On-demand Overlay Networks for Large Scientific Data Transfers

Lavanya Ramakrishnan*, Chin Guok*, Keith Jackson*, Ezra Kissel+ , D. Martin Swany+, Deborah Agarwal*
*Lawrence Berkeley National Laboratory, Berkeley, CA
+ University of Delaware, Newark, DE

Abstract. Large scale scientific data transfers are central to scientific processes. Data from large experimental facilities have to be moved to local institutions for analysis or often data needs to be moved between local clusters and large supercomputing centers. In this paper, we propose and evaluate a network overlay architecture to enable high-throughput, on-demand, coordinated data transfers over wide-area networks. Our work leverages Phoebus and On-demand Secure Circuits and Advance Reservation System (OSCARS) to provide high performance wide-area network connections. OSCARS enables dynamic provisioning of network paths with guaranteed bandwidth and Phoebus enables the coordination and effective utilization of the OSCARS network paths. Our evaluation shows that this approach leads to improved end-to-end data transfer throughput with minimal overheads. The achieved throughput using our overlay was limited only by the ability of the end hosts to sink the data.

Keywords – Overlay Networks, Scientific Data transfers, Advanced Reservations, Dynamic Provisioning

I. Introduction

Data has become central to scientific computations and collaborations[1]. Scientific users typically generate the data at instruments and computational facilities and then transfer this data from those sites to their local clusters and desktops to facilitate analysis and sharing of data and results with collaborators that might be geographically distributed.

Peta-scale data is usually analyzed where it was generated [36]. However, most science experiments and models only generate terabytes of data, and this data is moved to where the analysis of the data takes place. An example, is the experiments conducted at the Advanced Light Source (ALS) facility [2]. Scientists from around the country spend days to weeks in Berkeley gathering data from ALS X-ray detector experiments. Each data set collected from a typical experiment at the beamline contains about 10GB of data and about one data set is produced per hour. Overall a total of about 0.5 TB is generated over 24 hours. The data generated during the experiment subsequently needs to be transported to the scientist's home institution for interactive scientific analysis and long term archiving. Today these data transfers are accomplished by scientists hand-carrying portable disks.

At a recent ESnet[1] Workshop in 2007[1], DOE experiment facility users made it clear they would like to stop hand-carrying the data and outlined their needs to transfer large-scale data over the network, increased data throughput capabilities, distributed data analysis, and reliable "cargo-carrying" wide-area network infrastructure.

International scientific experiments such as Atlas [38] and CMS [39] schedule computations to run on a diverse set of globally distributed resources. Intermediate data products are moved between resources during the analysis to reach the next computation stage and high throughput for the data transfer is required to enable timely completion of the computations. A third example is cloud computing which provides on-demand access to resources on a pay-as-you-go model [4]. Data movement in and out of these clouds is a major open challenge.

These scientific use cases illustrate the following needs:

- High-throughput, *on-demand* data transfer over wide-area networks
- Infrastructure in the network that can provide network service predictability by managing dynamic data transfer paths
- Simple client-side tools that allow end scientists to better utilize the available throughput of wide-area networks.

In this paper, we describe our high-throughput, on-demand data transfer architecture in detail and evaluate its effectiveness for large-scale, wide-area data transfers. The on-demand nature of our architecture enables scientists to move their data and computation between any source and destination pairs. We report on our experiences in using this architecture to orchestrate wide-area data transfers on ESnet.

[1] ESnet is a multi-10Gbps backbone Tier 1 ISP connecting DOE research facilities and compute resources.

Specifically, in this paper, we make the following contributions

- Describe a secure, reliable, light-weight, on-demand overlay architecture to manage ad-hoc scientific data transfers over the wide-area.
- Demonstrate the advantage of dynamic on-demand overlay networks by comparing and contrasting it with the performance, overhead, and bottlenecks associated with current data transfer mechanisms.

The rest of this paper is organized as follows. We provide background and related work in Section II. We present our system architecture along with design and implementation details in Section III. We detail our evaluation and results in Section IV. We discuss the various aspects of the system with respect to use in real production environments in Section V and conclude in Section VI.

II. Background

Wide-area networking infrastructure facilities such as ESnet provide the capacity to enable rapid transfer of terabyte data sets, but, existing high-throughput data transfer techniques require extensive system and network administration skills to manually tune the network protocol parameters in the end-point machines and network infrastructure along the path. Scientists rarely have the resources to implement tuned end-to-end paths.

Some sites have implemented optimized data transfer nodes dedicated to moving large-scale data (e.g., Data transfer nodes at NERSC [27]). This approach works well when the transfers are between well-known end-points since these paths can be optimized. However, this approach does not generalize to moving data to any arbitrary remote site. Previous work has suggested a variety of strategies such as alternate path routing [24],[25],[26],[30] or public storage servers located strategically in the network to optimize such data transfers [3]. But these strategies incur an administrative burden to maintain the servers and are not scalable to arbitrary destinations.

One of the key issues is that these wide-area bulk data transfers require high-bandwidth over a high latency link. TCP's congestion avoidance on high latency links with even a small amount of packet loss will experience significantly less throughput than is available. The end-to-end performance seen by the user is also affected by TCP parameters (e.g. window size) and slow loss recovery due to high round trip times.

Distributed network storage [3] has been proposed as a way to manage data transfers using locality and caching of data. However maintenance of file servers at different points in the network is a nightmare for system and network administrators.

Application overlays have been effective in providing data aggregation and dissemination services in the general Internet[5],[6],[7],[19],[20]. Application overlays create a virtual topology on top of the physical network and optimize network utilization for the application data traffic. Application layer overlay networks, such as Gnutella [6], BitTorrent [5] and Skype [7] address the needs of applications with replicas, asymmetrical links, multiple source and destination choices for each file. Existing work at the application layer [8],[9],[10],[11],[12],[13],[14],[15],[16],[17],[18],[19],[20] can be applied to produce algorithms and protocols that make efficient and effective use of resources at lower layers of the network stack.

Our approach leverages the benefits of application overlays but embeds the overlay in the underlying network. We create on-demand embedded overlays for scientific applications to make efficient and effective use of the available network resources. Our system leverages Phoebus[21],[22] and OSCARS[23] to implement the on-demand overlay capability. Phoebus provides us the routing infrastructure to select an optimal path. High-bandwidth network segments are provisioned using the prototype OSCARS (On-demand Secure Circuits and Advanced Reservation System) service. Phoebus and OSCARS are both light-weight services and do not require access to storage or other high-end resources.

In earlier work, improvements to GridFTP used multi-hop path splitting and multi-pathing[30] to alleviate problems with current networks. However these algorithms are applied at the protocol level and require changes to the middleware such as GridFTP. Also, in earlier work, alternate path routing and selection have been used to alleviate the problems with current networks [24],[25],[26]. Phoebus is similar to this work in that it provides an alternate path for data transfers but the Phoebus-OSCARS combination differs in spirit as it addresses the need for the Internet architecture to support such applications on-demand and it provides better quality of service to applications.

We describe our data transfer architecture in detail in the next section.

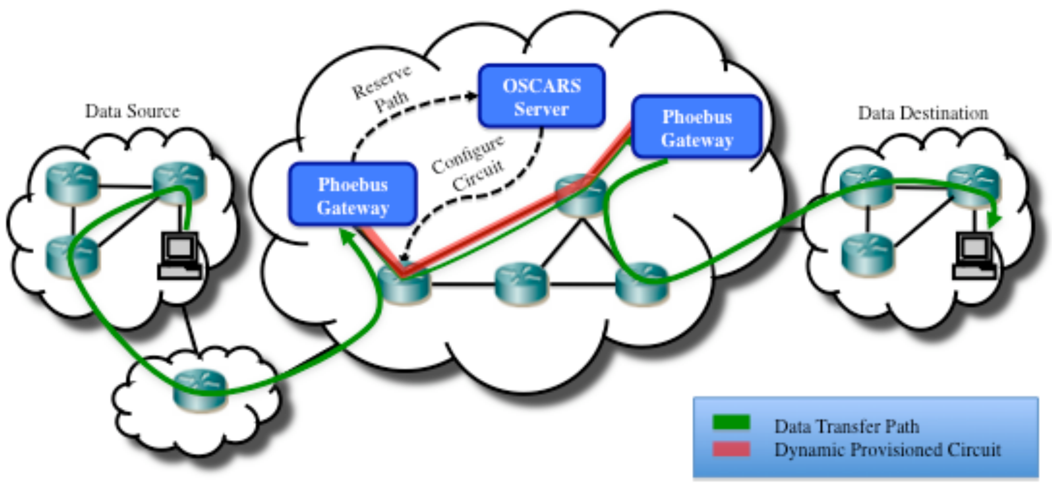

Figure 1. System components framework to facilitate high-speed data transfers

III. Data Transfer Architecture

Figure 1 shows the system components in our architecture to facilitate wide-area data transfers. Phoebus manages end-to-end throughput for long-distance data transfers by splitting the network path into constituent segments. OSCARS enables on-demand provisioning of secure circuits with guaranteed bandwidth. Using OSCARS on the high latency portions of the path allows Phoebus to obtain high bandwidth across the wide-area.

A client wishing to transfer data connects to a close-by Phoebus gateway and the Phoebus gateway then interacts securely with the OSCARS servers to obtain a reservation for a guaranteed bandwidth path to the Phoebus gateway nearest the destination. The OSCARS server checks to see if the reservation request can be satisfied. If the reservation can be satisfied the OSCARS servers configure the path. Once the circuit is setup the Phoebus gateway relays the data to the other Phoebus gateway which then sends it to the final destination. We detail the system components and our design decisions in the rest of this section.

A. Phoebus

Phoebus provides an infrastructure for managing high-performance wide-area networks that splits a single end-to-end data transfer session into multiple sessions on distinct network segments. The Phoebus gateways can use specific transport protocols and properties for each segment. Phoebus gateway provides basic buffering and 'store and forward' capabilities to facilitate different network link speeds.

The Phoebus gateway can be configured to tweak protocol specific parameters (e.g. TCP parameters, OSCARS authentication and reservation information) on incoming and outgoing connections and on connections to particular destinations. The gateway is configured with next hop information for each destination or subnet. The next hop in the path might be another Phoebus gateway or a direct connection to the end host. For example, a Phoebus gateway in Berkeley would redirect all traffic headed to Indiana University's subnet to the Phoebus gateway running at Starlight in Chicago and the Phoebus gateway at Starlight would then forward the traffic directly to the end host.

In our architecture we deploy Phoebus gateways close to our data source and at major wide-area network access points. The path between the Phoebus gateways is managed using the OSCARS framework. This strategy can be expanded easily to include multiple Phoebus gateways and multiple OSCARS provisioned pipes along the path.

B. OSCARS

OSCARS is a service that enables advance reservation of guaranteed bandwidth paths. OSCARS operates within ESnet and has the capability to interoperate with other network domains. OSCARS software infrastructure uses a web services model and supports PKI for authentication. The OSCARS service allows users to request a reservation from a source to a destination over a *specified network path* at a *specified bandwidth* for a *specified duration*. The OSCARS server interacts with the network devices along the path to configure the virtual circuit (VC). OSCARS supports user driven advanced reservations of dynamic VCs at layer 2 (Ethernet VLANs), and layer 3 (IP).

Phoebus initiates a request to the OSCARS server for a circuit when it gets a request that can benefit from a provisioned network path. Phoebus then periodically polls the OSCARS server and waits for the circuit to be setup. Once Phoebus notices that the

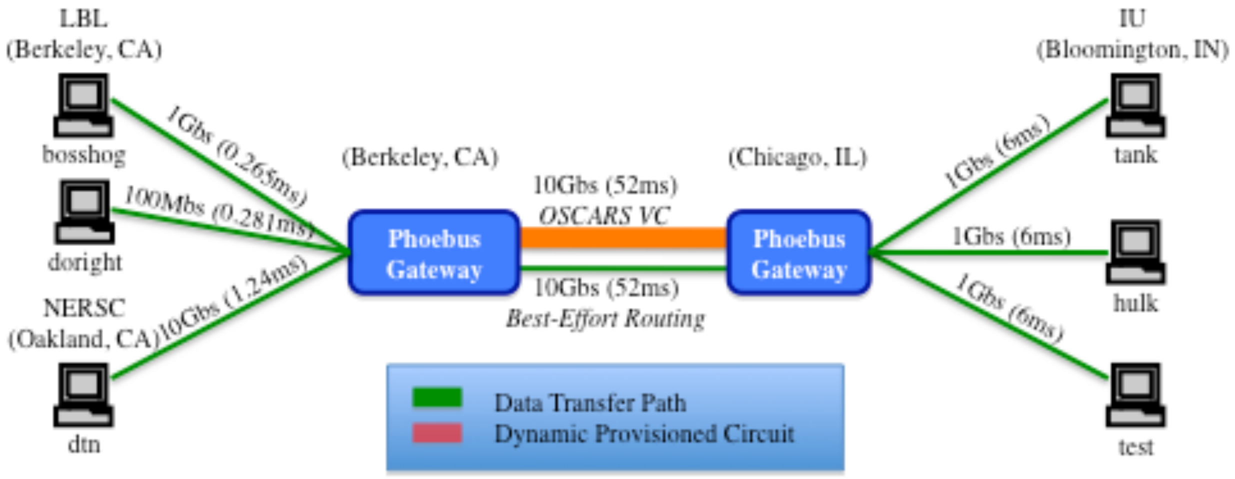

Figure 2. Network data plane topology of testbed

circuit is setup, it continues the data transfer that was initiated by the client along the newly setup path.

C. *Quality of Service*

The high latency in the wide-area transfers makes it hard for TCP congestion recovery algorithms to react in a timely manner. In our approach, we split our network path into two low latency paths at the ends and we use OSCARS to guarantee the required bandwidth over the high latency segment. Thus we are able to achieve superior levels of performance. This approach provides higher reliability levels since any problems in the end links can recover quickly due to the fact that they are low latency paths [35].

D. *Cost and Usability*

The OSCARS service that dynamically provisions and guarantees bandwidth is currently available to a limited set of users at no cost. As OSCARS becomes more widely used, it is likely to introduce a cost model. However, this is not unlike the allocation scheme used for shared computational resources today. In the case of deadline scheduling, where network and compute resources must be co-scheduled, the allocation framework may be closely coupled.

The Phoebus infrastructure is light-weight and has minimal impact on the end-user's interaction with existing software tools. The Phoebus wrapper library needs to be installed on the system that processes the packet headers to redirect traffic to the closest Phoebus gateway. Once these are installed on a system, socket based programs work transparently to the end-user. The Globus GridFTP server can with minor (already available) patches support the Phoebus stack through Globus XIO[34].

E. *Robustness*

Phoebus and OSCARS are both research, software projects. Experimental versions of the Phoebus software package are available today [33] for applications that might need large-scale, wide-area data transfers. Our extensive testing didn't reveal any functional problems. While additional testing and work would be needed to make these services available more widely on production systems, the light-weight design of these services means that it will be relatively easy to ensure robustness.

F. *Security*

Our infrastructure is inherently as secure as existing systems involved in the operation i.e., the end systems and the network infrastructure. OSCARS requires PKI authentication and the Phoebus installation is configured to use the appropriate credential. Existing security mechanisms such as Grid Security Infrastructure (GSI) can be used in this environment for end-to-end authentication, as required by tools such as GridFTP. Additional support will be needed to support user credential delegation at the Phoebus and OSCARS layer f they are to be used in multi-user production environments.

IV. EVALUATION

In our evaluation we compare and contrast the non-overlay end-to-end throughput with that obtained through our embedded dynamic overlay approach. Additionally we study the overhead associated with our system components. In the rest of this section we discuss our testbed setup, present measurements to show our system overheads, and present our throughput comparison results using iperf and GridFTP.

A. *Testbed Setup*

We have a wide-area network testbed setup between Lawrence Berkeley National Lab (LBL) in Berkeley, California and Indiana University (IU) in Bloomington, Indiana. Figure 2 shows the network topology plane of our testbed. Each experiment was run multiple times over a period of two months. The average results are presented.

Machines. At LBL, we used three machines bosshog – a local cluster machine, doright – a desktop machine, dtn – a production data transfer node at NERSC [27]. The first machine (bosshog) is 1G connected to a router at Berkeley, the second machine (doright) is 100Mb connected, and the third machine (dtn) is 10G connected to the ESnet router. Similarly we have three machines at IU – tank, hulk and test. These machines are 1G connected to the Internet. The machines at LBL have been tuned for optimal long latency TCP performance over the Internet.

Phoebus gateways. The ESnet routers at Berkeley and Chicago each have commodity linux boxes that are the Phoebus gateways. The circuits between the Phoebus gateways are managed through OSCARS.

Client Tools. We use two client tools to characterize the behavior of the system. We use iperf [28] for measuring throughput across the links. We also perform throughput tests using GridFTP [29] in memory-memory mode and with various randomly generated large files.

Theoretical Bandwidth.

	Window Size (MB)	Latency (ms)	Theoretical Bandwidth
tank(IU) to bosshog (LBL)	4	58	578 Mbps
tank(IU)-phoebus-star	4	6	5.6 Gbps*
phoebus-star-phoebus-lbl	16	52	2.5 Gbps
phoebus-lbl-bosshog (LBL)	16	0.265	506 Gbps*

Table 1. Theoretical bandwidth possible between IU and LBL and the individual links in our overlay network path. * The actual bandwidth is limited by the links which are 1 Gbps.

We measured the latency on our links and used the buffer size to calculate the theoretical bandwidth possible on each of the links. This gives us an upper bound on what to expect across the paths. The theoretical bandwidth between IU and LBL on the Internet is 578 Mbps. The maximum theoretical bandwidth from IU to LBL through the Phoebus gateways is significantly higher due to low latency at the end links and higher bandwidth available between the phoebus gateways (Table 1). Of course the actual bandwidth possible on these paths is limited by the low-bandwidth link which is 1 Gbps on all paths. Based on the theoretical bandwidth comparison we expect between 1.5 and 2 times speedup with our overlay networks approach.

Measured Bandwidth. We measured the bandwidth between different links in our testbed (Table 2). The bandwidth over the Internet between LBL and IU was between 350-400 Mbps. Tests from LBL to IU resulted in the higher bandwidth values (around 400Mbps). This was due to the TCP performance tuning of the LBL machine (bosshog). Tests from the IU and LBL hosts to the Phoebus gateways had bandwidths greater than 930 Mbps, which defines the upper limit on throughput we could achieve using the overlay network path. We also collected measurements from Windows machines that are representative of many scientists' desktop machines. Measurements from Windows server machines from LBL to IU, and to the Phoebus box at LBL, showed that Windows Server 2008 out performed Windows Server 2003 due to a better tuned TCP stack.

	tank (IU) (Mbps)	phoebus-lbl (Mbps)
Bosshog (LBL)	392	942
phoebus-star	935	-
LBL-Windows-2003	8.78	430
LBL-Windows-2008	368	950
	bosshog (LBL)(Mbps)	**phoebus-star (Mbps)**
Tank (IU)	290	931
phoebus-lbl	942	-

Table 2. Measured Bandwidth between links on our testbed.

B. System overheads and bottlenecks

We measured the overhead associated with initiating and setting up an OSCARS circuit. On average, the setup overhead was less than 2 minutes. For large data transfers this overhead is a very small percentage of the actual data transfer time.

Second, we measured the overhead of Phoebus itself. For this experiment we measured the bandwidth between two machines in the LBL network with and without Phoebus. Figure 3 shows the bandwidth comparison across machines in the LBL network through the direct path and through the Phoebus gateway in California. For this case, a desktop machine at LBL (doright) connected at 100Mbps was used as the sink for data transfers from the dtn (NERSC). The data transfer was limited by the smallest link capacity which was 100 Mbps in this case. As observed, the performance with and without the Phoebus gateway was comparable and the minimal difference arises from cross network congestion. Similarly we measured the traffic from dtn (NERSC) to bosshog (LBL). We observed that the bandwidth peaked around 940Mbps. Based on these results, Phoebus did not add any perceptible overhead to the system and the throughput that can be achieved is limited only by the bandwidth on the constituent links. Next we compared the throughput of wide-area end-to-end transfers.

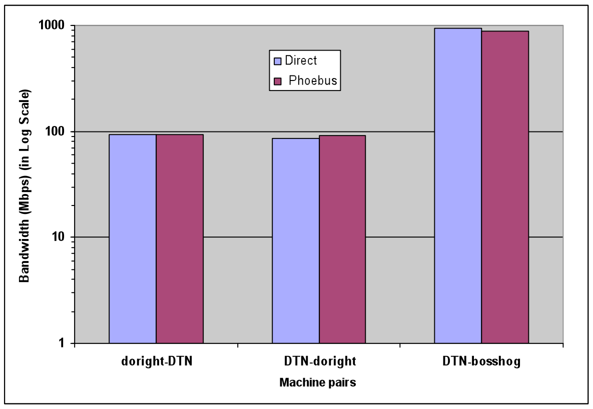

Figure 3. Bandwidth comparisons across two machines in the LBL network when transfer is a) direct across the internet and b) through a Phoebus gateway that redirects traffic.

C. *Throughput comparison*

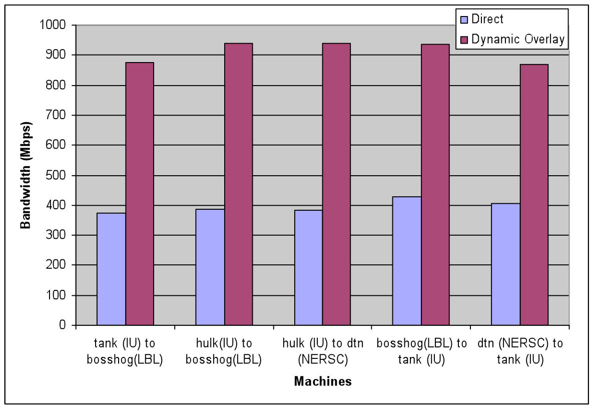

Figure 4. Bandwidth comparison across the different machine pairs across LBL and IU a) direct b) dynamic overlay.

iPerf. For our first set of tests we used iPerf (TCP). We compared the throughput difference between best-effort non-overlay routing (Direct) and the dynamic overlay system (Figure 4). We performed these experiments over various data transfer paths between LBL and IU. We compared the direct throughput with throughput using the Phoebus coordinated OSCARS circuit. In all cases, we saw an improvement of at least two times the bandwidth throughput as a result of using the overlay.

Next, we compared the performance of Phoebus without OSCARS and our on-demand overlay approach (with OSCARS). We did a test from the data transfer node (dtn) to IU (tank). Without OSCARS (i.e. best effort routing between the Phoebus boxes), we observed bandwidths around 890Mbps. Using OSCARS to provision the link between the two boxes we were able to get bandwidth around 940Mbps. Using Phoebus to route the traffic through ESnet enables routing over a high bandwidth path adding OSCARS to reserve the required bandwidth, we are able to do even better (because we do not see any loss on the reserved link).

GridFTP memory transfers. We performed a series of GridFTP memory to memory (i.e., /dev/zero to /dev/null) tests across LBL and IU to compare the throughput while accounting for the dynamic overlay setup time etc. These tests helped us understand the effective network performance possible with GridFTP without the effects of disks and other system factors (Figure 5).

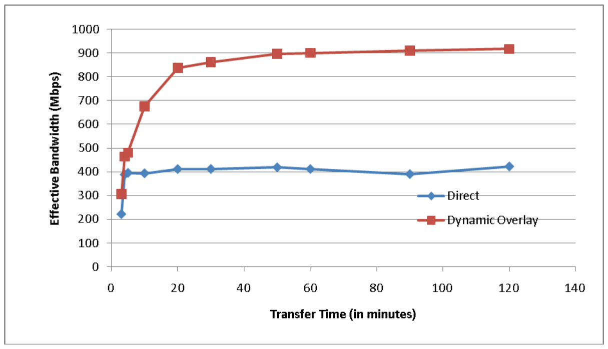

Figure 5. Comparison of throughput with GridFTP memory transfers

These tests showed that even for short transfers (of < 10 minutes) it was possible to achieve much higher effective throughput using the dynamic overlay network.

GridFTP file transfers. Finally we performed a series of tests to understand the throughput that was possible with actual disk to disk file transfers. The effective throughput from file transfers is affected by a number of factors including disk access speeds [31].

We measured the time taken to transfer different file sizes from tank (IU) to bosshog (LBL) and dtn (NERSC) with a single stream. As seen in Figure 6, for a 10GB file transfer from tank (IU) to bosshog (LBL), the direct approach, i.e., best effort routing, was slightly better than the dynamic overlay approach (due to the overlay setup overhead). However for larger files the effective throughput with the dynamic overlay approach exceeded that of the direct transfer approach (significantly shorter transfer times).

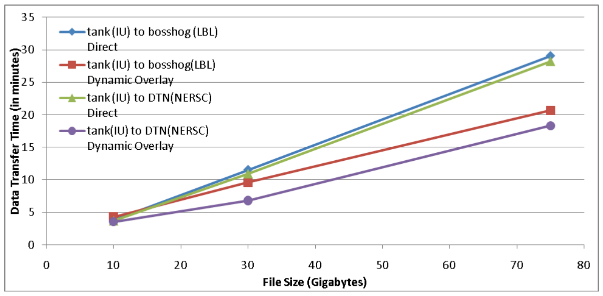

Figure 6. Comparison of time to transfer data using the direct and dynamic overlay approach using GridFTP single streams

GridFTP Parallel Streams. Next, we studied the effects of parallel streams on the data transfers over the direct and dynamic overlay approach. These tests were performed from the data transfer node – dtn (NERSC) to test (IU). Using parallel streams with the direct approach (i.e, best-effort routing) improved the performance as expected. For large file transfers, four parallel streams from dtn (NERSC) to test (IU) gave the best performance. The data transfer node at NERSC has been tuned for large-scale data transfers and four streams had been previously documented as the best configuration to use.

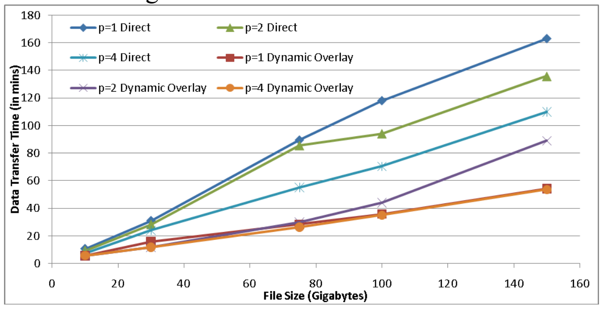

Figure 7. Comparison of time to transfer data between direct and dynamic overlay between data transfer node at NERSC and test at IU with varying number of streams (p=1, 2, or 4)

The dynamic overlay approach achieved better performance than the direct Internet routing even for file sizes as small as 10GB. Using parallel streams had little or no effect (p=4) when using the dynamic overlay approach since the performance with one stream already reached the limit of the IU host. In fact, for p=2, for larger files, we saw a drop in performance compared to using a single stream (due to overhead). Overall, the dynamic overlay did show better performance than what was achievable for data transfers from dtn (NERSC) with direct Internet routing.

Summary. Our evaluation shows that we can get an effective and higher data transfer throughput, using the dynamic, on-demand embedded overlay networks.

V. Discussion

Earlier work [21], [22], [23] and our experiments show that overlays embedded in the network using Phoebus and OSCARS provide a robust infrastructure that enables high throughput wide-area data transfers for end-user applications. In the remainder of this section we discuss our experiences.

Science Data Network and Internet2. All the tests described in this paper were carried out using ESnet. Phoebus and OSCAR deployments of the same capability can be made available on Internet2. This early work paves the way for future deployments across ESnet and Internet2 enabling wider accessibility of this infrastructure for wide-area, large-scale, on-demand data transfers.

End-to-end throughput typically limited by the end hosts. The end-to-end throughput is dependent on a number of factors. The combined infrastructure of Phoebus and OSCARS allows us to achieve consistently high throughput across the wide-area network but the overall achieved throughput of a data transfer can only be as good as the slowest link in the path or disk at the source or destination. Our results from the data transfer node at NERSC/LBL show that this problem can be addressed by using systems that are optimized for disk and network operations. In cases such as the ALS, the data transfer throughput will be the maximum possible to the remote host if a machine like dtn is installed at the ALS. Then, using the overlay network, the end user gets the same data transfer throughput to the remote file server as if the file transfer was originating local to the remote location.

Protocols. Phoebus provides support for UDT and in some cases UDT can provide improved performance on links with packet loss[22]. However, by using OSCARS on our wide-area links, we see no loss in the wide-area so UDT does not provide any performance advantage in our overlay environment. Phoebus and OSCARS provides support for multiple-users and multiple streams.

Gateway Configuration. The Phoebus gateways play a critical role in this infrastructure. It is important that these boxes are properly positioned in the network topology and well connected with the router. In our deployment, these machines run Ubuntu-Server 8.04. These machines were configured to use a Virtual Local Area Network (VLAN). Some network administration expertise was required for this configuration.

The Phoebus and the OSCARS clients that run on the network gateways are light-weight services that can run on simple Linux machines connected to a router. If Linux based routers become more common place, these services could be directly run on the router minimizing equipment needed.

Debugging and Tuning. It is important that the Phoebus boxes are tuned to get good performance.

Achieving good performance from 10Gbps Ethernet cards requires tuning on each card and machine to enable enhancements such as interrupt coalescing. Additionally due to the nature of layer 2 networks, debugging in these networks is tedious and time consuming. Tools such as perfSONAR[37] help with the troubleshooting.

File Servers. Sites such as LBL where large scale data is produced, can deploy well tuned file servers close to the source of the data. These file servers when used with the embedded on-demand overlays described in this paper can enable data transfers across the wide-area that achieve the performance of a local transfer at the remote institution.

VI. Conclusions

On-demand, embedded overlays enable high-throughput, large-scale scientific data transfers across the wide-area. The overlays use Phoebus to split network connections into a series of connections. This divides the data transfer path into low-latency 'local' links and a high-latency wide-area link. The high-latency wide-area link's path and bandwidth is reserved and managed using OSCARS, which provides guaranteed bandwidth. Our experiments show that this approach leads to increased large-scale, data transfer throughput end-to-end with minimal system overhead. This throughput can also be achieved with GridFTP transfers. Thus, a user can achieve large file transfers using the overlays that are typically as fast as the time to load or store the data from/to a locally mounted disk. Using data transfer nodes and on-demand, embedded overlays is an effective replacement for hand-carrying data disks.

VII. Acknowledgements

This work was supported by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. . The authors would like to thank DOE (DOE FG02-04Er25642) and Internet2 for Phoebus funding.

The authors would like to thank Evangelos Chaniotakis and David Robertson for helping with use of the OSCARS server.

References

[1] A. Szalay and J. Gray, "Science in an exponential world," *Nature,* Vol 440, 23 March 2006.

[2] BES Science Network Requirements report of the Basic Energy Sciences Network Requirements Workshop. *LBNL Technical Report LBNL/PUB-981*. June 2007.

[3] J. Plank, A. Bassi, M. Beck, T. Moore, D. M. Swany, and R. Wolski, Managing Data Storage in the Network. *IEEE Internet Computing* vol 5, Issue 5, Pages 50-58, Sep. 2001.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. *Technical Report No. UCB/EECS-2009-28*, EECS Department, University of California, Berkeley, Feb 2009.

[5] "BitTorrent", http://www.bittorrent.com/

[6] "Gnutella", http://www.gnutella.com/

[7] "Skype", http://www.skype.com/

[8] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Split-stream:High-bandwidth Content Distribution in Cooperative Environments. In *Proceedings of the 19th ACM Symposium on Operating System Principles*, October 2003.

[9] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. In *Proceedings of ACM SIGCOMM*, August 2001.

[10] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and Jr. J. W. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *Proceedings of Operating Systems Design and Implementation (OSDI)*, October 2000.

[11] D. Kostic, R. Braud, C. Killian, E. Vandekieft, J. W. Anderson, A. C. Snoeren, and A. Vahdat. Maintaining High-Bandwidth Under Dynamic Network Conditions. In *Proceedings of USENIX '05*, 2005.

[12] D. Kostic, A. Rodriguez, J. Albrecht, A. Bhirud, and A. Vahdat. Using Random Subsets to Build Scalable Network Services. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, March 2003.

[13] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using and Overlay Mesh. In *Proceedings of the 19th ACM Symposium on Operating System Principles*, October 2003.

[14] K. S. Park and V. S. Pai. Deploying large file transfer on an http content distribution network. In *Proceedings of the First Workshop on Real*, Large Distributed Systems (WORLDS '04), 2004.

[15] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems. In *Middleware 2001*, November 2001.

[16] A. Rowstron, A. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The Design of a Large-scale Event Notification Infrastructure. In *Third International Workshop on Networked Group Communication*, November 2001.

[17] R. Sherwood, R. Braud, and B. Bhattacharjee. Slurpie: A Cooperative Bulk Data Transfer Protocol. In *Proceedings of IEEE INFOCOM*, 2004.

[18] A. C. Snoeren, K. Conley, and D. K. Gifford. Mesh-Based Content Routing Using XML. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, October 2001.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, *ACM SIGCOMM 2001*, August 2001.

[20] A. Young, J. Chen, Z. Ma, A. Krishnamurthy, L. Peterson, and R. Y. Wang. Overlay mesh construction using interleaved spanning trees. In *Proceedings of IEEE INFOCOM*, 2004.

[21] A. Brown, A., E. Kissel, M. Swany, G. Almes, Phoebus: A Session Protocol for Dynamic and Heterogeneous Networks. *Technical Report 2008:334*, University of Delaware, 2008

[22] E. Kissel, A. Brown, M. Swany, Improving GridFTP Performance Using the Phoebus Session Layer. *SC 2009 Conference, Proceedings of the ACM/IEEE*, 2009.

[23] C. P. Guok, D. W. Robertson, E. Chaniotakis, M. R. Thompson, W. Johnston, B. Tierney. "A User Driven

Dynamic Circuit Network Implementation", *DANMS 2008, IEEE* 2008.

[24] N. Rao. Netlets: End-to-end QoS mechanisms for distributed computing over internet using two-paths. *Int. Conf. on Internet Computing*, 2001.

[25] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277.

[26] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. The case for resilient overlay networks. In *8th Annual Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, May 2001.

[27] NERSC Data Transfer Nodes. http://www.nersc.gov/nusers/systems/datatran/.

[28] Iperf http://sourceforge.net/projects/iperf/.

[29] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster.The Globus Striped GridFTP Framework and Server. *Proceedings of Super Computing 2005 (SC05),* November 2005.

[30] G. Khanna, U. Catalyurek, T. Kurc, R. Kettimuthu, P. Sadayappan, I. Foster and J. Saltz. Using Overlays For Efficient Data Transfer Over Shared Wide-Area Networks. *Proceedings of the 2008 ACM/IEEE conference on Supercomputing (SC 2008)*, November 2008

[31] W. Allcock and J. Bresnahan, "Maximizing Your Globus Toolkit™ GridFTP Server," in *ClusterWorld*, vol. 2, pp. 1—7, 2004.

[32] OSCARS Project Website http://www.es.net/OSCARS/.

[33] Phoebus Project Page, http://damsl.cis.udel.edu/projects/phoebus/about.php.

[34] W. Allcock, J. Bresnahan, R. Kettimuthu, J. Link. The Globus eXtensible Input/Output System (XIO): A protocol independent IO system for the Grid. *Proceedings of the Joint Workshop on High-Performance Grid Computing and High-Level Parallel Programming Models held in conjunction with International Parallel and Distributed Processing Symposium (IPDPS 2005)*, April 2005.

[35] M. Allman, V. Paxson, and W. Stevens, TCP Congestion Control. *RFC. RFC Editor.*, 1999

[36] G. Bell, J. Gray, A. Szalay, "Petascale Computational Systems," *Computer*, vol. 39, no. 1, pp. 110-112, Jan. 2006.

[37] PerfSONAR http://www.perfsonar.net/

[38] Atlas Experiment http://atlas.ch/

[39] Compact Muon Solenoid (CMS) Experiment http://cms.web.cern.ch/cms/index.html