LambdaBridge: Towards an Edge-Based Terabit Wide-Area Network

Xi Wang, Venkatram Vishwanath, Jason Leigh
Electronic Visualization Laboratory
Department of Computer Science
University of Illinois at Chicago
xiwang@uic.edu, venkat@evl.uic.edu, spiff@uic.edu

***Abstract*—LambdaGrid applications are likely to be the first users of terabit-level networking. The challenge of terabit networking will be how to manage traffic at the edges. This paper provides insight into the traffic characteristics of future terabit applications, and uses this to drive the design of an edge-based architecture called LambdaBridge. LambdaBridge will provision and control predictable-performance networks for end-systems (i.e. Grid clusters) using on-demand lambda/VLAN provisioning and end-system traffic shaping. The chief contribution of LambdaBridge is a deep understanding of how to synergistically bridge provisionable networks, end-systems, and future generation distributed terabit applications.**

Keywords-LambdaGrid, Grid Computing, Optical Networks, Flow Management, Terabit Applications

I. INTRODUCTION

Interactive exploration of multi-terabyte datasets has been identified as a critical enabler for scientists to glean new insights in a variety of disciplines, such as biomedical imaging, geoscience and high-energy physics [1-3]. Practically, these large-scale datasets must flow among a Grid of instruments, physical storage devices, visualization displays, and computational clusters. These applications have a real need for tens to hundreds of gigabits-per-second of bandwidth that are best satisfied by interconnecting Grid resources with dedicated networks dynamically created by concatenating optical lightpaths (lambdas). This is called a LambdaGrid.

LambdaGrid applications can generate hundreds to thousands of parallel flows. These flows emanate from network interfaces in the end-systems (i.e. the compute clusters) to communicate with other end-systems over multiple lightpaths. More complex flows include multiple parallel end-systems, inter-communicating over known, but arbitrary, physical network topologies. These flows impose differing demands on the host's system resources, such as memory, bus bandwidth and CPU. However, as the exponential growth of bandwidth now far exceeds storage and computing, a significant impedance mismatch exists between these high-capacity lambda-based networks and the end-systems that must absorb the bandwidth, resulting in inadequately performing applications. At the recent LCA06 (Linux Conference Australia 2006) conference, keynote speaker Van Jacobson resonated a similar sentiment "The end of the wire isn't the end of the net," – that is, the future challenges of high-performance networking reside at the edges [4]. While much prior work has focused on Quality of Service for the networks, this has not been the case within the edge devices, most notably the computers that must send, receive and process the network payload. On-demand networks built with optical technologies allow significantly better bounds on "competing" traffic and can therefore enable more-aggressive transmission protocols. However, parallel endpoints still face contention and resource sharing issues.

This paper anticipates the future needs of LambdaGrid applications by addressing key issues toward enabling Terabit-per-second network flows. These issues include: how to affordably and practically terminate hundreds of gigabits of bandwidth at the edges while minimizing the penalties associated with optical-to-electronic translation; how to efficiently manage the myriad parallel data flows among and within the end-systems; and, does treating these parallel communication channels as a single problem rather than as entirely uncoordinated flows allow either better utilization or more predictable performance?

We examine these issues and present an edge-centered architecture called LambdaBridge. In section II we present the requirements of future LambdaGrid applications, specifically focusing on distributed real-time collaborative visualization. In section III we give an overview of the LambdaBridge architecture which bridges terabit applications with the optical core networks. The two research programs: a traffic-to-lambda mapping scheme called *LambdaBridging*, and an end-system & network resource-aware flow management mechanism called *Synergistic Flow Framework* are elucidated in section IV and V respectively.

II. REQUIREMENTS OF LAMBDAGRID APPLICATIONS

LambdaGrid applications mainly operate on large numbers of computer nodes. Increasingly, traditional Grid systems are evolving into LambdaGrids by connecting them to dynamically configurable lambda-based networks. In this work, we target ultra-high-bandwidth collaborative visualization applications being developed by members of the Global Lambda Visualization Facility (GLVF) [2]. Networked visualization is chosen because it contains a more comprehensive variety of flow types than any other category of LambdaGrid application.

Interactive ultra-high-resolution Grid visualization applications routinely create hundreds of bidirectional streams between distant endpoints, each with differing flow requirements operating over differing transport protocols. Table I quantifies the broad variety of flows that simultaneously emanate from SAGE (Scalable Adaptive Graphics Environment) − a system to manage remotely rendered visualizations for viewing on tiled displays [5]. A single visualization rendering on a cluster of 8 computers streaming to a remote tiled display of 55 tiles (driven by 28 computers) will create as many as 96 visualization flows, 36 control flows, and 34 synchronization flows for a total of 166 flows. In a collaborative session of 3 remote sites sharing 3 visualizations, the number of flows could reach as high as 1500; and all of these flows can compete simultaneously for networking, memory, system bus, and CPU resources. These flows must be synergistically coordinated and intelligently mapped to the available resources so that the desired end-to-end performance is achieved. Currently this is managed by intuition on a configuration-by-configuration basis. Needed instead is an automated means to provide systemic quality of service. Explicitly coordinated, resource-aware network flows will lead to more predictable performance and enable both uniform and non-uniform distribution of network resources among the parallel endpoints.

III. LAMBDABRIDGE ARCHITECTURE

Driven by the insatiable and diversified traffic demands made by LambdaGrid applications, we present a LambdaBridge architecture enabling applications to more efficiently access lambda-based networks. Whereas much of the work thus far has focused on creating and provisioning the core network infrastructure, LambdaBridge focuses on a much-needed edge-based strategy to "bridge" applications on future terabit wide-area networks. The LambdaBridge architecture will provision and control predictable-performance networks for clustered endpoints using on-demand lambda/VLAN provisioning and endpoint traffic shaping. This will be realized using *LambdaBridging* and *Synergistic Flow Framework*.

- **LambdaBridging** investigates how to enable bridging points between end nodes and core networks to map parallel application data flows into parallel lambda paths so that both efficient data transport and optimized lambda utilization can be achieved.
- **Synergistic Flow Framework** investigates how to enable applications running on parallel computer nodes to generate/receive coordinated flows so that the resource of both networks and computer nodes can be shared and exploited efficiently.

There is much related prior work in this area: coordinated congestion control [6-8]; explicit feedback-based control, such as those in XCP, TeXCP, ECN and NetX; adaptation of real-time (non-commodity) systems for application performance [9-10]; feedback-based application adaptation on commodity end-systems (such as Active Harmony, Prophesy); provisionable user-controlled networks (such as UCLP, DRAGON, CHEETAH, OMNInet, EnLIGHTened, OptIPuter); network Quality of Service [11-12]; high-performance transport protocols [13-17]. The chief contribution of LambdaBridge is a deep understanding of how to synergistically bridge provisionable networks, end-systems, and future-generation distributed terabit applications.

Figure 1 envisions a scenario of LambdaBridge-empowered LambdaGrid computing. Two applications ("A" and "B") run on cluster nodes at three sites: UIC/Chicago, UCSD/San Diego and UvA/Amsterdam. These sites are interconnected via national and international optical links. LambdaBridge will manage and adapt all flows of each application and provide site-to-site lambda connections for them. To seamlessly support these functions, the LambdaBridge architecture will necessarily consist of the following key components: *SYNOPTIC*, *Synergistic End Node Controller*, *LambdaBridge Controller*, and *LambdaBridge Hardware*.

- *SYNOPTIC* is a high-performance, application-level protocol framework. It provides applications with the ability to compose desired flow characteristics and the power to express the relationship between its many flows. It communicates with SYNOPTICs on every other remote computer node with whom this node exchanges flows (end-to-end control).
- *Synergistic End Node Controller (SENC)* resides on each computer node. It elicits the network condition

TABLE I. NETWORK FLOWS CREATED BY AN ULTRA-HIGH-RESOLUTION GRID VISUALIZATION APPLICATION

Type of Flow	Number of Flows	Bandwidth per Flow	Latency Sensitive	Jitter Sensitive	Reliability Requirement	Burstiness	Message Size	Protocol
Audio Stream	1 per user	Low 1Mbps	Yes	Yes	Med	Constant	Small	UDP-based
HD Video Stream	1 per user	Med to High 25Mbps-1.5Gbps	Yes	Yes	Med	Constant	Small to Med	UDP-based
Application Stream	1-100 per application	High 1-2.5Gbps	Yes	Variable	High	Application Dependent	Large	UDP-based
Bulk Data	1 per render node	High	No	No	High	Application Dependent	Large	UDP/TCP-based
Annotations/ Static Content	1-10 per user	Low 1Mbps	No	No	High	One Burst	Small	TCP-based
Control Channel	1 per rendering node + 1 per display	Low 64Kbps	No	Yes	High	Short Burst	Small	TCP-based
Synchronization Channel	1 per rendering node + 1 per display	Low 1Mbps	Yes	Yes	High	Constant	Small	TCP-based
SAGE UI	1 per user	Low 64Kbps	No	No	High	Short Burst	Small	TCP-based
VNC Streams	1 per user	Low 1Mbps	Yes	Yes	High	Small Burst	Small	TCP-based

from the LambdaBridge Controller. It provides network and end-system-aware flow management for all SYNOPTIC flows sourcing and sinking at a node.

- The *LambdaBridge Controller*, present at each site, provides traffic management for all application traffic going in and out the site. It interacts with every local Synergistic End Node Controller to adapt the rate of all flows in its site. It maps flows into site-to-site traffic demands and then maps them onto lambda connections. It interacts with Optical Network C-plane (Optical Network Controller in the figure) for lightpath provisioning. It also controls LambdaBridge Hardware for appropriate traffic forwarding/switching.
- *LambdaBridge Hardware* is the physical device that provides traffic-to-lambda bridging for application data. It gathers traffic from all the computer nodes of the local site and maps it to multiple lambdas connected to different remote sites. It can be realized using a number of technologies, including Layer 1/2/3 devices, or a combination of them.

IV. LAMBDABRIDGING

A LambdaBridge consists of a software component- the LambdaBridge Controller; and a hardware component- the LambdaBridge Hardware.

A. LambdaBridge Controller

LambdaBridge Controller manages all traffic of its local site and provides lambda connections for them. It exchanges site information with other LambdaBridge Controllers through the Control Plane (C-plane) so that each LambdaBridge Controller knows the list of Grid clusters that are connected to other LambdaBridge Controllers in the network. The LambdaBridge Controller consists of three functional modules: a *Synergistic Network Controller*, a *VLAN Provisioner* and a *Lambda Provisioner*.

Synergistic Network Controller consists of a collection of

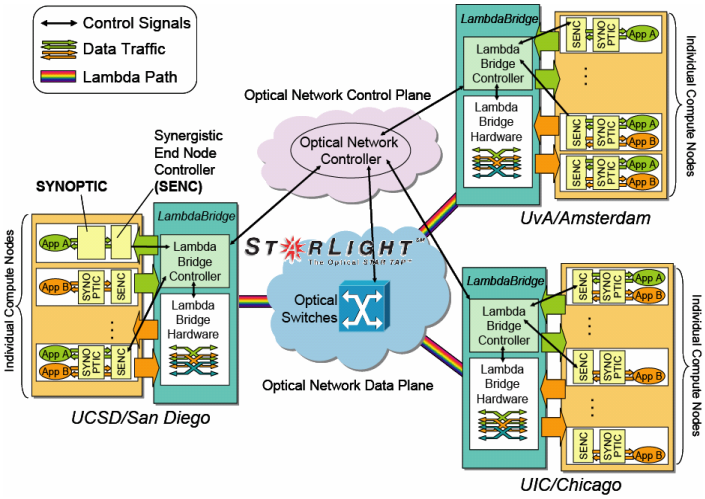

Figure 1. The LambdaBridge Testbed

Synergistic Group Controllers (SGCs) and an Inter-group Coordinator (IGC). SGC is responsible for all the flows of an application at a given site. There is a SGC controlling each application at each site. The SGC receives requests for bandwidth, jitter, etc., from all the flows of an application. The communication between SGCs and SENCs is realized via signaling protocols, such as RSVP and SBM. Each SGC clusters the requests into site-to-site (LambdaBridge-to-LambdaBridge) traffic demands according to the destination. These traffic demands will be passed on to the VLAN Provisioner for network resource allocation. The IGC manages, optimizes and coordinates traffic demands of multiple SGCs to enforce efficient resource sharing among applications. It advises each SGC on how it must adapt its bandwidth usage based on overall network conditions. Then, each SGC interacts with corresponding Synergistic End Node Controllers to adapt flows. The flow adaptation and optimization mechanism will be explained in detail in the next section.

VLAN Provisioner provides VLAN connections for site-to-site traffic demands. It manages a Connection Table containing current lambda connections and their VLAN setups. Each traffic demand is managed as a VLAN entity on a particular lambda connection. The VLAN-to-lambda mapping can be optimized using a channel allocation algorithm that seeks to satisfy QoS requirements of individual traffic demand, while maximizing overall lambda usage. For each new traffic demand, the VLAN Provisioner checks the Connection Table to see if there is an available lambda connection(s) to the remote site LambdaBridge. If YES, it sets up a VLAN and reserves the required bandwidth on it (them). If the requested bandwidth exceeds the available bandwidth of a single lambda, a group of VLANs on multiple lambdas is provided for the traffic demand. Note: The VLAN Provisioner does not actually reserve bandwidth; it only keeps track of the intended bandwidth usage of each VLAN (bandwidth control is performed by the Synergistic Flow Framework (SFF), which will be explained in the next section.) However, the VLAN Provisioner may incorporate a traffic monitoring mechanism and provide feedback to SFF. If there is no existing connection to the remote site or the total remaining bandwidth of existing connection(s) to it is not enough to accept the new traffic demand, the VLAN Provisioner will ask Lambda Provisioner for new or additional lambda connections.

Lambda Provisioner provides automatic lambda provisioning for site-to-site connections. It communicates with the optical network C-Plane to request lightpath setup, teardown or reconfiguration. It reports information about lambda connections (such as available bandwidth and duration) to the VLAN Provisioner for Connection Table update. Lambda provisioning can be triggered by requests from VLAN Provisioner for integrated VLAN/lambda configuration, or by requests from the optical network C-plane for domain-wide lightpath optimization. The Lambda Provisioner will leverage existing Control-Plane research and standards (such as GMPLS, UCLP, PIN/PDC, DRAGON) for scheduling and provisioning lightpaths. Authorization, authentication and accounting functionalities [18] can be integrated to enable policy and security enforced network resource utilization.

B. LambdaBridge Hardware

LambdaBridge Hardware is a device to bridge application traffic and lambdas. A variety of existing Layer 2/3 commodity products for terminating lambdas at the edges can be architected to build a LambdaBridge with sufficient capacity to scale to a Terabit. Also, it is important to examine the feasibility of alternative novel hardware configurations using Layer 1 technologies, commodity PCs and hybrid configurations to terminate lambdas. In this paper, we discuss only two of these solutions due to space limitation.

L2 LambdaBridge. An approach using today's commodity networking technology is to implement the LambdaBridge using Layer-2 (L2) switches with optical interfaces attached and a controlling service. In this case, LambdaBridging is realized using VLANs. In the LambdaBridge architecture, each site-to-site traffic demand (a group of flows) will incur a series of VLAN configurations. Specifically, all end nodes and LambdaBridges involved need to be assigned common VLAN IDs and/or (virtual) subnet IP addresses. Currently there is no practical way to automatically configure large numbers of VLANs among parallel endpoints connected by parallel paths in accordance with application flows dynamics. An automatic VLAN configuration tool is therefore needed to enable automatic, fast and secure VLAN configuration for both LambdaBridges and end nodes by means of autonomous and unified VLAN ID (and IP subnet addresses for L3 VLANs) assignment and signaling. Various allocation policies including centralized allocation, subset pre-allocation, and peer-to-peer negotiation can be applied. Prior work, such as Rbridges [19] and the IEEE L2 Scaling Enhancement activities (e.g. 802.1ah, 802.1ad, 802.1s), and IETF activities (e.g. [20]) are leveraged to address the L2 scalability-related issues. For scalable LambdaBridge hardware design, L1+L2 hybrid switches used in next-generation high-performance interconnects [21-22] will be implemented.

PC LambdaBridge. A more radical approach would be using PCs that are normally part of a Grid computing cluster, as direct termination points for lambdas, and using the cluster backplanes to switch the packets to their final destinations. We call this configuration the PC LambdaBridge. We conducted an analysis of terminating 1Tb using clusters of PCs [23] with an Infiniband/Myrinet backplane to achieve full bisection bandwidth, compared to a commercially available 1Tb switch, and found that the cost savings could potentially reach 80%. We realize that the PC-based solution is unlikely to perform to the degree of a dedicated switch; however, if the assumption is that lambdas will become cheaper than electronics, then it is not too far fetched to "waste" lambdas to compensate for the performance loss in a PC-based solution. The PC solution also has other advantages; it can bridge different Layer 2 technologies where no commercial devices exist and, as new lightpaths are added, it can utilize more cluster nodes as serve bridge nodes and switch incoming traffic. This approach lets us prototype capabilities that might be useful to include in future-generation L2 switches. A PC LambdaBridge can be realized using PCs with multiple-attached NICs. Both conventional NICs and WDM NICs can be used for lambda connections. WDM NICs are available now; wavelength-convertible WDM NICs are ready to implement and estimated to be available within the next few years. It would be important to evaluate the performance of PC LambdaBridges built with difference system specs (CPU, Memory, etc.) and NICs (such as high-end network processor-based NICs and WDM NICs). PC LambdaBridge also has the potential to take advantage of both L2 forwarding and IP addressing, which can be realized by implementing a PC router integrated with a new kernel driver that bypasses most of a CPU's packet-by-packet forwarding processing. XORP [24] could be a good starting point for such implementation.

V. SYNERGISTIC FLOW FRAMEWORK

In hybrid high-bandwidth networks, the main bottleneck is the commodity-off-the-shelf end-systems (i.e. computers) that are either unable to keep up with incoming packets or source more data than their receiver(s) can handle. The problem goes even deeper, as each sending node typically sends and receives multiple streams simultaneously and applications have expectations about how those streams should behave (illustrated in Table I).

The Synergistic Flow Framework consists of SYNOPTIC, the Synergistic End Node Controller (SENC) (Figure 2) and the Synergistic Network Controller (SNC). Synergy for a LambdaGrid computing application is achieved by:

- The SYNOTPIC protocol framework, which strives to provide systemic Quality of Service for applications by taking network and systems conditions into account. Network conditions are obtained from the SNC, and systems conditions are obtained from the SENC. We define systemic QoS as (1) the ability of an application to compose desired flow characteristics and (2) the ability of the system to deliver those characteristics by negotiating the necessary network (bandwidth, latency, jitter) and system constraints (priority, processor affinity, scheduling heuristics).
- The SENC, which monitors and synthesizes the end-systems' conditions, provides feedback to SYNOPTIC and schedules SYNOPTIC flows.

A. SYNOPTIC

SYNOPTIC is a high-performance, application-level, end-to-end, configurable, composable and extensible protocol

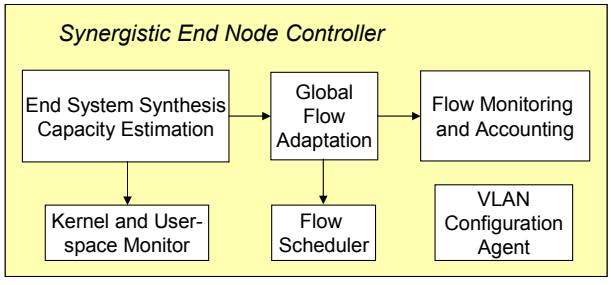

Figure 2. Functional modules of a Synergistic End Node Controller

framework. SYNOPTIC leverages our prior work in Quanta [25] (a cross-platform networking toolkit for supporting the diverse networking requirements of latency-sensitive and bandwidth-intensive applications), and related work in composable protocols [26]. Quanta provides a rich set of features, such as reliable transmission, unreliable transmission, forward error correction, streaming transfer, and reliable bulk-data transfers that are built on top of existing TCP and UDP transport protocols. SYNOPTIC extends Quanta to make it end-system, network-resource and group aware. SYNOPTIC provides applications with the ability to compose desired flow characteristics and the power to express the relationship between its many flows. SYNOPTIC is self-monitoring and provides timely feedback to applications.

SYNOPTIC can take the form of an application-level "native" SYNOPTIC that takes advantage TCP, UDP and DCCP and their native congestion control algorithms or an application-level UDP-based SYNOPTIC. The UDP-based SYNOPTIC has an UDP-based data channel and a control channel that could be a TCP or UDP control channel. The UDP-based SYNOPTIC leverages the configurable congestion control work done in UDT and extends it to express configurable congestion control for a group of flows of a parallel application. SYNOPTIC that takes advantage of explicit signaling-based group coordinated congestion control. A key research issue here is evaluation of explicit signaling-based feedback approaches versus traditional probing based approaches on ultra-high-speed networks. An advantage of an explicit-signaling based scheme over traditional probe-based approaches is that as a physical path is subdivided into multiple VLANs, explicit signaling enforces traffic restrictions over such VLANs over shared physical paths.

B. Synergistic End Node Controller (SENC)

The SENC monitors and synthesizes the end-system's conditions and provides feedback to SYNOPTIC. It adapts and schedules SYNOPTIC flows based on the Network conditions elicited from the Synergistic Group Controller (SGC) and the system conditions of an end-node. The SENC, as shown in Figure 2, consists of:

1) Kernel and User-Space Monitoring (MAGNET)

We are currently developing Monitoring Apparatus for Generic Kernel Event Tracing (MAGNET) v3.0 [27-28] to monitor Linux kernel subsystems, such as memory, I/O, network stack and scheduler. MAGNET leverages the dynamic-probes mechanism available in the Linux kernel and supports adaptive event filtering, event instrumentation and event sampling. This mitigates the overhead by limiting the monitoring to only those parameters that are most relevant to the application. In the future, we plan on extending MAGNET to incorporate user-space probes that can non-intrusively instrument and monitor applications.

2) End-System Performance Synthesis and Prediction (ESPSP)

We are modeling the end-system subsystems and designing a performance model of an end-system based on its load and flow characteristics. We are evaluating load-dependent queuing model [29] and a Discrete-Time Stochastic Control System [30] based on the synthesized feedback and the response latency.

3) End-System Flow Adaptation and Scheduler (ESFAS)

The ESFAS calculates a schedule for all the flows based on system synthesis from the ESPSP and the network feedback from the SGC and schedules the flows according to the computed Inter Packet Gap [31-32]. The flow scheduling can take place at the kernel layer, at an application layer or a hybrid combination of both. The scheduling of flows at the kernel layer can be achieved using IPROUTE2 in Linux. The kernel–based flow scheduling has the advantage of controlling all the flows of the system, whether they are SYNOPTIC or not. Flow scheduling at an application layer via a user-level scheduling daemon makes the scheme more portable and deployable. However this scheme suffers the scheduling and other effects of a normal user-level process in a commodity operating system. This can be mitigated by increasing the priority of the scheduler as a soft real-time process.

4) End-System Flow Monitoring and Accounting.

This is responsible for keeping track of the end-node capacity that has been provisioned and used. This facilitates admission control based on the end-system's capacity, i.e. even if the network can support the flow, we need to ensure that the end-system can support it as well.

C. Synergistic Network Controller (SNC)

As described in section IV, the SNC consists of a collection of SGCs and an IGC. The SGC receives requests from application nodes through SYNOPTIC and SENC regarding network provisioning. This request could range from a best-effort bandwidth request to a network-QoS request that specifies parameters such as bandwidth and latency. SGC aggregates and groups the request according to the destination site and type of traffic that needs to be provisioned. A traffic demand is made in terms of discrete quantas to the VLAN Provisioner, which facilitates traffic optimization and provisioning.

SGC also performs group traffic-to-VLAN bandwidth optimization and combined heuristics with the VLAN-to-Lambda optimization will likely be needed. As a bandwidth request of an application can be potentially satisfied by multiple VLANs based on the traffic provisioning heuristic of the VLAN Provisioner, adequate VLAN address configuration of the end-hosts is needed to satisfy the flow. The SGC informs SENC of the VLAN address to be configured. SENC then advises SYNOPTIC on how a flow should be split and the allowable bandwidth on each VLAN. SYNOPTIC stripes the payload into blocks to reduce the overhead of re-ordering at the receiver. We are investigating adaptive striping of payload (buffers) based on the individual path characteristics.

The SGC enables enforcement, i.e. it informs the SENC about the bandwidth usage of the application flows. The SENCs collectively ensure that the flows of an application do not use more than their provisioned bandwidth. This is especially needed when a lightpath is shared by multiple

applications. The usage of an application can be monitored either by collecting statistics from SENC or by querying the VLAN usage on compliant L2 switches which provide per-VLAN traffic statistics.

The SGC provides SYNOPTIC with an interface to dynamically provision bandwidth. SGC monitors the bandwidth usage of an application and can pre-provision bandwidth for an application before it is needed. This helps in overcoming signaling latency, a key issue associated with typical bandwidth provisioning heuristics.

An application can configure SYNOPTIC to use any rate-based congestion control algorithm to manage the flows of the application as a group. The application could for instance apply TCP-friendly congestion control or any algorithm that achieves Max-Min, proportional fairness criteria, etc.

VI. CONCLUSION

In this paper we presented the LambdaBridge architecture, a scalable and demonstrable approach for enabling LambdaGrid applications to efficiently run on terabit wide-area networks. LambdaBridge will work cohesively with end nodes and core networks to provide network-wide application flow coordination and end-system & network unified resource management, thereby achieving desirable end-to-end performance while maintaining balanced resource utilization. LambdaBridging will also contribute to a deep understanding of what influences the efficiency of data-intensive applications from the point of view of data flows and their interaction with backplane, OS, memory, CPU, system bus, and network elements.

There remain many areas of future investigation. Several key issues include scalable LambdaBridge software and hardware design, end system and network-aware transport protocols, end system performance monitoring, coordinate flow management, efficient flow aggregation and VLAN/lambda provisioning. These issues will be addressed in our future work.

REFERENCES

[1] http://www.igrid2005.org

[2] J. Leigh, et al., "The Global Lambda Visualization Facility: An International Ultra-High-Definition Wide-Area Visualization Collaboratory," *Journal of FGCS*, in press.

[3] L. Smarr, A. Chien, T. DcFanti, J. Leigh and P.Papadopoulos, "The OptIPuter," *Comm. of the ACM*, vol. 46(11), pp. 58-67, Nov. 2003.

[4] V. Jacobson and B. Felderman, "A modest proposal to help speed up & scale up the linux networking stack." *Proc. Linux Conference Australia, (LCA 2006)*, Dunedin, New Zealand, Jan 23-28, 2006.

[5] L. Renambot, et al., "SAGE: the Scalable Adaptive Graphics Environment," *Proc. WACE 2004*, Sept 23-24, 2004.

[6] H. Balakrishnan, H. Rahul, and S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts," *Proc. ACM SIGCOMM*, Cambridge, MA, Sept. 1999.

[7] V. Padmanabhan, "Coordinating Congestion Management and Bandwidth Sharing for Heterogeneous Data Streams," NOSSDAV '99

[8] D. Ott and K. Mayer-Patel, "Aggregate Congestion Control for Distributed Multimedia Applications," *Proc. Infocom 04*.

[9] K. Lakshman, R. Yavatkar and R. Finkel, "Integrated CPU and network-I/O QoS Management in an Endsystem," *Int. Workshop on Quality of Service (IWQoS)*, pp. 167-178, 1997.

[10] K. Nahrstedt, J. Smith, "The QoS Broker," *IEEE MultiMedia*, 1995.

[11] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: An Overview," RFC 1633, June 1994.

[12] L. Breslau, E. Knightly, S. Shenker, I. Stoica, and H. Zhang, "Endpoint Admission Control: Architectural Issues and Performance," *Proc. ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.

[13] I. Rhee and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variants," *Proc. PFLDnet 2005*, Feb. 2005.

[14] P. Mudambi, X. Zheng, M. Veeraraghavan, "A Transport Protocol for dedicated end-to-end circuit," IEEE ICC 2006. (to appear)

[15] Q. Wu, N. S. V. Rao, "Protocol for High-Speed Data Transport Over Dedicated Channels," *PFLDNet 2005*, Lyon, France, Feb 2-3, 2005.

[16] C. Xiong, J. Leigh, E. He, V. Vishwanath, T. Murata, L. Renambot, T. DeFanti, "LambdaStream – a Data Transport Protocol for Streaming Network-intensive Applications over Photonic Networks," *PFLDNet 2005*, Lyon, France, Feb 2-3, 2005.

[17] S. Floyd, "HighSpeed TCP for Large Congestion Windows," RFC 3649, December 2003.

[18] C. de Laat, et al., "Generic AAA Architecture," RFC 2903, Aug. 2000.

[19] R. Perlman, "Rbridges: Transparent Routing," *Proc. Infocom 2004*, March 2004.

[20] D. Papadimitriou, et al., "A Framework for GMPLS-controlled Ethernet Label Switching," <http://www.ietf.org/internet-drafts/draft-dimitri-gels-framework-00.txt>

[21] K. Barker, et al., "On the Feasibility of Optical Circuit Switching for High Performance Computing Systems," *Proc. SC'05*.

[22] J. Shalf, et al., "Analyzing Ultra-Scale Application Communication Requirements for a Reconfigurable Hybrid Interconnect," *Proc. SC'05*.

[23] C. Zhang, V. Viswanath, R. Singh, L. Renambot, J. Leigh, "Comparison of End-Point Routing/Switching (the LambdaRouter) vs Big Fat Routers," EVL technical document, 2005," <http://www.evl.uic.edu/cavern/rg/20050312_zhang/>

[24] http://www.xorp.org/

[25] E. He, J. Alimohideen, J. Eliason, O. Yu, J. Leigh, T. DeFanti, "Quanta: A Toolkit for High Performance Data Delivery," *Journal of FGCS*, vol. 1005, pp. 1–15, 2003.

[26] N.C. Hutchinson and L.L. Peterson, "The x-Kernel: An Architecture for Implementing Network Protocols," *IEEE Transactions on Software Engineering*, vol. 17, no. 1, pp. 64-76, 1991.

[27] V. Vishwanath, M. Gardner, MAGNET v 3.0 – Los Alamos National Laboratory Unclassified Report.

[28] M. Gardner, W. Feng, M. Broxton, A. Engelhart, G. Hurwitz, "MAGNET: A Tool for Debugging, Analysis and Adaptation in Computing Systems," *Proc. CCGrid 2003*, Tokyo, Japan, May 2003.

[29] E. Lazowska, J. Zahorjan, S. Graham, K. Sevcik, Quantitative System Performance: Computer System Analysis using Queuing Models, Prentice-Hall, Inc., 1984.

[30] D. Bertsekas and S. Shreve, "Stochastic Optimal Control−The Discrete Time Case," Athena Scientific, 1996

[31] I. Stoica, S. Shenker and H. Zhang. "Core-stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks," *Proc. ACM SIGCOMM*, Vancouver, CA, August 1998

[32] N. Bansal, A. Blum, S. Chawla and K. Dhamdhere, "Scheduling For Flow-Time with Admission Control (or, How to manage your to-do list)," *European Symposium on Algorithms (ESA 2003)*.