequently disconnected network paths. Recently, a “session layer” for DTN [13] has been proposed that will allow receiver-driven applications to manage relationships between individual “bundles” of data. SLaBS shares common themes with the bundling approach through slabbing, but we do so at a much coarser granularity while targeting high-performance network environments.

Mills et al. set the stage for optical burst switching (OBS) with an architecture called Highball [14] and associated scheduling algorithms [15]. Their early work outlines architectural considerations for a wide area network that reserves access ahead of time (or “just-in-time”) for bursts of data staged at the edge of the network. This proposed architecture did not include buffering within the network as is the case with SLaBS, but the possibility of staging data bursts in node controllers was considered. The proposed reservation-time-division multiple access protocol (R-TDMA) for use in configuring crossbar switches ahead of the data burst parallels our proposed slab scheduling approaches between session layer gateways.

More recently, OBS advances [16] manage bursty Internet traffic and guide the design of the next generation Optical Internet with IP over WDM. Control and management techniques for OBS have been proposed in recent years [17], [18]; however, these approaches are geared towards specific optical switching hardware whereas SLaBS applies similar concepts via a general session-layer protocol for use over existing heterogeneous networks.

The pervading opinion regarding a solution to network performance issues is that new or improved end-to-end transport protocols are needed. As the dominant Transport protocol in the Internet, a great deal of effort has gone into modifying TCP’s congestion control algorithm. FAST TCP [19] has demonstrated good performance in long distance, high performance networks, but is not available since its commercialization. There are additional approaches (HighSpeed TCP [20], Hamilton TCP [21], Vegas TCP [22], and others too numerous to cite here) which postulate that modifications to TCP’s congestion control model will yield positive results. While many of these approaches can dramatically improve performance over high bandwidth-delay product (BDP) paths, they simply do not offer a general solution for high performance data movement when operating over “hybrid” networks with both shared and dedicated network resources.

Other clearly related work involves dynamic network resource allocation. Systems like Terapaths [23], LambdaStation [24], and VINCI [25] are all approaches related to the dynamic network environment approach, which also use the OSCARS system [26] for dynamic network provisioning. In addition, the DRAGON [27] system is used to provision resources within Internet2's ION [28] network. These groups are collaborating and as their approaches move closer together, SLaBS will be in a position to make use of their advances.

VII. Conclusion and Future Work

We have introduced Session Layer Burst Switching (SLaBS) as a model for optimizing network performance through effective buffer management and framing techniques enabled by a general session-layer protocol. By forming right-sized bursts, or "slabs", at buffering gateways within the network, we enable the optimal transmission of these slabs over dedicated network links with minimal overhead. Our preliminary performance results have shown a significant improvement in network utilization compared to direct transfers through the use of our prototype SLaBS implementation.

As future work, we will build upon our existing implementation to increase both performance and stability through more efficient buffer management and protocol signaling techniques. A number of larger questions concerning optimal slab formation strategies and scheduling approaches must be answered via simulation and further empirical evaluation. In addition, we plan to investigate the role RDMA and link-layer protocols can play in optimizing slab transmission over dedicated WAN network links with no loss. The result of this work will ensure that the SLaBS model can scale effectively as a service in both national and global network environments.

References

[1] E. Kissel, M. Swany, and A. Brown, "Phoebus: A system for high throughput data movement," *Journal of Parallel and Distributed Computing*, vol. In Press, Corrected Proof, pp. –, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6WKJ-50X2NF0-2/2/cd398ec15b98b38b3ddbfa44f71616a1

[2] V. G. Cerf, S. C. Burleigh, A. J. Hooke, L. Torgerson, R. C. Durst, K. L. Scott, K. Fall, and H. S. Weiss, "Rfc 4838: Delay-tolerant network architecture," http://www.ietf.org/rfc/rfc4838.txt, April 2007.

[3] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," Computer Communications Review, 27(3), July 1997., 1997.

[4] S. Amstutz, "Burst Switching–An Introduction," *Communications Magazine, IEEE*, vol. 21, no. 8, 1983.

[5] G. Kramer and G. Pesavento, "Ethernet passive optical network (epon): building a next-generation optical access network," *Communications magazine, IEEE*, vol. 40, no. 2, pp. 66–73, 2002.

[6] H. Song, B.-W. Kim, and B. Mukherjee, "Long-reach optical access networks: A survey of research challenges, demonstrations, and bandwidth assignment mechanisms," *Communications Surveys and Tutorials, IEEE*, vol. 12, no. 1, 2010.

[7] Y. Gu and R. Grossman, "Udt: Udp-based data transfer for high-speed wide area networks," *Computer Networks (Elsevier) Volume 51, Issue 7.*, 2007.

[8] "GridFTP,"
http://www.globus.org/datagrid/gridftp.html.

[9] "Net:netem," http://www.linux-foundation.org/en/Net:Netem.

[10] "PSpacer," http://www.gridmpi.org/pspacer/index.en.jsp.

[11] J. Nagle, "RFC 896: Congestion control in IP/TCP internetworks," Jan. 1984, status: UNKNOWN. [Online]. Available: ftp://ftp.internic.net/rfc/rfc896.txt,ftp://ftp.math.utah.edu/pub/rfc/rfc896.txt

[12] R. Wang, X. Wu, T. Wang, and T. Taleb, "Experimental evaluation of delay tolerant networking (dtn) protocols for long-delay cislunar communications," in *GLOBECOM'09: Proceedings of the 28th IEEE conference on Global telecommunications*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 3497–3501.

[13] M. Demmer and K. Fall, "The design and implementation of a session layer for delay-tolerant networks," *Comput. Commun.*, vol. 32, no. 16, pp. 1724–1730, 2009.

[14] D. L. Mills, C. G. Boncelet, J. G. Elias, P. A. Schragger, and A. W. Jackson, "Highball: a high speed, reserved-access, wide area network," Tech. Rep. 90-9-1, Electronic Engineering Department, University of Delaware, September 1990.

[15] P. Schragger, "Scheduling algorithms for burst reservations on wide area high speed networks," in *INFOCOM*, 1991, pp. 589–596.

[16] C. Y. M. Qiao, "Optical Burst switching (obs) - a new paradigm for an optical internet," *Journal of High Speed Networks*, vol. 8, pp. 69–84, 1999.

[17] J. J. P. C. Rodrigues and M. M. Freire, "Performance assessment of enhanced just-in-time protocol in obs networks taking into account control packet process ing and optical switch configuration times," in *AINAW '08: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 434–439.

[18] N. M. Garcia, J. R. Santos, M. M. Freire, and P. P. Monteiro, "A new architecture for optical burst switched networks based on a common control channel," in *ICNICONSMCL '06: Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies*. Washington, DC, USA: IEEE Computer Society, 2006, p. 110.

[19] D. Wei, C. Jin, and S. Low, "Fast TCP: motivation, architecture, algorithms, performance," in *Proceedings of IEEE Infocom*, 2004.

[20] S. Floyd, "HighSpeed TCP for large congestion windows," Internet Engineering Task Force, INTERNET-DRAFT, draft-ietf-tsvwg-highspeed-01.txt, 2003.

[21] D. Leith and R. Shorten, "H-TCP: TCP congestion control for high bandwidth-delay product paths," Internet Engineering Task Force, INTERNET-DRAFT, draft-leith-tcp-htcp-00.txt, 2005.

[22] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP vegas: New techniques for congestion detection and avoidance," in *SIGCOMM*, 1994, pp. 24–35.

[23] B. Gibbard, D. Katramatos, and D. Yu, "Terapaths: A qos-enabled collaborative data sharing infrastructure for peta-scale computing research," in *Proceedings of the 3rd Intenational Conference on Broadband Communications (IEEE)*, 2006.

[24] A. Bobyshev, M. Crawford, P. DeMar, V. Grigaliunas, M. Grigoriev, A. Moibenko, D. Petravick, and R. Rechenmacher, "Lambda station: On-demand flow based routing for data intensive grid applications over multitopology networks," in *Proceedings of the 3rd Intenational Conference on Broadband Communications (IEEE)*, 2006.

[25] "VINCI: Virtual Intelligent Networks for Computing Infrastructures," http://monalisa.caltech.edu/.

[26] "ESnet On-demand Secure Circuits and Advance Reservation System (OSCARS)," http://www.es.net/oscars/.

[27] T. Lehman, X. Yang, C. P. Guok, N. S. V. Rao, A. Lake, J. Vollbrecht, and N. Ghani, "Control plane architecture and design considerations for multi-service, multi-layer, multi-domain hybrid networks," in *High Speed Networking Workshop, INFOCOM 2007*, May 2007.

[28] "Internet2 ION," http://www.internet2.edu/ion/.