MNEMONIC: A Network Environment for Automatic Optimization and Tuning of Data Movement over Advanced Networks

Patrick McClory, Ezra Kissel, Martin Swany, Michela Taufer
Department of Computer & Information Sciences
University of Delaware, Newark, DE 19716
{*mcclory, kissel, swany, taufer*}*@cis.udel.edu*

***Abstract*— Computational science is considered the third pillar of science but while we continue to make significant strides in high-performance computing, the ability to transfer the vast amounts of data generated and consumed by scientists has fallen behind. This divergence has significantly constrained scientific progress.**

This paper describes MNEMONIC, an end-to-end cyber-infrastructure system designed to improve scientists' productivity by automatically and transparently optimizing and tuning data movement over the network. MNEMONIC integrates advanced technologies, i.e., network interfaces, protocols, and measurement toolkits, to seamlessly manage high-performance data movement. We have integrated the Java implementation of GridFTP in MNEMONIC to provide a cross-platform data transfer capability. Preliminary results show that MNEMONIC+GridFTP outperforms the TCP+GridFTP implementation by over a factor of 10x, while remaining easy to install and operate.

I. Introduction

The scientific community is approaching a crisis. Despite constant effort and development, reasonable network performance is still unobtainable for most users. The so-called "hero gap" refers to the fact that tuning network throughput is possible only with significant effort, including data placement and tuning of hosts and networks. While existing network capabilities are constantly improving, it is often difficult or beyond the reach of many in the research community to take full advantage of these improvements. The Department of Energy's 2003 report *DOE Science Networking Challenge: Roadmap to 2008* noted the growing "End-to-End (E2E) Performance gap between backbone capacity and application throughput using Transmission Control Protocol (TCP)." Despite this warning over five years ago, the issue has still not been adequately addressed.

To undertake this challenge, we present MNEMONIC, an end-to-end network middleware environment designed to significantly improve scientific productivity by addressing the data transfer needs of the research community transparently and without manual configuration of the edge nodes. MNEMONIC helps scientists by automatically optimizing and tuning data movement over advanced networks. The MNEMONIC system composes various emerging network services into a new network paradigm in which the responsibility for assuring good application performance is not with the edge nodes, but with the network and its services. In this paper, we refer to this as an Optimizing Network Environment, or *ONE*.

Today users must manually perform host tuning in order to achieve good wide-area network performance. A quick web search will report many sites that give the details of how to tune system variables to improve performance. Users need to know the time it takes a packet to arrive at the destination and the bandwidth available, do a few calculations, apply some heuristics, and change some system variables. The steps involved are relatively easy, but it is manual, time consuming, and requires some understanding of networking. When placed in perspective, it is simply amazing that we accept that this is how it must be. A decade ago, mechanisms began to be developed to do some amount of protocol tuning automatically [28], and Linux and Windows support doing so, yet it is obvious that the problem is not solved. The MNEMONIC approach centers around the idea that the network itself, or network-centric services co-located with network devices, should help users use the network to their benefit.

The MNEMONIC vision is a synthesis of emerging software and network elements into a network "inlay" that can provide high-performance data movement as a *network service*. By integrating a variety of existing technologies into a unified framework, MNEMONIC aims to enable broad access to network features without burdening users with the details of the underlying network infrastructure and protocol behavior. In particular, we claim that a *ONE* can be constructed using the MNEMONIC components that will provide reasonable

978-1-4244-4581-3/09/$25.00 ©2009 IEEE

network throughput with no tuning.

To assess our claim, we consider in this paper a typical cross-platform data transfer that scientists often install and use when dealing with data across the network. In particular, we consider two scenarios, one in which we use the plain transfer tools including the Java CoG in GridFTP and a second in which these tools are integrated in MNEMONIC. Preliminary results presented here show that MNEMONIC+GridFTP outperforms the vanilla GridFTP implementation by over a factor of 10x, providing high-performance data movement for scientists using distributed computing resources.

The rest of the paper is organized as follows: Section II presents the MNEMONIC components; Section III shows the performance measurements with and without a prototype MNEMONIC system in a simulated environment; Section IV discusses related work; and Section V concludes this paper.

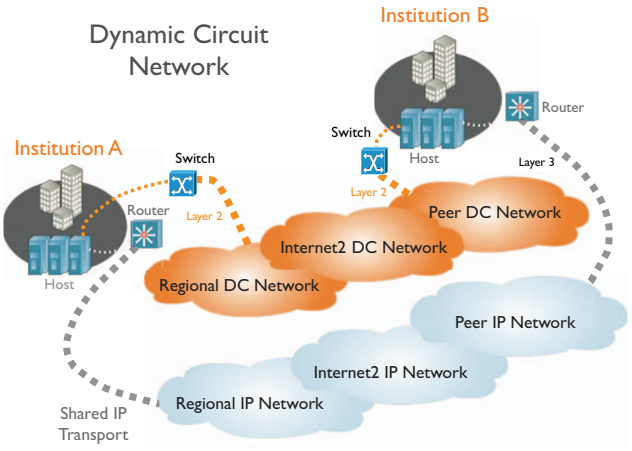


Fig. 1: *Backbone and dynamic circuit networks like those deployed by Internet2*

II. Framework

MNEMONIC integrates various existing technologies to form a unified system to seamlessly manage high-performance data movement. MNEMONIC technologies include: integration of dynamic networks, a session-layer based network adaptation environment (***Phoebus***), an extensible session layer protocol (***ESP***), a portable advanced file transfer interface (***jTopaz***), and a network measurement and metric environment (***perfSONAR***) for monitoring and diagnosing network performance problems.

Together, these system components provide the proven building blocks for a *data movement* management system that can automatically *direct, adapt, and optimize* data flow in advanced, high-performance networks in a way that is transparent to the end user. The following describes each of these components in further detail.

A. Dynamic Networks

Dynamic Network Environments (DNEs) allow network resources to be provisioned "on the fly" by users, services, and advanced applications. Demand-driven allocation of these ephemeral, dedicated links and paths enables unprecedented optimization of network utilization and is an ideal tool for demanding network applications. These networks currently support high performance and Grid computing applications that must reliably and quickly move large quantities of data, such as those supporting the Large Hadron Collider (LHC) in Switzerland.

Dynamic circuit (or bandwidth-on-demand) networks are being deployed by more and more research and education (R&E) networks. Major R&E networks like Internet2, the US Department of Energy's (DOE) ESnet, and GÉANT2 in Europe have led the way in developing and deploying this functionality. Many high-end applications are currently utilizing this infrastructure. Figure 1 depicts this architecture as realized in the Internet2 DCN as it is used in MNEMONIC. As shown, this network infrastructure exists in parallel with the existing shared connectivity.

Currently, Internet2's Dynamic Circuit Network (DCN), ESnet's Science Data Network (SDN) and GÉANT2's AutoBAHN are providing pre-production DNEs that share a commonly-developed protocol refered to as the InterDomain Controller Protocol (IDCP). With other international partners joining, it is possible to dynamically reserve network resources around the world. In the US, the common software base is using components based on the ESnet OSCARS [24] project and the DRAGON [20] project.

While many demanding applications are currently making use of this functionality, there are significant configuration requirements for the use of such networks. These networks are tremendously powerful, but they are not automatic or particularly easy to adapt for use. The other MNEMONIC technologies presented below address this challenge.

B. Phoebus

Phoebus is a system that can dramatically improve end-to-end throughput by forwarding data via intermediaries, called Phoebus Gateways (PGs), placed at strategic locations in the network. Phoebus is a follow-on to the Logistical Session Layer (LSL) [29], which used a similar protocol and mechanism. Various experimental versions of Phoebus have been deployed and tested in Internet2's previous networks, Abilene and HOPI, as well as in GÉANT2, ESnet, RNP in Brazil, and on links between the US and Korea.

Phoebus provides protocol optimization, translation, and data buffering, and these features help address the well-known problems with the Internet Protocol (IP) suite's Transmission Control Protocol (TCP) in long-distance, high-bandwidth networks [23]. The operation of Transport-layer protocols such as TCP is central to network performance issues and efforts. There are many approaches (some described in Section IV) which postulate that modifications to TCP's congestion control model will address the problem, and others who believe that alternative transport protocols, like UDP-based Data Transfer for High-Speed Wide Area Networks (UDT) [13], can be used to work around the problem. While these approaches have their benefits, none is by any means a panacea. Perhaps most serious of all, they must all be deployed at the "edge" of the network; putting this burden on users and network administrators will slow adoption of any solution (in the event that a general one can be found).

The Phoebus component is critical to the MNEMONIC system in that it offloads the burden of assuring network performance from the end-systems to the network itself and acts as an "on-ramp" to DNEs.

C. Extensible Session Protocol

The Extensible Session Protocol (ESP) resides in the Session Layer (Layer 5) in terms of the ISO protocol model, a layer above TCP. While defined by the ISO [9], the Session Layer does not really exist in the Internet architecture, with the exception of the Session Initiation Protocol [27] (SIP), which as its name implies, only deals with the creation of a session – generally for voice or video.

The current Internet model binds all end-to-end connections to a transport protocol such as TCP. The ESP model binds this communication to a Session protocol. Thus, ESP is able to explicitly manage the heterogeneity in network environments by conceptually breaking the end-to-end connection into a series of connections, each potentially spanning a different network segment. This also encompasses enabling applications to signal the control plane of dynamic, service-rich networks such as DCNs. Applications can provide the network with information about the requirements, purpose, and characteristics (e.g. duration) of the data transfer to allow the network to adapt itself accordingly. A direct application of ESP is Phoebus, which is a Session-layer based network adaptation environment that uses ESP internally.

The ESP implementation also features a library that is compatible with the standard "Sockets" network Application Programming Interface (API). This means that applications can be trivially adapted to use ESP, and thus Phoebus, DCN, and thus, MNEMONIC. In some cases, applications can be modified at runtime, without recompilation, using a dynamic library loading technique. When the application source is modified, it requires only a simple textual substitution initially and then additional control is available when desired. This enables transparent enablement of existing applications to interact with the control plane.

D. jTopaz

jTopaz is a Firefox browser "add-on" that provides a user-friendly and portable interface to high-performance data movement systems [3], [33]. As a browser plugin, jTopaz is extremely easy to install and use. It offers a standard, intuitive two window file copy interface. jTopaz originally focused on GridFTP [12] but we are now investigating extending it to support other data transfer systems like SCP and SFTP [10].

The GridFTP support in jTopaz makes use of the CoG [19] Java implementation of GridFTP. This version is compatible with the high-performance compiled versions of GridFTP, although lacking in some features. In contrast to the native version, it installs trivially and runs on all platforms that support Java. The Java implementation of GridFTP is thus quite portable, but this portability comes with some performance penalty presented in Section III.

In MEMONIC's optimizing network environment, this Firefox component acts as the interface into the system. By providing a simple to install and intuitive interface that can automatically enable high-performance file transfers, we believe it will truly facilitate broader access.

E. perfSONAR

perfSONAR [14] is a framework that enables network performance information to be gathered and exchanged in a multi-domain, federated environment. The goal of perfSONAR is to enable ubiquitous gathering and sharing of this performance information to simplify management of advanced networks, facilitate cross-domain troubleshooting and to allow advanced applications to tailor their execution to the state of the network. This system has been designed to accommodate easy extensibility for new network metrics and to facilitate the automatic processing of these metrics as much as possible.

The aim of perfSONAR is to create framework allowing a variety of network metrics to be gathered and exchanged in a multi-domain, heterogeneous, federated manner. perfSONAR is targeting a wide range of use cases. For example, current use cases include: collection and publication of latency data, collection and publication of achievable bandwidth results, publication of utilization data, publication of network topology data, diagnosing performance issues, and several others. While

perfSONAR is currently focused on publication of network metrics, it is designed to be flexible enough to handle new metrics from technologies such as middleware or host monitoring.

perfSONAR has been widely deployed in R&E networks worldwide. The network topology and performance components of perfSONAR provide the input to the route computation in the MNEMONIC framework, provding the input to the Phoebus routing tables. Due to space constraints, this topic is not discussed further in this paper. We mention it here as it is a key part of the architecture and to address whether such a network performance system could be widely deployed.

F. MNEMONIC

The complete MNEMONIC system provides an Optimizing Network Environment (ONE) that is easy to use and leverages advanced networks, like the DCN, and network services like Phoebus and perfSONAR. ESP allows for those resources to be signaled transparently, while providing a familiar interface and socket-compatible library. perfSONAR provides network topology and performance information for route construction and for hints about resource configuration.

Together, the MNEMONIC system provides a state of the art environment for data transfer. As mentioned, many of these components are already in use in R&E networks worldwide. In this paper, we merely evaluate the feasibility of deploying a zero-configuration browser plugin that can provide reasonable throughput using the MNEMONIC framework.

III. Performance

In this section we consider two scenarios, one using the plain Java GridFTP implementation and a second using MNEMONIC, to support the file transfer across the Internet. When using MNEMONIC, the Java GridFTP implementation is integrated in our environment and is supported by its components, i.e., ESP, Phoebus, and jTopaz. We compare the two scenarios in terms of the performance that a Java client code achieves when sending data to a server at a different Internet site.

Note that the time it takes to allocate a circuit in a DNE is not represented in these performance measurements. First, the cost is the same for the MNEMONIC cases and without. Second, DNEs can configure longer-lived circuits based on demand and use MNEMONIC as a transparent on-ramp to them. Finally, the time to allocate a circuit is decreasing as the technology matures, so actual values from today's deployed software do little to validate or invalidate our basic hypothesis.

A. Test Configuration

Our goal is to quantify the performance increase of real applications when using a prototype MNEMONIC system. To guarantee the repeatability of our experiments under different network configurations, we emulate a range of network conditions in a controlled environment using *netem* [1], a Linux kernel module. This module enables modification of how packets are handled by outgoing IP interfaces. It can buffer packets to create artificial latency as well as cause loss of packets. For our testing environment, we use the *netem* module to emulate various distances and loss rates.

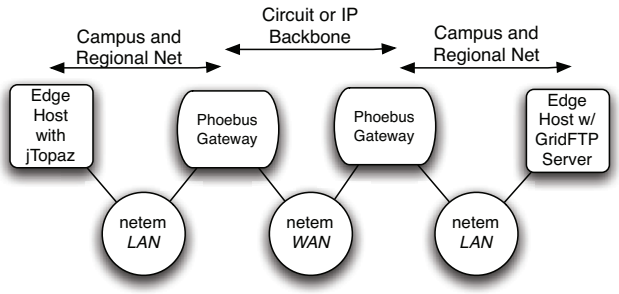


Fig. 2: *MNEMONIC testbed showing simulated networks between Phoebus Gateways*

Figure 2 illustrates our experimental setup. Two end hosts are used as the GridFTP source and destination. There are also two Phoebus gateways, one at either side of the backbone network. The *netem* module suggests using the module on the same host as the application sending or receiving data. This requires us to add three hosts in between the aforementioned hosts to function as *netem* forwarding nodes. These nodes are configured to forward the data while transparently applying the latency and loss modifications.

The environment allows us to test direct end-to-end connections as well as connections using Phoebus with exact same paths, guaranteeing the same network conditions. A typical network path generally traverses a Local Area Network (LAN) with low latency and some congestion-induced, and a Wide Area Network (WAN) that traverses some distance but has little loss. In the case of a dynamically allocated circuit over DCN, there is no competition, so there will be no loss. Our experiments reflect this behavior with LAN loss of 0.001%, low LAN latencies ranging from 10ms to 25ms, and WAN latencies between 50ms and 100ms. These were chosen to model a connection traversing a campus LAN and a regional network to reach a POP of a national backbone. For instance, a current measurement from U. Delaware to UC Santa Barbara is $\sim$80ms, with a backbone traversal of $\sim$60ms, with $\sim$10ms on each end. Similarly, it is around 120ms from U. Delaware to CERN in Switzerland. While not exact, these cases guided our choice of delay parameters.

In these tests, the end systems were configured with

TCP buffers of 16MB/#streams. While we keep the loss constant (.001%), we test different type of loss, i.e., independent loss (packets that get dropped are selected randomly) and dependent or packet burst loss (a packet dropped makes it more likely the next packet is also dropped). In all of our tests, the GridFTP source runs either the Windows or Linux Java client and the destination is configured with the C version of a MNEMONIC-enabled GridFTP server. The direction of data flow for all transfers is from the server to the client.

B. Results

Each experiment is executed 10 times and the values presented here are average values. For the experiments we transfer from /dev/zero (which just sends 0's as long as it has being read) on the server node to the client node for a specified amount of time (i.e., 10 sec, 20 sec, 30 sec, 45 sec, 60 sec). Different times were used to demonstrate different transfer sizes, but in this case, it is easier to vary size than absolute data amount. For each experiment we calculate the speed, in Megabits/second (Mb/s), based on the amount of data transferred and the transfer duration.

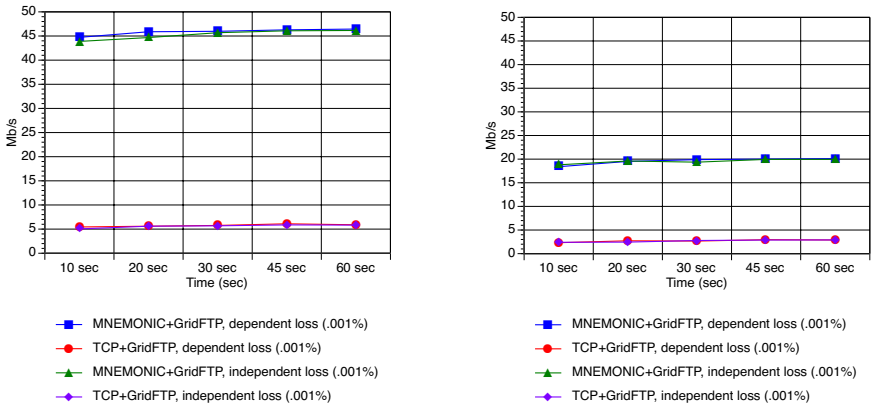
Figure 3 demonstrates the performance of MNEMONIC+GridFTP versus TCP+GridFTP running in the wide-area emulation testbed described above. Note that we are using Java GridFTP. Figure 3(a) shows the performance of a client running on an untuned Windows XP system with a simulation environment of 10ms of LAN latency with 0.001% loss and a WAN latency of 50ms and Figure 3(b) shows the same experiments with 25ms of LAN latency and 100ms of WAN latency. These cases show performance for a Windows user using the Topaz client for interacting with the rest of the MNEMONIC framework without any other modifications or tuning of the end host. Figure 4(a) and (b) demonstrate the same performance of MNEMONIC+GridFTP versus TCP+GridFTP for a Windows XP environment tuned with a TCP window size of 16 MB and Figures 5(a) and (b) show the performance for a Linux environment tuned with a TCP window size of 16 MB. Overall these results show that a portable Java file transfer application can achieve very good performance in the MNEMONIC environment with little or no user intervention. For both Windows and Linux end hosts, MNEMONIC+GridFTP outperforms TCP+GridFTP with data transfer speeds that are up to 10x faster. The gain in performance is particularly noticeable for Windows hosts, both when tuned and untuned configurations are used.

IV. Related Work

Most of the work related to the core goals of MNEMONIC involves network protocols. The pervading opinion regarding a solution to network performance issues is that we need new or improved end-to-end transport protocols. A great deal of effort has gone into modifying the congestion control in the Transmission Control Protocol (TCP), which is the dominant Transport protocol in the Internet. FAST TCP [15], [32] has demonstrated good performance in long distance, high performance networks, but is not available since its commercialization. Other variants like Highspeed TCP [8], Hamilton TCP [21] and Vegas TCP [6] are available in Linux, but even then must be explicitly enabled. This only scratches the surface of the TCP efforts, but all of these approaches face significant deployment problems. They simply have not had an impact for the majority of network users. Other clearly related work involves dynamic network resource allocation. Systems like Terapaths [11] and LambdaStation [4] and VINCI [31] are all approaches related to the DCN/SDN approach, which uses the OSCARS system [24]. These groups are collaborating and as their approaches move closer together, we will be able to take advantage of their advances. Another major class of related work involves breaking end-to-end connections similar to the Phoebus approach. Performance enhancing proxies (PEPs) [5], [26] perform a similar function, but generally attempt to be transparent and are not controlled with a Session protocol. TCP splicing [22] is understood to improve performance in certain cases. Overlay networks with similar goals have been constructed [2], [17], [25], [30]. In each of these overlay architectures, either a specific application or a smaller subset of network performance issues is targeted. In contrast, MNEMONIC provides a more general and holistic approach that allows a broad range of applications to reap the benefits. Our previous work [18] evaluated the Phoebus component of MNEMONIC on a similarly configured testbed where we demonstrated increased performance with Linux/C versions of GridFTP under a variety of network conditions. GridFTP itself has also seen development in order to bring it onto the Windows platform [7] as well as provide performance-enhancing interfaces [16]. We envision that both of these developments may be integrated with our proposed architecture.

V. Conclusion

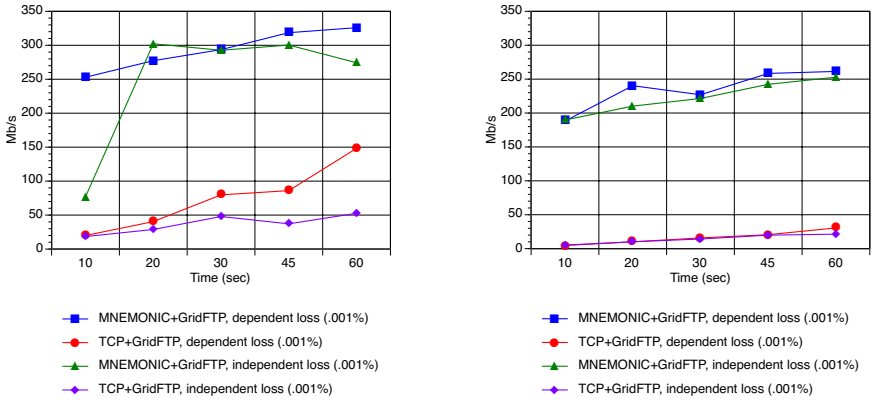
In this paper we present the architecture of a dynamic data movement system called MNEMONIC. Although MNEMONIC may resemble what are referred to as overlays, it is actually more of a network "inlay" that incorporates services into the network fabric. Preliminary results presented in the paper using a first prototype of MNEMONIC consider different network configurations with different latencies and types of loss as well as different operating systems. The results show a significant



(a) Windows untuned with 10ms LAN - 50ms WAN, 0.001%LAN loss

(b) Windows untuned with 25ms LAN - 100ms WAN, 0.001%LAN loss

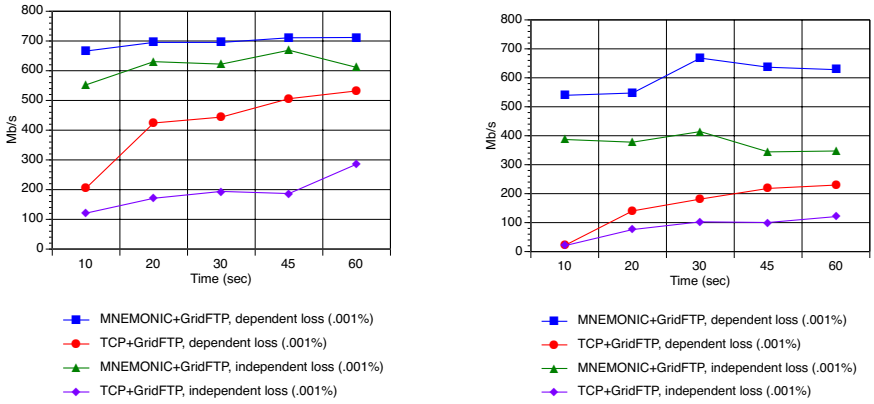
Fig. 3: MNEMONIC+GridFTP vs. TCP+GridFTP on untuned Windows end hosts



(a) Windows tuned with 10ms LAN - 50ms WAN, 0.001%LAN loss

(b) Windows tuned with 25ms LAN - 100ms WAN, 0.001%LAN loss

Fig. 4: MNEMONIC+GridFTP vs. TCP+GridFTP on tuned Windows end hosts



(a) Linux tuned with 10ms LAN - 50ms WAN, 0.001%LAN loss

(b) Linux tuned with 25ms LAN - 100ms WAN, 0.001%LAN loss

Fig. 5: MNEMONIC+GridFTP vs. TCP+GridFTP on tuned Linux end hosts

increase in user throughput (up to 10x faster) and yet do not require any host tuning.

MNEMONIC's preliminary results are very promising and suggest that the deployment of our system can capture the power of today's networks and make it accessible to scientists, helping them solve the data transfer issues that still inhibit their endeavors. At the same time, MNEMONIC builds a bridge that will allow continued innovation in network protocols and technology.

VI. Acknowledgements

This work was supported by the National Science Foundation under grants OCI-0802650 and OCI-0721902 and by Internet2. The Phoebus system was supported by the Department of Energy under grant FG02-04Er25642.

Thanks to Jeff Boote, Eric Boyd, Aaron Brown, Andy Lake, John Vollbrecht, Matt Zekauskas and Jason Zurawski of Internet2, and to the ESnet OSCARS and perfSONAR teams, for many contributions to this overall architecture and for their development efforts on the various components.

References

[1] Net:netem. `http://www.linux-foundation.org/en/Net:Netem`.

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

[3] K. Bhatia, B. Stearn, M. Taufer, R. Zamudio, and D. Catarino. Extending Grid Protocols onto the Desktop using the Mozilla Framework. In *2nd International Workshop on Grid Computing Environments (GCE)*, 2006.

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

[5] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. Performance enhancing proxies intended to mitigate link-related degradations. RFC 3135, June 2001.

[6] L. Brakmo, S. O'Malley, and L. Peterson. Tcp vegas: New techniques for congestion detection and avoidance. In *Proceedings of the SIGCOMM Symposium*, 1994.

[7] J. Feng, L. Cui, G. Wasson, and M. Humphrey. Toward seamless grid data access: Design and implementation of gridftp on .net. In *Proceedings of GRID '05*, 2005.

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

[9] A. N. S. for Information Processing Systems. Open systems interconnection – basic connection oriented session protocol specification. ANSI/ISO 8327-1987, 1992.

[10] J. Galbraith and O. Saarenmaa. SSH file transfer protocol. Internet Engineering Task Force, INTERNET-DRAFT, draft-ietf-secsh-filexfer-13.txt, 2006.

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

[12] GridFTP. `http://www.globus.org/datagrid/gridftp.html`.

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

[14] A. Hanemann, J. Boote, E. Boyd, J. Durand, L. Kudarimoti, R. Lapacz, M. Swany, S. Trocha, and J. Zurawski. PerfSONAR: A service oriented architecture for multi-domain network monitoring. In *In Proceedings of the Third International Conference on Service Oriented Computing (ICSOC 2005)*, ACM Sigsoft and Sigweb, pages 241–254, December 2005.

[15] C. Jin, D. Wei, and S. Low. Fast tcp for high-speed long-distance networks. Internet Engineering Task Force, INTERNET-DRAFT, draft-jin-wei-low-tcp-fast-01, 2003.

[16] R. Kettimuthu, W. Allcock, L. Liming, J. Navarro, and I. Foster. Gridcopy: Moving data fast on the grid. In *Proceedings of the Fourth High Performance Grid Computing Workshop to be held in conjunction with International Parallel and Distributed Processing Symposium (IPDPS 2007)*, 2007.

[17] 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. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12, Piscataway, NJ, USA, 2008. IEEE Press.

[18] E. Kissel, A. Brown, and M. Swany. Improving GridFTP Performance Using the Phoebus Session Layer. Technical report. UDCIS 2009:337, `http://damsl.cis.udel.edu/projects/phoebus`.

[19] G. V. Laszewski, I. Foster, and J. Gawor. Cog kits: A bridge between commodity distributed computing and high-performance grids. In *in ACM 2000 Java Grande Conference*, pages 97–106, 2000.

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

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

[22] D. Maltz and P. Bhagwat. Tcp splicing for application layer proxy performance, 1998.

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

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

[25] P. Rizk, C. Kiddle, R. Simmonds, and B. Unger. Performance of a gridftp overlay network. *Future Gener. Comput. Syst.*, 24(5):442–451, 2008.

[26] P. Rodriguez, S. Sibal, and O. Spatscheck. TPOT: Translucent proxying of TCP, 2000.

[27] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, and M. H. andE. Schooler. Sip: Session initiation protocol. RFC 3261, June 2002.

[28] J. Semke, J. Mahdavi, and M. Mathis. Automatic TCP buffer tuning. In *SIGCOMM*, pages 315–323, 1998.

[29] M. Swany. Improving Throughput for Grid Applications with Network Logistics. In *Supercomputing 2004*, November 2004.

[30] J. Turner, P. Crowley, J. Dehart, A. Freestone, B. Heller, F. Kuhms, S. Kumar, J. Lockwood, J. Lu, M.Wilson, C. Wiseman, and D. Zar. Supercharging planetlab - high performance, multi-application, overlay network platform. In *SIGCOMM 2007*, 2006.

[31] VINCI: Virtual Intelligent Networks for Computing Infrastructures. `http://monalisa.caltech.edu/`.

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

[33] R. Zamudio, D. Catarino, M. Taufer, K. Bhatia, and B. Stearn. Topaz: Extending Firefox to Accommodate the GridFTP Protocol. In *4th High-Performance Grid Computing Workshop (HPGC)*, 2007.