LambdaBridge: A Scalable Architecture for Future Generation Terabit Applications

(Invited Paper)

Xi Wang*, Venkatram Vishwanath, Byungil Jeong, Ratko Jagodic,
Eric He, Luc Renambot, Andrew Johnson, Jason Leigh
Electronic Visualization Laboratory
Department of Computer Science
University of Illinois at Chicago
* xiwang@uic.edu

***Abstract*—LambdaGrid applications as typified by data-intensive collaborative visualization are likely to be the first users of terabit-level networking. This paper features a main enabler of collaborative visualization over LambdaGrid, the Scalable Adaptive Graphics Environment (SAGE) in particular, and anticipates the future needs of LambdaGrid applications in general. We present a scalable architecture called LambdaBridge to 'bridge' LambdaGrid applications with the terabit optical core networks. 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. This paper also introduces Scalable Visualcasting – a multicasting service for LambdaGrid that incorporates IP multicasting and optical multicasting.**

I. INTRODUCTION

In a decade's time, high-performance computing has proven its value in Science, Homeland Security, Medicine, Engineering, Education, and Filmmaking. These data-intensive domains rely on the Grid to process terabytes of raw data to produce meaningful insight. Typically, 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 and deterministic QoS that are best satisfied by interconnecting Grid resources with dedicated networks dynamically created by concatenating optical lightpaths (lambdas). This is called a LambdaGrid [1-4].

One of the pioneering LambdaGrid applications is collaborative visualization which allows users from geographically distant institutions to interactively visualize and analyze the shared data. A fundamental goal of data-intensive collaborative visualization on LambdaGrid is to enable users to collectively interpret large-scale (e.g. multi-terabyte) remote datasets in real-time at extremely high resolutions, thereby dramatically increasing the productivity of data interpretation. We envision that in the future it will become routine for users to work and collaborate in rooms whose walls are made from seamless ultra-high-resolution displays that are fed by data streamed over ultra-high-speed networks from distantly located visualization, storage servers, and high-definition video cameras [3-4]. We have taken the first steps towards this vision by building LambdaVision – a tiled display built from 55 LCD screens with a total resolution of 100 megapixels (see Figure 1). LambdaVision is primarily enabled by the Scalable Adaptive Graphics Environment (SAGE) [5], a middleware system for managing visualization and high-definition video streams that are presented on scalable displays. Specifically, SAGE allows remote groups of users to simultaneously display imagery of multiple remote visualization applications on high-resolution tiled displays, thereby providing an ideal distant collaborative visualization environment with multiple endpoints.

LambdaGrid applications as typified by SAGE applications routinely access remote large-scale datasets and visualize the rendered pixels on high-resolution displays. The network bandwidth requirement is in the range of several tens to hundreds of gigabits per second. In addition to the huge bandwidth usage, these applications usually generate hundreds to thousands of parallel flows. These flows emanate from network interfaces in the endpoints (i.e. the compute clusters) to communicate with other endpoints over multiple lightpaths. More complex flows include multiple parallel endpoints, inter-

Figure 1. LambdaVision driven by SAGE

1-4244-0425-8/06/$20.00 ©2006 IEEE

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 endpoints that must absorb the bandwidth, resulting in inadequately performing applications. At the 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 [6]. 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 endpoints; 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 a scalable architecture called LambdaBridge to support future generation LambdaGrid applications. In section II, we feature data-intensive collaborative visualization - a pioneering LambdaGrid application and its main enabler – the SAGE middleware. After briefly describing the SAGE framework and its typical applications, we present the specific network requirements made by LambdaGrid applications. In section III we give an overview of the LambdaBridge architecture which bridges LambdaGrid 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. Section IV introduces the concept of *Visualcasting* - a specifically designed image multicasting service for ultra-definition visualization. We present several approaches including both IP multicasting and optical multicasting; the LambdaBridge multicast support issue is also discussed. Section V concludes the paper.

II. SCALABLE ADAPTIVE GRAPHICS ENVIRONMENT

A fundamental goal of visualization and collaboration on LambdaGrid is to enable users to collectively interpret enormous data-sets in real-time at extremely high resolutions. An increasingly important model is to conduct the visualization using large pools of computing resources (such as clusters of powerful computers equipped with high-performance graphics processors) and streaming the results to the collaborating end-points. These end-points may range from PDAs all the way up to ultra-high-resolution display walls such as those built from stitching together dozens of LCD panels. The image streams shown on these display devices may consist of offline rendered movies as well as real-time visualizations, and high-definition video. This approach provides significant advantages: firstly the pooling of computing resources increases utilization, especially when they are cast as Grid services that can be combined with other services to form a pipeline that could link large-scale data sources with visualization resources. Secondly, since networking is diminishing in cost at a rate exceeding that of computing and storage, it becomes more cost-effective for users to build low-cost, networked thin-clients than to have to purchase and maintain their own rendering farms, storage repositories, etc.

We developed the Scalable Adaptive Graphics Environment (SAGE) to put this model into practice. SAGE is specialized middleware for enabling data, high-definition video and extremely high-resolution graphics to be streamed in real-time from remotely distributed rendering and storage clusters to scalable displays over ultra high-speed networks. Each visualization application (such as real-time or offline rendered visualizations, remote desktop, high-definition video streams, 2D maps etc.) streams its rendered pixels (or graphics primitives) to the virtual high-resolution frame buffer of SAGE, allowing user-definable window position and size on the displays (e.g. the output of arbitrary M by N pixel rendering cluster nodes can be streamed to X by Y pixel display screens). Furthermore, SAGE enables users to freely move, resize and overlap the application windows by dynamically reconfiguring pixel streams.

The most unique feature of SAGE is the high-speed graphics streaming capability over wide-area networks as shown in Figure 2. SAGE can use various streaming protocols such as LambdaStream [18-19], that are designed for high-bandwidth and high round-trip time networks. By decoupling graphics rendering from graphics display, visualization applications developed on various environments can easily

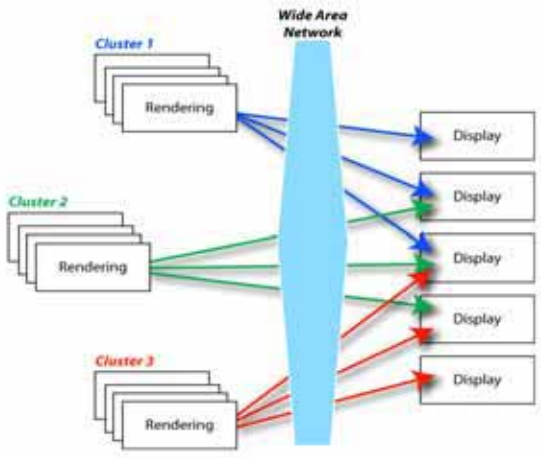

Figure 2. SAGE over Wide Area Network

migrate into SAGE by streaming their pixels into the SAGE virtual frame buffer. Also, SAGE provides scalability by supporting any number of rendering and displaying nodes, number of tiles, and screen resolution. The SAGE visualization applications have extremely fast access to huge datasets at remote or local sites taking advantage of affordable ultra-high-bandwidth networks. Moreover, we are extending SAGE to scalably support distance collaboration with multiple endpoints by streaming pixels to all the participating endpoints using either traditional IP multicasting or optical multicasting. We will discuss more about the multicast issue later.

SAGE has successfully supported our high-resolution-display LambdaVision (Figure 1) that is a 17-foot wide, tiled display built from an 11x5 array of LCD screens with a total resolution of 100 megapixels. A high-resolution display like LambdaVision is essential in visualizing large datasets without losing details. Geoscientists working with aerial and satellite imagery (365Kx365K pixels maps) and neurobiologists imaging the brain with montages consisting of thousands of pictures from high-resolution microscopes (4Kx4K pixels sensor) are good examples of SAGE and LambdaVision users. SAGE now runs across most Global Lambda Visualization Facility (GLVF) [2] research sites as well as industrial sites including Rincon Research Corporation and Nortel.

A. SAGE Framework

The SAGE framework consists of various components: Free Space Manager (FSManager), SAGE Application Interface Library (SAIL), SAGE Receiver, synchronization channel, and UI Clients as shown in Figure 3. The Free Space Manager (FSManager) is the window manager of SAGE. This is akin to a traditional desktop manager in a windowing system, except that it can scale from a single tablet PC screen to a desktop spanning over 100 million pixel displays. The FSManager receives various user commands from UI clients such as application execution, window move, resizing or z-order change (overlapping windows) and then executes the commands by sending control messages to SAIL and (or) SAGE Receivers. SAIL is a very simple API Library that allows SAGE applications to communicate with the FSManager and stream pixels to SAGE receivers. A SAGE Receiver is a software object running on each display node that is in charge of receiving pixel streams of one application instance. The received pixels then are loaded into the graphics card memory and drawn on the screens driven by the display node. To display an application image on the tiled display, we need to synchronize the sub-images on each tile and make one large consistent image. We designed the display synchronization channel among SAGE receivers and the rendering synchronization channel among SAIL instances for parallel applications. UI Clients are provided to allow users to control the Free Space Manager and monitor the status of SAGE. UI Clients can be Graphical User Interface, text-based console or tracked devices. Any UI client can execute, shutdown, move, and resize SAGE applications in a manner very similar to a typical contemporary windowing system. Furthermore, UI clients can reside on any machine (laptop, tablet, desktop etc.) that can be connected to the Free Space Manager over any network.

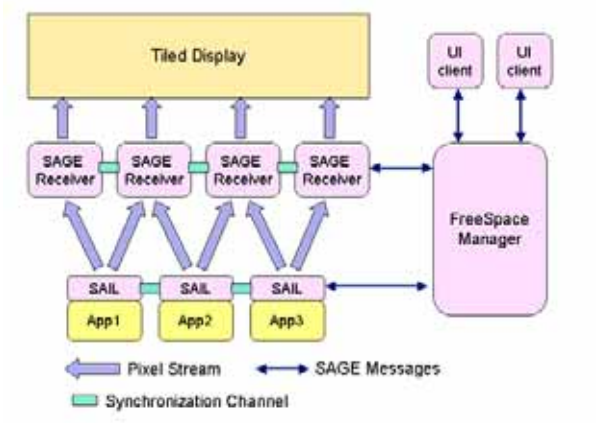

Figure 3. SAGE components

B. Demonstration of SAGE Applications

Figure 1 shows four real applications used for a typical SAGE demonstration. The display in the figure is located in the Electronic Visualization Laboratory (EVL) in Chicago. MagicCarpet on the right is an interactive ultra-high-resolution multi-resolution image viewer. It was streaming Blue Marble dataset created by NASA from San Diego to EVL using UDP. JuxtaView [7] in the middle is a high-resolution image viewer that can pan and zoom over a huge image dataset such as 356Kx356K aerial photography. It was locally streaming the aerial photography of downtown Chicago using TCP. Bitplayer on the top-left is an HD animation player developed by the National Center for Supercomputing Application (NCSA). It was streaming an animation of a tornado simulation from the StarLight facility (located three miles away in downtown Chicago) to EVL using UDP. Scalable Visualization Consumer (SVC) on the bottom-left developed by Gwang-ju Institute of Science and Technology (GIST) was locally streaming HD camera live feed using TCP. Table I shows the sustained bandwidth consumption, frame rate, rendering resolution and the number of rendering nodes of these applications in this experiment. SAGE can simultaneously support these applications without decreasing their performance.

C. Requirements of LambdaGrid Visualization Applications

Interactive ultra-high-resolution LambdaGrid visualization applications routinely access remote datasets spanning multiple terabyte and visualize the rendered pixels on high-resolution displays. The network bandwidth requirements for browsing these datasets or pushing the rendered pixels to remote displays are in the range of several tens to hundreds of gigabits per second. In addition to the huge bandwidth usage, these applications usually create hundreds of bidirectional streams between distant endpoints, each with differing flow

TABLE I. PERFORMANCE OF SAGE APPLICATIONS

Application	Bandwidth (Mbps)	Frame Rate (fps)	Rendering Resolution	Node Num
MagicCarpet	6737.3	33.7	3200x3000	10
JuxtaView	850.6	4.0	3200x3200	8
Bitplayer	516.8	11.3	1920x1080	1
SVC	538.4	24.9	1440x1080	1

TABLE II. NETWORK FLOWS CREATED BY ULTRA-HIGH-RESOLUTION GRID VISUALIZATION APPLICATIONS

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	Medium	Constant	Small	UDP-based
HD Video Stream	1 per user	Medium to High 25Mbps-1.5Gbps	Yes	Yes	Medium	Constant	Small to Medium	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

requirements operating over differing transport protocols. Table II quantifies the broad variety of flows that simultaneously emanate from SAGE. 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. What is 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 [8-10]; explicit feedback-based control, such as those in XCP, TeXCP, ECN and NetX; adaptation of real-time (non-commodity) systems for application performance [11-12]; feedback-based application adaptation on commodity end-systems (such as Active Harmony, Prophesy); provisionable user-controlled networks (such as UCLP, PIN/PDC, DRAGON, CHEETAH, OMNInet, EnLIGHTened, OptIPuter); network Quality of Service [13-14]; high-performance transport protocols [15-20]. 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 4 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. The applications, cluster nodes and optical links are currently in place; the controller and hardware of LambdaBridge are under development. LambdaBridge will manage and adapt all flows

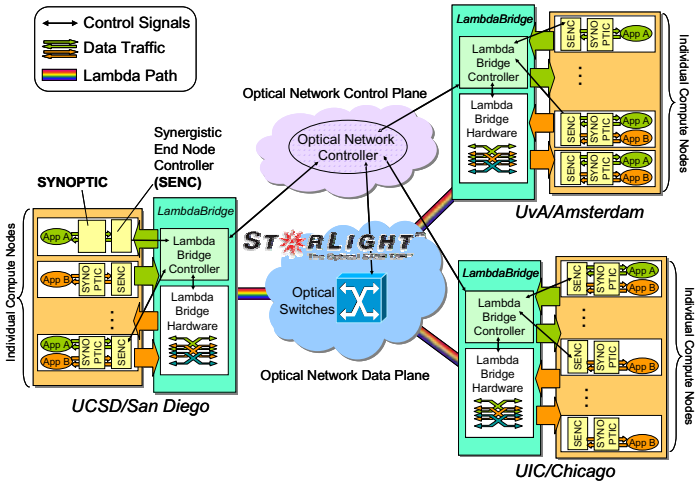

Figure 4. The LambdaBridge testbed

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 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.

A. LambdaBridging

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

1) 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 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 subsection *B*.

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 subsection.) 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 [21] can be integrated to enable policy and security enforced network resource utilization.

2) 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 present two of these solutions.

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 [22] and the IEEE L2 Scaling Enhancement activities (e.g. 802.1ah, 802.1ad, 802.1s), and IETF activities (e.g. [23]) 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 [24-25] 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 [26] 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 [27] could be a good starting point for such implementation.

B. 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 II).

The Synergistic Flow Framework consists of SYNOPTIC, the Synergistic End Node Controller (SENC) 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 monitors and synthesizes the end-systems' conditions, provides feedback to SYNOPTIC and schedules SYNOPTIC flows.

1) SYNOPTIC

SYNOPTIC is a high-performance, application-level, end-to-end, configurable, composable and extensible protocol framework. SYNOPTIC leverages our prior work in Quanta [28] (a cross-platform networking toolkit for supporting the diverse networking requirements of latency-sensitive and bandwidth-intensive applications), and related work in composable protocols [29]. 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 shared physical paths.

2) 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 5, consists of:

a) Kernel and User-Space Monitoring (MAGNET)

We are currently developing Monitoring Apparatus for Generic Kernel Event Tracing (MAGNET) v3.0 [30-31] 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.

b) 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 [32] and a Discrete-Time Stochastic Control System [33] based on the synthesized feedback and the response latency.

c) 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 [34-35]. 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.

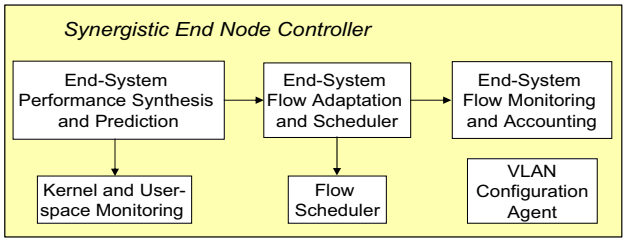

Figure 5. Functional modules of a Synergistic End Node Controller

d) 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.

3) Synergistic Network Controller (SNC)

As described in subsection *A*, 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. as long as it satisfies the constraints of the network and end-system capacity. It is possible to build an application level rate-based congestion control framework wherein an application can plugin any suitable congestion control heuristic. An application could also specify different traffic shaping heuristics for flows with differing properties.

IV. SCALABLE VISUALCASTING

An essential requirement of all collaboration systems is the ability to broadcast or multicast information to all collaborating sites so that all participants can simultaneously see and interact with the data. Displays such as LambdaVision are expected to support such collaborative visualizations. In the next few years, a crucial goal for GLVF [2] is to develop a capability where high-definition video and ultra-definition visualizations can be scalably multicasted, in real time, across all the distributed sites. Typical uncompressed HDTV streams at 30 frames per second require 1 Gbps of network bandwidth and cluster driven tiled displays can render video up to several tens to hundreds of gigabits per second. Although a variety of techniques exist for supporting reliable multicast, high-bandwidth and low-latency, reliable multicast is an unsolved problem and an active area of research within the Grid community [36]. It is a particularly challenging problem at the endpoints that must receive the multicast traffic; and must be solved for any applications that seek to use networked tiled-displays. A scalable solution is needed for distribution of video and rendered graphics streams to multiple tiled-displays at the order of several tens of gigabits per second.

We investigate this problem by developing a scheme called *Scalable Visualcasting* that is specifically designed to provide the kind of image multicasting service needed for ultra-definition visualization. The Scalable Visualcasting scheme will be built on top of SAGE. In the SAGE model, the image generation source is decoupled from the display and fed by a high-speed network. The bandwidth utilized is dependent on the image resolution of the rendering source. As images on the tiled display are resized or repositioned on the walls, SAGE must re-route the multiple streams from the rendering source to the computers that drive the displays. When multicasting in introduced, the problem becomes much more complex. This is because the end-points of the streams may comprise entirely different display configurations (See Figure 6: top diagram). The image generation source must now scalably replicate and route the pixels to the correct destinations on all the tiled displays, not just one.

IP Multicast. There are two main approaches to solving this problem. The first approach uses layered IP-multicasting to interleave each source stream over multiple multicast addresses and coordinates the receivers so that each display knows which layered stream to subscribe in order to assemble the images for the desired screen resolution (Figure 6: middle diagram). The second approach uses a cluster of PCs to replicate the streams on behalf of the rendering cluster (Figure 6: bottom diagram). These replication services can be dynamically scaled in proportion to the number of remote sites that are participating in a collaborative session. Understanding the requirements, benefits and limits of each approach will provide valuable input into future Internet systems design.

Optical Multicast. A more innovative solution to Scalable Visualcasting is optical multicast. With the fast growing interest in bandwidth-intensive visualization and collaboration over all-optical networks, supporting multicast in the optical

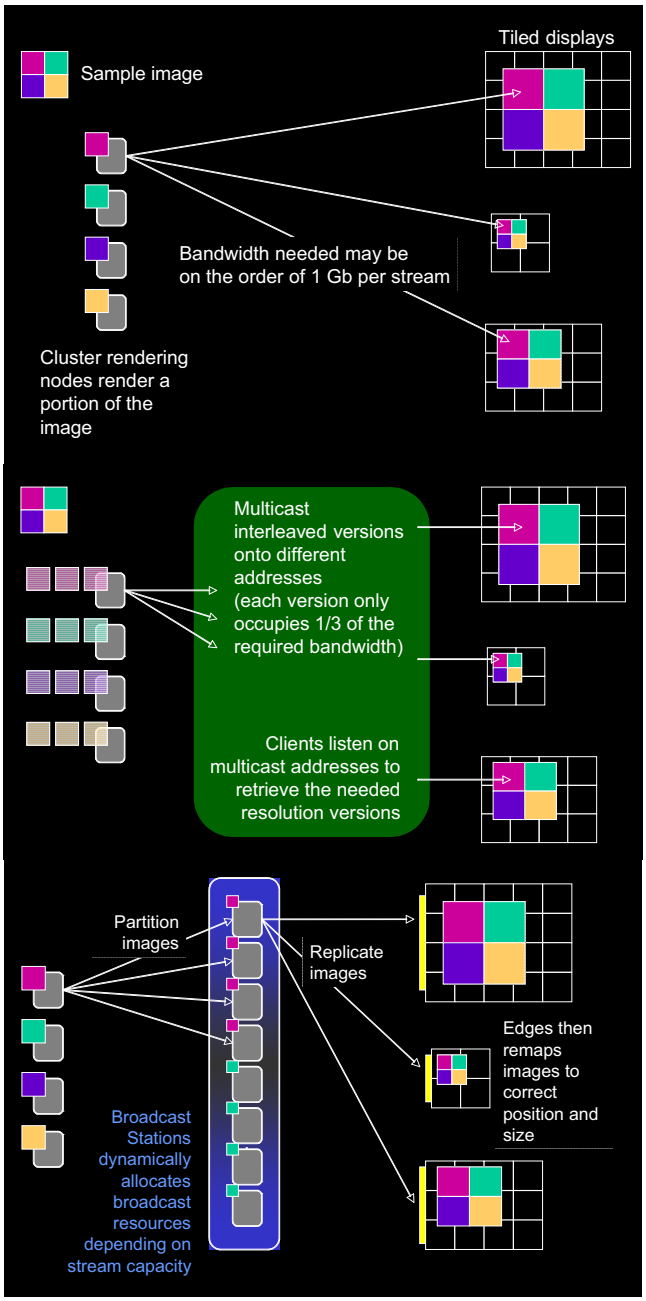

Figure 6. Multicasting visualizations to multiple tiled displays of non-uniform configuration.

layer is becoming an important topic. Multicasting in optical domain requires light splitting functionality, which means that a given incoming optical channel can be split to multiple copies, with each copy able to be switched to a different output channel. Optical splitter is the key component to realize multicast in optical networks. Optical multicast is becoming possible since 1:2 and 1:4 splitters are widely available and more recently 1:8 splitters have been developed. To compensate the power penalty introduced during the optical signal splitting, signal repeaters (either all-optical or O-E-O) are necessary to push the power levels back up. By using multiple signal repeaters and optical splitters in a tree hierarchy, it is possible to build a multicast tree and create as many copies of the original signal as possible. DWDM units can be used to multiplex streams from multiple sources (or multicast groups) on the same distribution tree. This concept is illustrated in Figure 7.

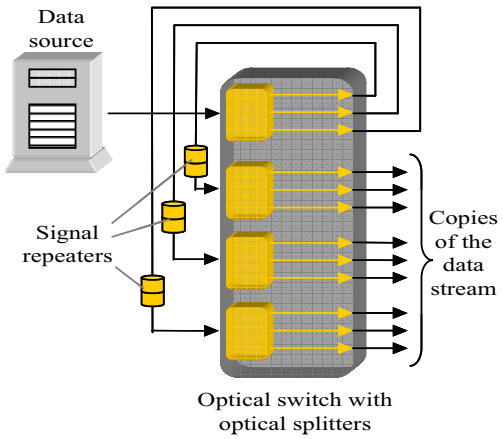

Figure 7. The concept of optical multicast

We carried out a proof-of-concept experiment to test the basic functionalities of optical Visualcasting. Figure 8 shows the experimental setup. We used a Glimmerglass REFLEXION RFX-64 optical switch with its optical multicast module, which allows replication of a light signal into up to four copies of itself. As shown in the figure, the sending NIC is connected in full duplex with receiver 1 so that the sender's link is operational fully in both directions. Not doing so causes the sender NIC to report that the link was down during transmission. For the other two receivers (receivers 2 and 3) we forced the Auto-Negotiation to be switched off and the NICs were working in half duplex mode. We used SysKonnect SK-9841 GigE NICs for our experiments. All receivers get the senders transmit signal after it was replicated by the multicast module. For the actual data transfer, TeraVision [37] was used to send high-resolution uncompressed video at about 750 Mbps over the testbed. The video data was successfully distributed to 3 receivers and video images properly displayed. We plan to continue exploring optical multicasting and its impact on distributed collaborative applications. In addition to experimenting on optical multicast with 10Gb NICs, we will also run tests on national and international optical network testbeds to examine methods for wide-area multicasting.

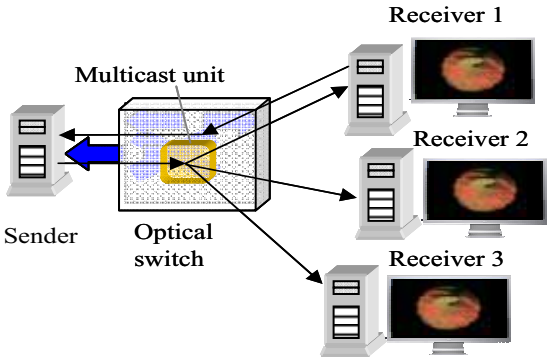

Figure 8. Optical multicast experimental setup

LambdaBridge Multicast Support. Various solutions can be conceived to incorporate multicast functions into LambdaBridge. One of the most straightforward approaches is to let the LambdaBridge at the sender site work as 'broadcast stations' as those shown in the bottom diagram of Figure 6. This can be realized by integrating SAGE image (pixel) streaming functionality with the LambdaBridge (in short, "SAGE LambdaBridge"). The SAGE LambdaBridge intercepts each image stream and performs replication using either IP multicast or optical multicast. In the case of IP multicast, the SAGE LambdaBridge is also possible to perform re-routing operations on behalf of display clients by means of dynamically routing desired layer of image streams to destination display nodes. For the case of optical multicast, the SAGE LambdaBridge can use embedded multicast unit (optical splitters) to generate multiple optical streams, or work jointly with multicast-capable core optical switches to replicate streams in the core optical network. Note that in this case, pixel re-routing is realized at the endpoint. An IP-optical hybrid multicast approach is also conceivable for achieving enhanced flexibility and scalability. The multicast functionality is closely tied together with the traffic management and lambda provisioning mechanism of LambdaBridge, thereby providing improved application performance while optimizing the network resource utilization.

V. CONCLUSION

This paper described the insatiable and diversified traffic demands made by LambdaGrid applications as characterized by SAGE-based collaborative visualization and 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 unified resource management that includes end-systems and the network. This achieves desirable end-to-end performance while optimizing the resource utilization. LambdaBridging contributes to a deep understanding of the factors that influence the performance of data-intensive applications from the point of view of application data flows and their interaction with backplane, OS, memory, CPU, system bus, and network elements. This paper also introduced the concept of Scalable Visualcasting, a way of providing high-bandwidth multicasting

service needed for ultra-definition visualization. Several approaches including both IP multicasting and optical multicasting are presented and the ways of incorporating various multicasting strategies into the LambdaBridge framework are discussed. Work is underway to build a prototype LambdaBridge and develop a variety of protocols, software and tools necessary for LambdaBridge control and management.

There remain numerous research problems for future investigation. These include: understanding how LambdaBridge software and hardware will scale in practice to Terabits of bandwidth; how to develop transport protocols that are both end-system and network-aware; how to monitor end-systems performance at tens of gigabits per second rates with minimal overhead; how to scalably manage and coordinate thousands of simultaneous flows of varying characteristics; how to efficiently aggregate flows and provision VLANs to Lambdas. These challenging issues will be addressed in future work.

ACKNOWLEDGMENT

We would like to thank Philip Papadopoulos at UC San Diego and Cees de Laat at the University of Amsterdam whose opinions have helped shape this concept. This project was funded in part by the following grants from the National Science Foundation: CNS 0420477 and ANI-0225642.

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. DeFanti, J. Leigh and P.Papadopoulos, "The OptIPuter," *Comm. of the ACM*, vol. 46(11), pp. 58-67, Nov. 2003.

[4] J. Leigh, L. Renambot, T. DeFanti, et al, "An Experimental OptIPuter Architecture for Data-Intensive Collaborative Visualization," 3rd Workshop on Advanced Collaborative Environments, Seattle, WA, June 2003.

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

[6] 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.

[7] N. K. Krishnaprasad, V. Vishwanath, S. Venkataraman, A. G. Rao, L. Renambot, J. Leigh, A. E. Johnson, and B. Davis "JuxtaView – a Tool for Interactive Visualization of Large Imagery on Scalable Tiled Displays" , *Proc. IEEE Cluster 2004*, San Diego, Sept 20-23, 2004.

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

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

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

[11] 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.

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

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

[14] 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.

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

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

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

[18] 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.

[19] V. Vishwanath, J. Leigh, E. He, M. D. Brown, L. Long, L. Renambot, A. Verlo, X. Wang, T. A. DeFanti, "Wide-Area experiments with LambdaStream over dedicated high-bandwidth networks," IEEE *INFOCOM 2006*, Barcelona, Spain, Apr. 2006.

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

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

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

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

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

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

[26] 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/>

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

[28] 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.

[29] 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.

[30] V. Vishwanath, W. Feng, M. Gardner, and J. Leigh, "A High-Performance Sensor for Cluster Monitoring and Adaptation," EVL technical document, 2006. <http://www.evl.uic.edu/cavern/rg/20060505_vishwanath/>

[31] 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.

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

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

[34] 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.

[35] 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)*.

[36] M. Burger, T. Kielmann, H. E. Bal, "Balanced Multicasting: High-throughput Communication for Grid Applications", *Proc. SC 05*.

[37] R. Singh, B. Jeong, L. Renambot, A. Johnson and J. Leigh "TeraVision: a Distributed, Scalable, High Resolution Graphics Streaming System" , *Proc. IEEE Cluster 2004*, San Diego, Sept 20-23, 2004.