# Phoebus: A Session Protocol for Dynamic and Heterogeneous Networks

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

**Guy Almes**
Texas A&M University, College Station, Texas 77843
*almes@tamu.edu*

## Abstract

*In this paper we present a session protocol and infrastructure for high-performance, long-distance networks. Phoebus allows the creation of data transfer sessions that are not bound directly to a single end-to-end transport connection. Using Phoebus "Gateways" as middleboxes, we can use segment-specific transport protocols and can hide the allocation details when utilizing dynamic network resources. We demonstrate performance improvement across Internet2's IP and Dynamic Circuit (DC) HOPI testbed.*

## 1 Introduction

The Internet protocol suite has been tremendously successful in providing a basis for universal connectivity. Science and industry are increasingly global endeavors as ubiquitous, reliable, high-performance Internet connectivity enables incredible connectivity. As the Internet continues to evolve, we must continue to ask whether this basis is still sufficient to provide for the breadth of services that this connectivity promises to provide in nearly every facet of our lives. While the current architecture has served us well, we believe that an additional layer of network infrastructure, and associated protocols, can provide important functionality that naturally augments the current Internet model and allows it to grow to meet demands. This paper describes an architecture to realize that goal.

Today, a typical end-to-end path through the Internet traverses a variety of technologies, each of which have unique characteristics. Everything from wireless to high-speed Ethernet, from optical to satellite links, can be utilized as data flows from source to destination. These networks often have dramatically different characteristics and the interaction between them can lead to less than desirable performance. Reliable backbone networks with low loss and long latency, combined with shared access networks with some loss and low latency, can interact to cause performance problems. A typical connection may cross multiple domains and these interactions affect the throughput of the connection. The current Internet model must simply ignore the differences in networks and trust the end nodes to address them as best they can.

As the variety of network links has increased, so has the need for high-speed communication between nodes separated by long distances. Grid computing has been enabled by high-performance networking but has in turn placed demands on the network in terms of moving large amounts of data over potentially large distances. For instance, the infrastructure surrounding the Large Hadron Collider provides an excellent example of the extreme distances that data often has to move. From its location in Switzerland, the data will be used by thousands of scientists who may be located in the most distant reaches of the Internet. This is simply the first of many large scale projects to come that will require the reliable, deterministic transport of massive amounts of data. Telemedicine, radio astronomy, and even the distribution of high-definition video content will place considerable demands on the current network architecture. This increase in both distances and network heterogeneity, which exacerbate inherent performance issues, inspires a rethinking of the way in which data is transported.

The end-to-end argument [19] has, for decades, provided a conceptual basis for the functionality of transport protocols. The common interpretation of this argu-

ment indicates that the core of the network should remain simple, and that all functionality, beyond merely forwarding packets, should be handled by the end hosts. This line of thinking has permitted the Internet to successfully expand from thousands of nodes to millions.

There are some flaws in an absolutist interpretation of the end-to-end argument, however. The "black box" view of the network has forced transport protocols to expend significant effort trying to estimate the current state of the network. When a router has large queues of data to send, it must start dropping packets. However, packets can also be lost due to other factors in the network. Thus, when the end hosts detect packet loss, the algorithms make implicit assumptions as to why the packet was lost, and they must react to mitigate potential congestion. Over the years a number of methods have been proposed to improve the transport layer's ability to discern congestion related loss and react accordingly.

The Phoebus project has sought to encourage a paradigm shift in the way traditional edge and backbone networks are utilized in order to improve end-to-end throughput over long distances. By augmenting the current Internet model with an additional service layer, Phoebus embeds "intelligence" in the network that allows a connection to become articulated and adapt to the environment on a segment by segment basis. The system includes a protocol and software infrastructure that can address some of the fundamental issues in long distance data movement.

Internet2 is in the process of deploying a prototype of this system to act as a gateway to their new Dynamic Circuits (DC) Network (see `http://www.internet2.edu/network/dc`.) The Internet2 DC network is a hybrid optical and packet infrastructure in which users can configure short-term, dedicated, point-to-point circuits for demanding applications. Phoebus provides a gateway to these services for legacy applications and is able to transparently improve achievable throughput in many cases.

In this paper, we discuss the architecture of the Phoebus system and its use in the Internet2 network. We also discuss how we can easily allow existing applications to use our new infrastructure. Finally, we show some of the performance improvements that are possible by detailing some recent experimental results.

## 2 Problem Statement

In cases where the network exhibits heterogeneity, a single end-to-end transport protocol will have difficulties. The Internet2 network is a prime example of the disparity between the edge, or "last mile", connections and the Internet backbone. Through proactive monitoring and improvements, the Internet2 network engineering staff has been able to reduce the loss along the backbone to almost zero. However, even with all the work spent tuning Internet2, the end users may not have noticed as significant an increase in their throughput as compared to the improvements seen in the core. This is due, in general, to problems occurring not in the backbone network, but in the users' networks and applications.

### 2.1 End User Problems

The first problem that many end users run into is that they are not skilled network administrators. Large institutions may be able to hire staff to help users tune their applications for the network conditions that they encounter. However, smaller institutions, average consumers, small businesses, etc., do not have this luxury. Thus, if a user wishes to transmit a large file to a colleague across the country, a good understanding of how to manually tune the connection to ensure performance is necessary. This configuration issue is exacerbated because, in order to achieve high throughput, the entire protocol stack, from transport protocol to Ethernet driver must be tuned. Even if the end user is able to put in the effort to tune the machine, many applications do not actually set the parameters. This can be worked around by setting the default parameters. However, setting defaults that will work effectively over a long distance will waste memory and processor time for shorter distance connections or connections of lesser importance.

The end user's applications and machine are not the only cause of low throughput for end users. For instance, many backbone providers can devote significant time, effort and money to ensuring that their infrastructure is high-speed and low-loss. The end users' institutions, however, don't necessarily have the resources to address the problems of reasonably high loss rates and low bandwidth for the small percentage of their users that require these features. Thus, it is not uncommon in end user networks to see problems such as duplex mismatch errors, low MTUs, faulty equipment and high loss rates. The untuned network problem can occur on both ends of a connection. By mitigating the effect that the end networks have on the end-to-end connection, significant performance increases could be had. However, as an end-to-end protocol, TCP on its own cannot mitigate the problem of the edge networks.

## 3 Related Work

There have been a number of projects that attempt to work around the problems presented in the previous section. Phoebus is similar in spirit to work in application level routing (or overlay networks) and non-defaut route

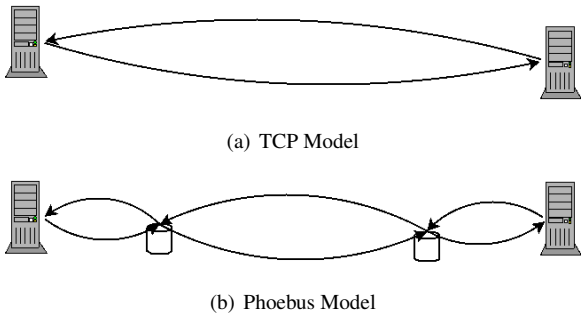
selection [4, 18, 20]. This work has addressed a number of issues including route asymmetry and optimal, or parallel, route selection. Phoebus differs in that it is presented as an evolution of the Internet architecture, rather than a workaround for ineffective routing policy.

Performance enhancing proxies (PEPs) are described in RFC 3135 and are commonly used to handle the issues that occur when using TCP over wireless or satellite links. However, the focus of PEPs has been to address performance problems on specific, less common links and does not provide a general solution.

To a certain extent, our results are based on the nature of TCP's flow control. The research community has recognized the issues inherent in using TCP over networks with high bandwidth/delay products. There is a tremendous body of research [7, 11–13, 13–15], too vast to poperly cite here, devoted to understanding and improving TCP's performance. TCP's performance is widely understood to depend greatly on the end-to-end Round-Trip-Time (RTT). We leverage the well undertood stability and fariness qualities of TCP by utilizing it between our own adaptation points in the network. Whereas more drastic changes to TCP (such as Explicit Congestion Notification) may take time for their ramifications to be fully understood (and to be deployed ubiquitously), Phoebus is treading in more familiar territory.

The NSF-funded DRAGON (Dynamic Resource Allocation via GMPLS Optical Networks) project which includes collaborators from MAX, University of Southern California (USC) Information Sciences Institute (ISI) East, and George Mason University attempts to bridge optical and packet networks [6]. The DRAGON architecture allows edge hosts to create a high-speed packet-switched network link between two points. While this provides a way to allow applications to more effectively use high performance optical networks with no changes to the application itself, it does not resolve the performance issues inherent to long-distance, high-bandwidth connections. Any lost data must be retransmitted from the source, still forcing redundant long-distance network traversals. Also, the mode necessitates significant configuration of the end hosts to be able to signal and subsequently utilize the established connection.

The Internet Backplane Protocol (IBP) is a project from University of Tennessee with the goal of providing storage from nodes in the core of the network. Applications can store their data on these intelligent storage nodes to help increase the speed of transmission of data to other users, in similar fashion to the Akamai [3] caching model. However, due to IBP's focus on file-oriented data storage, IBP makes it difficult to transfer streams of data, which precludes its use in numerous applications and makes transparently adapting existing applications to use it more difficult.



(a) TCP Model

(b) Phoebus Model

**Figure 1. TCP Model vs. Phoebus Model**

I-TCP uses a protocol similar to that in Phoebus designed to handle the problems inherent in using TCP for communication with mobile devices [5]. Mobile devices often experience large amounts of loss in their normal communication with other hosts, which TCP was not designed to handle. I-TCP breaks a TCP connection into two sub-connections, a connection from the wireless node to the edge of the wired network and a connection from the edge of the wired network to the end host. This allows for applications to use a protocol optimized for the wireless link while still using TCP to connect to legacy end hosts. However, the focus of I-TCP was on bridging wireless and wired networking which means it has some requirements that don't make sense for a general purpose protocol.

Another method, Optical Burst Switching (OBS), is a way for optical networks to ensure better utilization while providing a packet-switched interface to users [17]. Conceptually, it is very similar to Phoebus. In OBS, packets from different flows are queued up. Once a minimum number of packets has been queued up, the hardware allocates a "burst" and sends the packets. By intelligently choosing the minimum and maximum amount of packets to queue up, the hardware can ameliorate the cost of session setup to ensure that paths are allocated only when the cost has been distributed over a large number of packets.

When using an OBS network, no buffering of the data is performed. Thus, if packet loss occurs after the data has traversed the OBS network, the lost data must again be retransmitted across the network, taking up valuable space inside a burst. This end-to-end nature also mandates that the end hosts be highly tuned to properly make use of the network.

## 4 The Phoebus Model

The idea underlying the Phoebus model is to embed Phoebus Gateways (PGs) in the network. These gate-

ways can be placed at the edges of backbone networks and function as store-and-forward buffers for user connections. Instead of a user making a direct connect to the end host, the user would send data via these PGs.

Initially, the end user makes a TCP connection to the first hop gateway on the edge of the backbone network. At this point, the user authenticates himself to the gateway and requests buffering service from the gateway. The edge gateway verifies that it is able to grant access to the specified user and checks whether or not the amount of buffering or bandwidth requested exceeds current resource limitations or the limitations imposed on the connecting user. If the gateway is unable to fulfill the user's request, it will inform him of the possible resource limits the user would need to abide by, or redirect him to another gateway that may be better able to serve him.

If the PG is able to meet the needs of the user, it accepts the connection and determines a suitable next hop. In the simplest case, the gateway simply looks up the edge PG closest to the destination host. This lookup could involve contacting a network measurement and topology service like perfSONAR [16], consulting routing table information, or performing a traceroute to find any PGs along the path between the the gateway and the destination host. Once the gateway has found the best downstream PG, it makes a connection to it. The upstream gateway then authenticates itself with the downstream gateway, and informs the downstream gateway on behalf of which user the upstream gateway is working. The upstream gateway then informs the downstream gateway of the user's requirements. If the downstream gateway is unable to meet the user's needs, the upstream gateway can begin searching for other gateways that are positioned close to the destination. If none are found, the upstream gateway informs the user that it is unable to handle his request.

If the downstream gateway is able to fulfill the request, the upstream and downstream gateways negotiate how they will exchange traffic. This would involve determining the best way to exchange the data between them which could involve tuned TCP, a more appropriate transport protocol or even a specially-allocated circuit. Once this decision has been made, the gateways establish a data connection between them. In addition, if a circuit-based connection is desired, the gateways still establish a conventional connection while the circuit connection is being established, thus hiding the signalling and allocation latency.

Once the data connection has been established, the downstream gateway makes a direct TCP connection to the destination host. The settings for the TCP connection can either be left as defaults or can be calculated on the fly by the gateway. Once a connection has been established, the downstream gateway informs the upstream gateway, who in turn informs the end user. At this point, the chain of connections is established and the two end hosts can communicate as though they had a direct connection.

The difference between the end-to-end connection can be seen in Figure 1. The top picture shows the TCP model where the end user connects directly to the destination host. The bottom picture shows the Phoebus model where there are three connections in series, utilizing two gateways.

## 4.1 Phoebus Benefits

### 4.1.1 Round-Trip Time

The first benefit is that the RTT of the individual connections is less than the RTT of the end-to-end connection. TCP is heavily dependent on the RTT, which means any decrease in the RTT can produce an increase in throughput. However, short of increasing the speed of light, the distance travelled by the packets will still need to be the same. By establishing two TCP connections, each having a shorter RTT, the speed of each of the shorter connections should be faster than the speed of the overall connection. If this is the case, the throughput of the overall connection in the Phoebus case should run close to the speed of the slowest individual connection.

### 4.1.2 Loss

Another benefit is in how the independent connections handle loss. When a loss occurs on an end-to-end connection, it causes a reduction in the throughput for the entire connection. However, with the shorter, independent connections, the throughput is only dropped for one portion of the connection. The others continue running at the same speed. The node with the prior to the slowed connection will simply buffer the new data while the slowed connection recovers. This works to prevent intermittent drops from substantially affecting the entire connection.

This will not, however, help much if substantial problems occur on one of the TCP streams. The buffers on the node on the sending side of that connection will begin to fill up due to the asymmetry in the speed with which its receiving and the speed with which it is able to send. Once the buffers on the node are full, the node ceases to read from the kernel, whose buffers also fill. When this happens, the receiving TCP offers no available window to the sender, thus using flow control to throttle the sender. At some point an equilibrium will be achieved, but the upstream connection will have slowed to this equilibrium point. If the downstream connection recovers, then the upstream connection will speed

up again when resources become available. If the disturbance occurs for long enough, the slowdown will propagate all the way back to the sender.

The biggest benefit with Phoebus, however, comes from the distance retransmissions need to travel.

With a normal end-to-end connection, the end user transmits data from her machine, across the country to the other machine. If loss occurs anywhere on the link, the end user is responsible for retransmitting the data back to the other machine. Thus, for every loss, the data must be retransmitted the full length of the connection, holding up any data that has already been received on the far side. Similarly, it requires a full round-trip time for the end user to realize that the packet has been lost and must be retransmitted.

In the Phoebus model, the end-to-end connection is broken into at least three segments: a short RTT connection between the end user and the edge network PG, the negotiated inter-PG connections and the connection from the far-side edge PG to the destination host. Also, in the Phoebus model, once a PG has received the data, it becomes responsible for ensuring that that data is received at the next hop. These two properties combine to provide for improved end-to-end performance.

TCP connections are highly dependent on the RTT to ensure good throughput [14]. The shorter a TCP connection's RTT is, the faster it can recover from loss. In the Phoebus model, the end user is transmitting the data to the PG on the edge of the backbone network. Once the gateway has the data, it buffers the data and becomes responsible for transmitting the data to the next hop. Thus, if a drop occurs on this segment, the retransmission will be noticed by the end host more quickly than in an end-to-end model. Also, when the drop occurs, the retransmission only needs to travel the length of the segment instead of the whole end-to-end path. The Phoebus model separates the segments of the path where drops are most likely to occur, the end users' networks, from the lower-loss long distance portion which will work to minimize redundant transmissions.

## 4.2 Authentication and Authorization

There are costs in bandwidth, memory and CPU associated with buffering a user's data on their behalf. Therefore, to ensure that a single user does not attempt to use all of the resources available on the PG, some form of authentication is required. To this end, we have implemented three forms of authentication.

The most basic form of authentication is no authentication at all. This authentication is similar to anonymous FTP where a user just specifies an email address when at connect time. This would require no work on the part of the administrator, but wouldn't offer any security.

A more secure form of authentication is password based. In this scenario, the user would specify a username, and the PG would return a randomly-generated one-time token. The user would then hash the token and his password using SHA-1 [2], HMAC [10] or similar and send the result to the server. The server would then authenticate the user. This approach is more secure than the anonymous method, but requires the system maintain a list of password hashes for all of users, and requires the users to remember what password is associated with each PG.

To combat the problems associated with password based authentication, the other option available in the current Phoebus implementation is use the authentication providers in the Globus Toolkit [8], which are based on GSI [1]. In this scenario, a user would obtain a credential from his local provider authenticating him. He could then use this credential to authenticate with the PG. If the PG trusted the user's provider, then it could be reasonably sure that the user was who he or she claimed to be. Thus, any user who uses Globus would be able to authenticate with the PG. The administrator would then only need keep track of who is allowed to use the server, not specific passwords.

## 4.3 Transparency

The introduction of a new protocol, with few exceptions, requires that applications be rewritten or otherwise modified to ensure that they can make use of the new features offered by the protocol. This can cause severe problems for adoption depending on the effort required to ensure that legacy applications can work with the new protocol. The Phoebus project has produced two methods to allow existing applications to make use of the protocol that require little to no changes to existing code.

### 4.3.1 Wrapper Library

The first method is a wrapper library. The wrapper library overrides the standard operating system socket calls so that any application linked against it can transparently use Phoebus infrastructure. Specific information such as the set of PGs that a given application or connection should use can be set by either the user or the administrator through the use of environmental variables.

This library can be used in two ways. The application can be explicitly linked against the library at compile time, or by setting the LD_PRELOAD environmental variable before executing the application. The only requirement from the users perspective is that he must install the software package and then set two environmental variables: one to load the library and another to

specify the first PG along the path. Besides the ease of configuration, the wrapper library allows the users and administrators to target which applications will use the Phoebus infrastructure. Thus, a user or administrator could setup GridFTP [9] to transparently make use of the library while leaving mail clients or system updates to simply connect directly to the destination. The wrapper library also allows for a more fine-grained select of which connections should be Phoebus-enabled by allowing specific destination addresses, address ranges or ports to be redirect over the Phoebus infrastructure. This allows users to use Phoebus for long distance *GridFTP transfers while using a direct connection for short distance* GridFTP connections, Email or Web traffic.

#### 4.3.2 Transparent Redirection

The second method for transparently enabling applications to use the Phoebus infrastructure makes use of the firewall infrastructure available in Linux. In this scenario, the administrator would setup a machine on his network running the Phoebus code. The end users would then route their traffic through this box. The administrator could then setup some firewall rules that would transparently redirect TCP connections destined for specific hosts or services through the forwarding software. The end user would no longer need to perform any local changes except to route their traffic through the forwarding box. Even this step could be mitigated by having the administrator reconfigure a switch to redirect the traffic transparently.

The downside to the second method is that the user may be unaware of or unable to get around his use of the Phoebus infrastructure. If the end application does not know that it is indirectly communicating with the end host, it will appear that it is talking to the edge PG. This may prove problematic for applications when they try to authenticate the service they are connecting to or vice-versa as both sides will see themselves talking to a node in between and believe that a man-in-the-middle attack is occurring. Another issue would be if the end user knows that a direct connection is better suited to his needs. In these two cases, the only options available are to use the Phoebus library to communicate directly with the first PG so as to avoid the confusion or to ask the administrator to turn off the redirection. The site administrator could minimize this inconvenience through the use of targeted firewall rules to ensure that only specific connections are redirected. However, the user is at the mercy of the firewall rules to ensure that his connections do not get redirected.

However, this form of transparent redirection has a major benefit when it comes to configuring large numbers of machines to use the software: it requires very little per machine configuration. If an administrator has a cluster of machines and wishes to switch to some other high-speed protocol, he must modify the software running on all the machines. This may not even be possible if the protocol has not been ported to the operating system that the software is running. However, with transparent redirection, the administrator can simply modify the outbound switch to redirect the new computers connections to his local PG, a modification that may need be performed only once for an entire cluster of machines, and all he would need to do is configure the local PG to support the new or changed protocol. This allows for the easy addition of new computers into the network independent of operating system.

## 5 Experiments

We tested the effectiveness of Phoebus using the prototype Phoebus infrastructure available on Internet2. This provided us with the opportunity to observe how the Phoebus concepts played out in the real world as well as to see how well it would work for the users of Internet2's Phoebus service.

### 5.1 Test Setup

Internet2's Phoebus infrastructure consists of gateway software running on machines located at the various Internet2 Points-of-Presence (POP). There are currently machines located in Los Angeles, Sunnyvale, Seattle, Washington D.C., Atlanta, New York City, and Chicago. For the purposes of this test, we used the machines located in Los Angeles, Chicago and New York.

These machines are Dual Processor Pentium 4 Xeons with 4G of RAM and a single 10G S2io NIC connecting it to the outside world. The backbone network that they are all connected to is a 10G network and is the same network used by regular Internet2 traffic.

The PGs that we used were a machine from SDSC whose link to Internet2 passed through the Los Angeles POP and a machine from Columbia whose link to Internet2 passed through the POP in either New York City or Chicago, depending on its destination. If the machine in Columbia were connecting to a cross-country host, its path would take it through Chicago. If it were going to somewhere on the east coast, the path would take it through New York. Each of these machines had a Gigabit connection to the outside world.

For this experiment, we performed 74 tests. Each test consisted of running *iperf* between each node in the test for a duration of 120 seconds. This allowed us to see the performance of each link to give us an idea of how they affected the overall transfer rates. The Phoebus portion of each test consisted of two separate runs. One used

**Table 1. Test Runs Between SDSC and Columbia**

| Source | Destination | Type | Throughput(Mb/s) |
|---|---|---|---|
| SDSC | Columbia | Direct | 243.69 +/- 37.46 |
| SDSC | Columbia | Phoebus | 382.91 +/- 99.54 |
| SDSC via NYC | Columbia | Phoebus | 715.67 +/- 135.33 |
| SDSC | Los Angeles | Direct | 751.47 +/- 150.74 |
| SDSC | Chicago | Direct | 569.91 +/- 207.8 |
| SDSC | NYC | Direct | 536.55 +/- 198.85 |
| Los Angeles | Chicago | Direct | 3832.53 +/- 827.88 |
| Los Angeles | NYC | Direct | 5053.2 +/- 22.67 |
| Los Angeles | Columbia | Direct | 210.68 +/- 63.61 |
| Chicago | Columbia | Direct | 419.51 +/- 39.16 |
| NYC | Columbia | Direct | 873.47 +/- 21.16 |

Chicago as the edge POP and the other used New York City as the edge POP. The run using Chicago as the POP was necessary to be able to make valid comparisons between a direct connection and Phoebus. Including New York City in the comparison allows us to see how well the system can do using the best route available, an option not actually available using direct connections.

To force the *iperf* client to make use of the Phoebus infrastructure, we used the preloaded library transparency method and set an environmental variable to tell it which edge PG to use. No modifications were required on the server end. In all of these tests, the data flowed from the west coast to the east coast. Thus, SDSC was always the client and Columbia was always the server.

The *iperf* client and server both had 32M TCP windows in both the Phoebus and direct cases. The gateways each had 8M TCP window sizes, both incoming and outgoing. TCP was chosen as the transport protocol for inter-PG connections in order to present a fair comparison between the direct TCP tests and those using Phoebus.

### 5.2 Results

The results show that a large throughput increase is possible using Phoebus. Table 1 gives the observed bandwidth for tests performed along the entire end-to-end path as well as each segment along the path. Performing a direct connection from SDSC to Columbia, we were able to obtain 243 Mb/s using a large window size. However, by simply setting a single environmental variable causing *iperf* to switch over to the Phoebus infrastructure and leaving all other variables the same, we were able to obtain a transfer rate of 382 Mb/s, an improvement of 57%.

The results support the notion that a Phoebus connection is hindered by the slowest link in the path. This hindrance means that the end-to-end connection runs a bit slower than the speed of the slowest link. In the connection between SDSC and Columbia using Chicago as the egress point, the slowest link is the link between the Chicago PSN and Columbia which runs at around 419 Mb/s. In this case, Phoebus runs a little below that at 383 Mb/s. The path using New York City as the egress point provides a completely different view. In that path, we found the slowest link to be the connection between SDSC and the Los Angeles PSN. The overall Phoebus connection using this path also runs a bit below the slowest link at 715 Mb/s.

To determine the effects of various transfer sizes on overall bandwidth, we ran a series of iperf tests with transfer sizes between 32M and 4G bytes.. Figure 2 shows the performance difference between the direct transfers and those utilizing the Phoebus infrastructure. These tests were also run between SDSC and Columbia using the default egress point at the Chicago POP and exhibits similar performance along that path as shown in Table 1. For small transfers, using Phoebus provides an increase upwards of 300% over direct tranfers. As the amount of data transfered increases, the initial TCP connection startup is amortized over the length of the transfer and we see a Phoebus performance improvement between 50%-60% up to 4G bytes.

Using the Phoebus infrastructure, the end user would have been able to see around a 50%-60% increase in throughput assuming he used the default path that was available to him during a sustained transfer. Transfering smaller amounts provides an even larger benefit. Achieving these results would have required no special knowledge of the network as the PSNs would have found his edge POP for him. Had he used a system like perfSONAR [16] to obtain information about the paths available, he could have found the faster path through New York City, specified it directly, and obtained a 293% inmprovement in throughput overall.

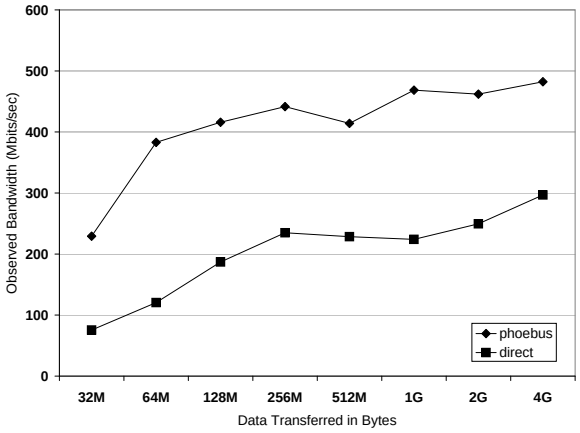



**Figure 2. Phoebus vs. Direct Transfers**

## 6 Conclusion

We propose a different architecture to better handle the needs of applications using long distance, high-throughput links. This architecture places servers at the edges of backbone networks whose job is to buffer users' data and take responsibility for transmitting it to the end host. This is a major benefit to both end users, network administrators and backbone networks. End users can achieve higher throughput with less work on their part as little tuning is need to ensure good speed over the short distance between the end user's machine and the edge of the high-speed backbone network. Network administrators can easily help users achieve high throughput by using transparent redirecting of their TCP connections to make use of the Phoebus infrastructure. The network backbones also have see a major benefit in that redundant packets are less likely to be transmitted over the the backbone, freeing up the resources for more useful traffic. We implemented these ideas in a service available on Internet2. In testing, we found that we were able to achieve a significant performance improvement by simply switching over to the Phoebus infrastructure and keeping all other variables the same.

## References

[1] Globus: Security. `http://www.globus.org/Security/`.

[2] Rfc 3174. `http://www.faqs.org/rfcs/rfc3174.html`.

[3] Akamai technologies. `http://www.akamai.com`.

[4] 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.

[5] A. Bakre and B. R. Badrinath. I-tcp: indirect tcp for mobile hosts. Proceedings - International Conference on Distributed Computing Systems, page 136, Vancouver, Can, 1995. IEEE, Piscataway, NJ, USA.

[6] Dragon. `http://dragon.maxgigapop.net`.

[7] S. Floyd. Connections with multiple congested gateways in packet-switched networks part1: One-way traffic. Computer Communication Review, V.21 N.5, October 1991.

[8] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

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

[10] M. B. H. Krawczyk and R. Canetti. Hmac: Keyed-hashing for message authentication. RFC 2104, February 1997.

[11] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. Network Working Group, Internet Engineering Task Force. Request For Comments: 1323, May 1992.

[12] T. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss, 1997.

[13] 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.

[14] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP throughput: A simple model and its empirical validation. *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, 1998.

[15] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis. An architecture for large-scale internet measurement. *IEEE Communications*, 1988.

[16] perfSONAR. `http://www.perfsonar.net/`.

[17] C. Qiao and M. Yoo. Optical burst switching (obs) - a new paradigm for an optical internet. `citeseer.ist.psu.edu/qiao99optical.html`, 1999.

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

[19] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, 1984.

[20] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of internet path selection. In *SIGCOMM*, pages 289–299, 1999.