Towards Bridging the Gap between Peak and Average Loads on Science Networks

Sam Nickolay[a], Eun-Sung Jung[b,*], Rajkumar Kettimuthu[c], Ian Foster[c,a]

[a]*Computer Science Department, University of Chicago, USA*
[b]*Hongik University, South Korea*
[c]*Math and Computer Science Division, Argonne National Laboratory, USA*

Abstract

Backbone networks are typically overprovisioned in order to sustain peak loads. Research and education networks (RENs) are designed to operate at 20-30% loads. For example, Internet2 upgrades the backbone interconnects when the weekly 95th-percentile load is reliably above 30% of link capacity. Our analyses of traffic on ESnet between DOE facilities also show that there is a huge gap between peak and average utilization. As science data volumes increase exponentially, it is unclear whether this over provisioning trend will continue in the future. Even if over provisioning is possible, it may not be the most cost-effective (and desirable) approach going forward. Under the current mode of free access to RENs, the traffic at peak load may range from flows that need to be transferred in near-real time, for example, for computation and instrument monitoring and steering, to flows that are less time-critical, for example, archival and storage replication operations. Thus, peak load does not necessarily indicate the capacity that is absolutely required. In this paper, we study how data transfers are impacted when the average network load is increased while the network capacity is kept at the current levels. We also classify data transfers into on-demand (ones that are time-critical) and best-effort (ones that are less time-critical) and study the impact on both classes for different proportions of both the number of on-demand transfers and amount of bandwidth allocated for on-demand transfers. We use real transfer logs from production GridFTP servers for our study. Our results indicate that the average slowdown experienced by the data transfers is under 1.5x even when the load is doubled with the network capacity fixed at the current levels. When the transfers are classified into on-demand and best-effort, on-demand transfers experience almost no slowdown and the slowdown experienced by best-effort transfers is under 2x, even when 50% of transfers were treated as on-demand.

Keywords: Network utilization, Data transfer scheduling, Network planning

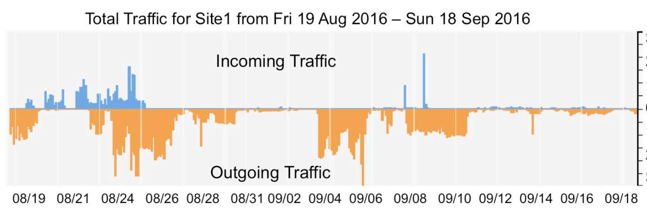
1. Introduction

Scientific applications in various domains such as high-energy physics, cosmology, genomics, etc., generate large data sets that need to be transported over the network for a variety of reasons. In order to support these applications, federal agencies in different countries fund organizations to build and operate high-speed research and education networks. These networks are typically overprovisioned in order to sustain peak loads so that all science users continue to have adequate network resources for their science workflows. Thus, they are underutilized most of the time. Research and education networks (RENs), such as Internet2, have a policy of operating their networks at light loads (25–30%) to allow the networks to absorb surges in traffic [1]. Other RENs such as ESnet and GEANT are also engineered to operate at similar average load levels. Given predictions of science traffics exponential growth, several recent studies project that network overprovisioning may not continue [2, 3, 4]. Recent reports on science network requirements note that different transfers have different needs [5, 6, 7]. Some transfers, such as the ones in use cases where the remote analysis result of one experiment is used to guide the next, are time-critical and have tight constraints. In contrast, some transfer, such as the ones in certain data replication, backup, archiving use cases, have more flexibility and may only need to be completed within a window several times longer than the transfer time under average load. Under the current mode of free access to RENs, the traffic at peak load may include a combination of different types of transfers including the ones that are less time-critical. We argue that measures to spread the load and keep the peak under control are important.

The main contributions of this study are:

- How data transfers are impacted when the average network load is increased while the network capacity is kept at the current levels?
- How does the impact change when the data transfers are classified into on-demand (ones that are time-critical) and best-effort (ones that are less time-critical) with on-demand transfers getting a relatively larger

*Corresponding author
Email addresses: `samnickolay@uchicago.edu` (Sam Nickolay), `ejung@hongik.ac.kr` (Eun-Sung Jung), `kettimut@anl.gov` (Rajkumar Kettimuthu), `foster@anl.gov` (Ian Foster)



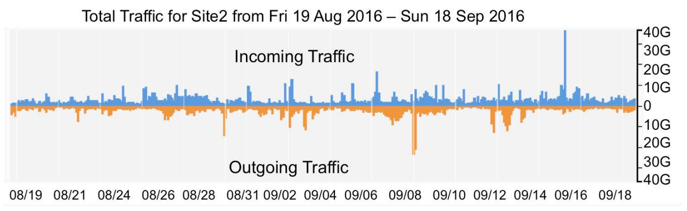


Figure 1: WAN traffic pattern of HPC facilities (source: my.es.net)

share of the bandwidth?

We use real transfer logs from production GridFTP servers for our study. Our results indicate that the average slowdown experienced by the data transfers is under 1.5x even when the load is doubled with the network capacity fixed at the current level for all the four GridFTP server logs we studied. For the logs where the peak load is 5x or more than the average load, average slowdown experienced by the data transfers is under 1.1x. Under the same scenario of doubled network load with same network capacity, when the transfers are classified into on-demand and best-effort, on-demand transfers experience almost no slowdown and the slowdown experienced by best-effort transfers is under 2x, even when 50% of transfers were treated as on-demand and on-demand transfers are given a reasonably higher share (70%) of the bandwidth. For the logs where the peak load is 5x or more than the average load, average slowdown experienced by best-effort data transfers is under 1.2x.

The rest of the paper is organized as follows. Section 2 introduces the two aspects that motivated our research. Section 3 presents the problem our research addresses and the metrics we use to evaluate our results. Section 4 defines the algorithm we developed for our research. Section 5 evaluates the results from our experiments. Section 6 examines the related work. Section 7 discusses the conclusions from our research.

2. Background and Motivation

We describe two aspects that inform our work.

2.1. Gap between Peak and Average Network Load

Figure 1 shows the wide-area traffic for a one-month period for two HPC facilities. This data was obtained from the website: my.es.net. It can be noticed that there is a huge gap between average and peak loads. We obtained logs from the anonymized usage statistics that Globus GridFTP servers send to a usage collector. These logs include transfer size, start time, and end time. We collected the logs of the 4 servers that transferred the most bytes in a one month period. For each of those servers, we then picked the log for the day in that month in which the most bytes were transferred by those servers. Figure 2 shows the aggregate throughput over that 24-hour period for each of the 4 trace logs. It can be noted that the peak throughput is approximately 4x, 8x, 5x, and 6x compared to the mean throughput for $trace_1$, $trace_2$, $trace_3$, and $trace_4$ respectively.

2.2. Some Transfers can Tolerate Delays

While certain transfer requests are time-sensitive, others can tolerate larger delays. Science communities replicate large quantities for performance [8, 9], fault tolerance [10], and/or preservation [11] and this replication is often cited as a common relatively time-insensitive data transfer use case [5, 6, 7]. Replicating TB datasets overnight is cited as one of the use cases, which can be done in a much less time in todays REN and HPC environments. Because subsequent processing involves manual steps, there is no advantage in completing the transfer earlier. Replicating 100TB within a month is another use case cited. For these use cases, transfer times can vary by at least an order of magnitude without compromising science goals.

3. Problem Statement and Goals

In this section, we formally define the problem to be addressed and the associated terms used throughout this paper. We also describe the goals from the perspective of our performance metrics.

3.1. Problem Statement

Our target networks are wide-area networks composed of many components such as switches, data transfer nodes (DTNs), etc. The networks can be either circuit-switched networks where paths are reserved ahead of data transfers or packet-oriented networks where the packets from multiple data transfers can share links on the networks. In reality, most networks used by data transfer users such as scientists are packet-switched networks except special networks such as ESnet [12] and Internet2 [13] which support circuits. However, we assume that both kinds of networks offer ways (e.g. tc in linux) to do traffic engineering on each flow.

In general, network traffic or data transfers can be categorized into multiple classes ranging from near-real time traffic to best-effort traffic depending on the importance of response time. In this paper, only two classes, on-demand (OD) data transfer and best-effort (BE) data transfer, are of our interest. OD data transfers have higher priority than BE data transfers since we assume that OD data transfers have stricter timing constraints than BE data transfers. However, our data transfer model does not have explicit timing constraints other than data transfer type. Thus, one transfer can be described by a five-tuple (*source*, *destination*, *transfer type*, *data size*, *requested rate*) where *source* is a data sender, *destination* is a data receiver, *transfer type* is either OD or BE, *data size* is size of data

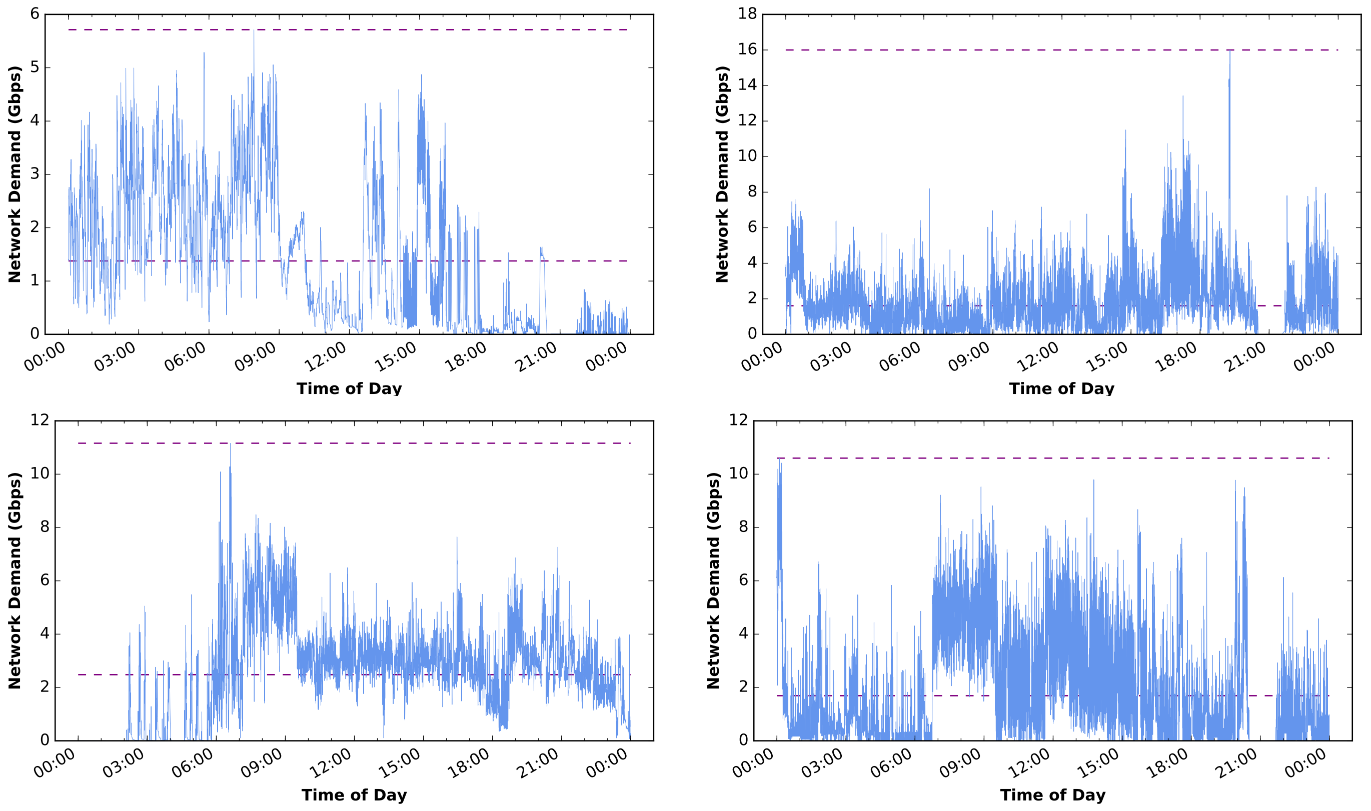


Figure 2: Network Demand for $trace_1$, $trace_2$, $trace_3$, $trace_4$.

to send, and *requested rate* is the estimated throughput of the data transfer when there is no network contention.

Another important assumption in this paper is that a single centralized controller is responsible for admission control and scheduling of all the data transfer requests. The single centralized controller thus has a single data transfer request queue and sequentially selects the next data transfer to schedule/perform. In order to properly schedule/perform the next data transfer, the controller will check the available bandwidth of the path for the data transfer. The available bandwidths of links in the networks that the controller is responsible for are assumed to be monitored periodically using some mechanisms such as SNMP. Without any congestion or failures, the path taken for a certain data transfer will remain the same throughout the data transfer even in packet-switched networks.

In our scheduling framework, all transfers have 2 rates associated with them: current rate, and requested rate. The current rate is the transfer rate or bandwidth currently allocated to the transfer by a centralized controller. The requested rate is the estimated throughput provided in the data transfer request information. For our simulations, the requested rate is set to the transfer rate in the original trace log. The scheduling framework operates under the assumption that we can selectively limit the bandwidth allocated to an individual transfer at any given time. It also assumes that a transfer's requested rate represents the transfer throughput when there is no limiting external load and so the requested rate is the highest bandwidth that the transfer could achieve. Thus, a transfers current rate can never be higher than its requested rate.

3.2. Goals and Performance Metrics

The goals in our work are two-fold. First, we would like to reduce the gap between the peak link utilization and the average link utilization on the path so that efficient network utilization can prevent expensive network planning for upgrade, e.g. 100Gbps to 1Tbps. Second, we would like to ensure that OD data transfers can be initiated immediately without much delay while BE data transfers suffer only reasonable amount of delays.

For such reasons, we use two performance metrics: 1) slowdown and 2) gap of the peak rate and the average rate. *Slowdown*, commonly used in the context of parallel transfer scheduling [14], indicates the factor by which a data transfer is slowed relative to the actual transfer time it takes. *Slowdown* is generally defined to be $\frac{Waittime+Runtime}{Runtime}$ where *waittime* is the time between the transfer's arrival time and scheduled time. However, *Slowdown* may become so large for a tiny data transfer with a large wait time that it can distort the overall *slowdown* average. *Bounded slowdown* is introduced to bound such cases to some extent using *max* operation, and the formal definition for an individual data transfer is defined as Equation 1. *Bound* is a user-defined threshold used to limit the influence of very short transfers on slowdown.

$$BS = \frac{\text{Waittime} + max(\text{Simulation Runtime}, \text{Bound})}{max(\text{Log Runtime}, \text{Bound})} \tag{1}$$

Since we do simulations with real data trace logs, we make several tweaks in the original slowdown equation. *Runtime* in the numerator is replaced by *simulation runtime*, which is the transfer's runtime in the simulation. *Runtime* in the denominator is replaced by *log runtime*, which is the transfer's runtime in the original data transfer log. *Waittime* is the time difference between when the transfer started in the original log and in the simulation, For our simulations, *bound* is set to be equal to the scheduling interval used for scheduling the transfers, which is 1 second.

For the simulations, we actually use a total of four different bounded slowdown metrics.

- Avg_BS_{OD}: Average bounded slowdown of all OD data transfers, defined as Equation 2.
- Max_BS_{OD}: Maximum bounded slowdown among all OD data transfers, defined as Equation 3.
- Avg_BS_{BE}: Average bounded slowdown of all BE data transfers, defined likewise as Avg_BS_{OD}.
- Max_BS_{BE}: Maximum bounded slowdown among all BE data transfers, defined likewise as Max_BS_{OD}.

$$Avg\_BS_{OD} = \left(\sum_{\forall t} (Bounded\ Slowdown)_t \right) / T \tag{2}$$

$$Max\_BS_{OD} = \max_{\forall t} (Bounded\ Slowdown)_t \tag{3}$$

where $t \in$ OD transfers in a trace log,

T is the total number of OD transfers.

4. Scheduling Algorithm

After presenting our formal problem statement and goals, we now present the algorithm we have developed for scheduling OD and BE data transfers to achieve the goals within our simulation framework. Table 1 summarizes the terminology used in our algorithm.

Table 1: Summary of terminology

Item	Description
OD transfers	On-demand transfers
BE transfers	Best-effort transfers
transfer.rate	Bandwidth transfer is currently allocated
transfer.requestRate	Bandwidth transfer has requested as an estimated throughput; In simulation, this is set to the transfer throughput in the original trace log
$requestLoad_{OD}$	$\sum_{\forall \text{running OD transfers}} transfer.requestRate$
$requestLoad_{BE}$	$\sum_{\forall \text{running BE transfers}} transfer.requestRate$
$availBW_{OD}$	Bandwidth available for OD transfers
$availBW_{BE}$	Bandwidth available for BE transfers
netCap	Maximum bandwidth for the network
$netCap_{OD}$	Bandwidth allocated for OD transfers
$netCap_{BE}$	Bandwidth allocated for BE transfers

Listing 1: Scheduling Heuristic

```
# 1. Start OD transfers and BE transfers
for transfer in OD transfer queue:
  start transfer
for transfer in BE transfer queue
  start transfer

# 2. Compute BE transfers available bandwidth
if $requestLoad_{OD}$ < $netCap_{OD}$:
  $availBW_{BE}$ = netCap - $requestLoad_{OD}$
else:
  $availBW_{BE}$ = $netCap_{BE}$

# 3. Update BE transfer rates
for transfer in BE running transfers:
  fairshareBandwidth = transfer.requestRate /
      $requestLoad_{BE}$ × $availBW_{BE}$
  transfer.rate = min(fairshareBandwidth,
      transfer.requestedRate)

# 4. Compute OD transfers available bandwidth
if $requestLoad_{BE}$ < $netCap_{BE}$:
  $availBW_{OD}$ = netCap - $requestLoad_{BE}$
else:
  $availBW_{OD}$ = $netCap_{OD}$

# 5. Update OD transfer rates
for transfer in OD running transfers:
  fairshareBandwidth = transfer.requestRate /
      $requestLoad_{OD}$ × $availBW_{OD}$
   transfer.rate = min(fairshareBandwidth,
       transfer.requestedRate)
```

4.1. Overview of Algorithm

Our scheduling algorithm works with two distinct categories of data transfers, OD and BE. Our scheduling algorithm does not give any implicit priority or scheduling advantage to either type of transfer. Instead it is though explicitly setting the allocated OD bandwidth ($netCap_{OD}$) and BE bandwidth ($netCap_{BE}$) values that priority is given to one transfer type over another. For the sake of clarity, we define $netCap_{OD}$ and $netCap_{BE}$ in terms of a percentage of the total $netCap$, and since $netCap_{OD} + netCap_{BE} = netCap$, the two percentages always sum to 1. In our work, OD transfers are designated as being more time critical than BE transfers. To account for this in our simulations, we allocate OD transfers a $netCap_{OD}$ that is a higher percentage of the total $netCap$ than the percentage of OD transfers in the simulation. For instance, consider a simulation in which OD transfers make up 30% of all the transfers and BE transfers account for the remaining 70%. By allocating OD transfers a guaranteed bandwidth that is more than 30% of the total network bandwidth, such as setting $netCap_{OD} = netCap \times 40\%$ and $netCap_{BE} = netCap \times 60\%$, we indirectly give OD transfers a higher priority in our scheduling algorithm.

The detailed algorithm is elaborated in Listing 1. The scheduling cycle repeats every n seconds; `OD transfer queue` and `BE transfer queue` contain any new transfers that arrived in those n seconds. (In our implementation, $n = 1$.) At the start of each cycle, the scheduler starts any OD and BE transfers that are in `OD transfer queue` and `BE transfer queue` respectively. Next, the algorithm computes the available bandwidth for BE transfers ($availBW_{BE}$) by comparing $requestLoad_{OD}$ with $netCap_{OD}$. If the requested OD bandwidth is less than the the allocated OD bandwidth, then there is extra bandwidth that was allocated for OD transfers but is now available for BE transfers. Thus, $availBW_{BE}$ is equal to the difference between total bandwidth and the bandwidth requested by OD transfers ($netCap - requstedLoad_{OD}$). Otherwise, the available BE bandwidth is equal to the allocated BE bandwidth ($netCap_{BE}$).

Then, for every active BE transfer, the scheduler calculates the transfer's $fairshareBandwidth$:

$$FairshareBandwidth = \frac{transfer.requestRate \times availBW}{requestLoad} \quad (4)$$

where $requestLoad$ is the requested bandwidth for all active transfers of the same type as the current transfer, and $availBW$ is the bandwidth available to transfers of the same type as the current transfer. As noted above, $transfer.currentRate$ is capped by $transfer.requestRate$, and so a transfer's current rate is the minimum of its $fairshareBandwidth$ and requested rate.

After updating the transfer rates for all BE transfers, the process is repeated for OD transfers. First, the scheduler computes the available bandwidth for OD transfers ($availBW_{OD}$). If $requestLoad_{BE}$ is less than $netCap_{BE}$, then there is extra BE bandwidth available for OD transfers, and $availBW_{OD}$ equals the difference between total bandwidth and the bandwidth requested by BE transfers ($netCap - requstedLoad_{BE}$). Otherwise, $availBW_{OD}$ equals the allocated OD bandwidth ($netCap_{OD}$). Then, for every active OD transfer, the algorithm computes its $fairshareBandwidth$ as defined above, and sets the transfer's current rate to be the minimum of its *fairshareBandwidth* and requested rate.

Our algorithm is focused only on two end points and we assume that the path between two end points does not change over time and no abrupt data transfer sharing the links on the path takes place. The sophisticated algorithm on the general network topology will be left as future work and in this paper we aim to verify that capping the link utilization is feasible through well-designed data transfer scheduling.

4.2. Illustrative Example

To understand how the dynamic scheduling algorithm distributes the network bandwidth to OD and BE transfers, consider a simulation in which $netCap_{OD} = 30\% \times netCap$. In this case, OD transfers are guaranteed access to 30% of the total bandwidth regardless of the bandwidth requested by BE transfers. However, this doesnt mean that the total OD bandwidth will always be 30%. For example, if at one point during the simulation, $requestLoad_{OD} = 15\% \times netCap$ and $requestLoad_{BE} = 100\% \times netCap$. In this case, OD transfers will only use the bandwidth they requested ($15\% \times netCap$), and the remaining 85% of the bandwidth will be available for BE transfers. If at another point in the simulation the situation is reversed and $requestLoad_{OD} = 100\% \times netCap$ and $requestLoad_{BE} = 15\% \times netCap$. Then BE transfers will use the bandwidth they requested ($15\% \times netCap$), and OD transfers will use the remaining 85%. In short, each transfer type is guaranteed a certain percentage of the *netCap*. However, if at any point one transfer type requests less bandwidth that it is allocated, any remaining bandwidth becomes available for the other transfer type.

Next, to understand how the fairshare heuristic allocates bandwidth, consider an example where $availBW_{OD}$ is 5Mbps and there are 4 active OD transfers. We will denote the transfers as t_1, t_2, t_3, t_4, with requested transfer rates equal to 1Mbps, 2Mbps, 3Mbps, and 4Mbps respectively, and so $requestLoad_{OD}$ is 10Mbps. Calculating the $fairshareBandwidth$ for the OD transfers we get $t_1 = 0.5Mbps$ ($1 \times 5 \div 10$), $t_2 = 1.0Mbps$ ($2 \times 5 \div 10$), $t_3 = 1.5Mbps$ ($3 \times 5 \div 10$), $t_4 = 2.0Mbps$ ($4 \times 5 \div 10$). In this case, the transfers' current rates are limited by their $fairshareBandwidth$ since $availBW_{OD}$ was less than $requestLoad_{OD}$. If $availBW_{OD}$ was higher, for example 20Mbps, then the transfers' current rates would instead be limited by the transfer's requested rates and would be 1Mbps, 2Mbps, 3Mbps, and 4Mbps respectively.

5. Experimental Evaluation

In this section, we present comparative results of our algorithm with regard to the original transfer trace logs through extensive simulations and discuss the results at the end.

5.1. Simulation Environment

We constructed a simulation framework that uses the transfers in a particular trace log to run simulations based on a number of user-defined variables. The simulator reads the logs in an online fashion, meaning that future transfer requests are not known a priori. Our framework is a discrete event simulator specifically designed to simulate transfers between two endpoints based on an inputted trace log. It operates under the principle that it schedules and processes transfers based on a user-defined scheduling interval. For our experiments, the scheduling interval was set to 1 second which we found provided good performance and accuracy.

At the start of each scheduling interval, the simulator updates any transfers from the previous interval and removes any transfers that completed transferring. It then runs the scheduling algorithm described above before moving on to the next interval. Each of our experiments consists of a 24 hour simulation from 12:00am to 12:00am the following day. Since data traffic patterns and quantities tend to vary over the course of a day, we wanted to ensure we captured a full days worth of transfers.

Ultimately our goal is to minimize the gap between the peak and average throughput for all OD and BE transfers. The mean throughput should remain the same for all of our simulations, because a different mean would imply that a different amount of total data was transferred. Therefore, our proposed scheduling algorithm should be able to minimize the peak throughput, which is limited by the network capacity, without significantly affecting the slowdown of transfers. Although our objective is to keep slowdown values down to a reasonable level for both OD and BE transfers, since OD transfers are more response critical than BE transfers, the threshold for impractical slowdown values is much lower for OD transfers than it is for BE transfers.

5.2. Data Transfer Workload

We evaluated our algorithm with 4 real traces from Globus GridFTP servers, each of which contains information for all of the transfers that went from a particular pair of source and destination endpoints on one day. We chose four trace logs that have varying peak and mean throughput. The peak throughput for the trace logs varies from 5.7Gbps to 16.0Gbps, and the mean throughput varies from 1.4Gbps to 2.5Gbps. Though more importantly, all four trace logs have a mean throughput between 10% and 25% of the peak. We assign each transfer an original transfer throughput, which we refer to as the transfer's *requestedRate*, based on limited information in the trace logs. We assume that the assigned original throughput is the achievable throughput between two end points when there is no network contention.

5.3. Simulation Variables

We ran our simulation framework with various parameter configurations to assess how our algorithm performs under different workloads. The variables we adjusted are % OD transfers, % OD bandwidth, and transfer load ratio. % OD transfers is the percentage of transfers that are labeled as OD transfers and all other transfers (100% - % OD transfers) are labeled as BE transfers. When the simulator reads in a trace log, it randomly splits the transfers into OD and BE categories based on the % OD transfers. % OD Bandwidth is the percentage of the total path capacity that is dedicated to OD transfers. Transfer load ratio is the ratio of the transfer load size used in the simulation compared to the transfer load size used in the original trace log. This parameter is used to test the simulation under different intensities of transfer loads. For example, a simulation with a transfer load ratio of 2, uses the same transfers as defined in original trace log, but every transfer has double the transfer size and duration resulting in a total transfer load that is twice as big as the original transfer load.

We used 4 different % OD transfers, 4 different % OD bandwidth, and 4 different transfer load ratios, which is a total of 64 different configurations for each trace log. We simulated with % OD transfers $\in \{10\%, 30\%, 50\%, 70\%\}$. For the % OD bandwidth, we varied the values based on the % OD transfers in order to make the results more pertinent. In order to give OD transfers a higher priority than BE transfers as we intended, the percent % OD bandwidth had to be greater than or equal to the % OD transfers. Therefore, given the current % OD transfers, we experimented with % OD bandwidth values that are equal to the % OD transfers, 10% and 20% greater than the % OD transfers, and 100% of the total path bandwidth. For example, simulations where the % OD transfers is 50%, we evaluated the following percentages for % OD bandwidth $\in \{50\%, 60\%, 70\%, 100\%\}$. Finally, we used the following transfer load ratios $\in \{1.0x, 1.5x, 2.0x, 2.5x\}$. Although the set of transfer load ratios is the same for all trace logs, the resulting transfer loads are different, since the default (1.0x) transfer loads vary for each trace log. For instance, a transfer load resulting from a 2x transfer load ratio for one trace log is different than the transfer load resulting from a 2x transfer load ratio for a different trace file

After simulating each configuration, we calculated the 4 performance metrics (i.e. Avg_BS_{OD}, Max_BS_{OD}, Avg_BS_{BE}, and Max_BS_{BE}), defined in Section 3.2.

5.4. Experimental Results

We illustrate our results with a series of figures, which contains the average and max bounded slowdown results for $trace_1$, $trace_2$, $trace_3$, and $trace_4$. Due to space constraints we only include max slowdown graphs for $trace_1$, $trace_2$; however, the trends in the max slowdown graphs for $trace_1$, $trace_2$ were comparable.

In Figures 3 through 6, the top and bottom row show the average OD bounded slowdown, and average BE bounded slowdown respectively. Each row is divided into 4 subplots for simulation runs with different % OD transfers ranging from 10% to 70%. Then, each individual subplot contains four separate lines, one for each different % OD bandwidth value. Finally each % OD bandwidth line within a subplot has 4 datapoints, one for each transfer load ratio. As a result, the x-axes, indicating the transfer load ratios, are the same for all subplots in every trace logs, and the y-axes, which illustrate bounded slowdown using a logarithmic scale, are shared between the columns within a row.

To compare the performance of our scheduling heuristic, we also ran simulations without our scheduling heuristic as baselines. These control simulations have 100% BE transfers and 0% OD transfers, and BE and OD bandwidths are 100% and 0% respectively. For each control experiment, we computed the performance metrics defined in Section 3.2, and represent these metric values in the figures with a yellow shaded region on each of the plots. Almost all of the OD slowdown plot lines fall inside the shaded region, while the BE slowdown plot lines are above the shaded region, meaning that our algorithm performed better for OD slowdown, but worse for BE slowdown compared to the control experiments. Thus, our scheduling

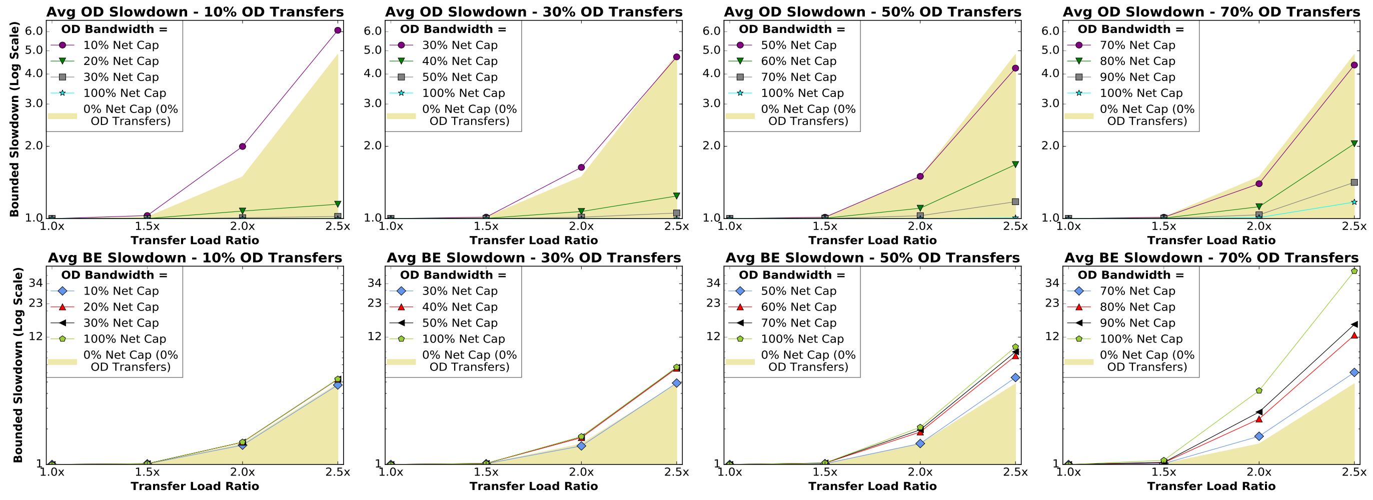


Figure 3: Average Bounded Slowdown for $trace_1$.

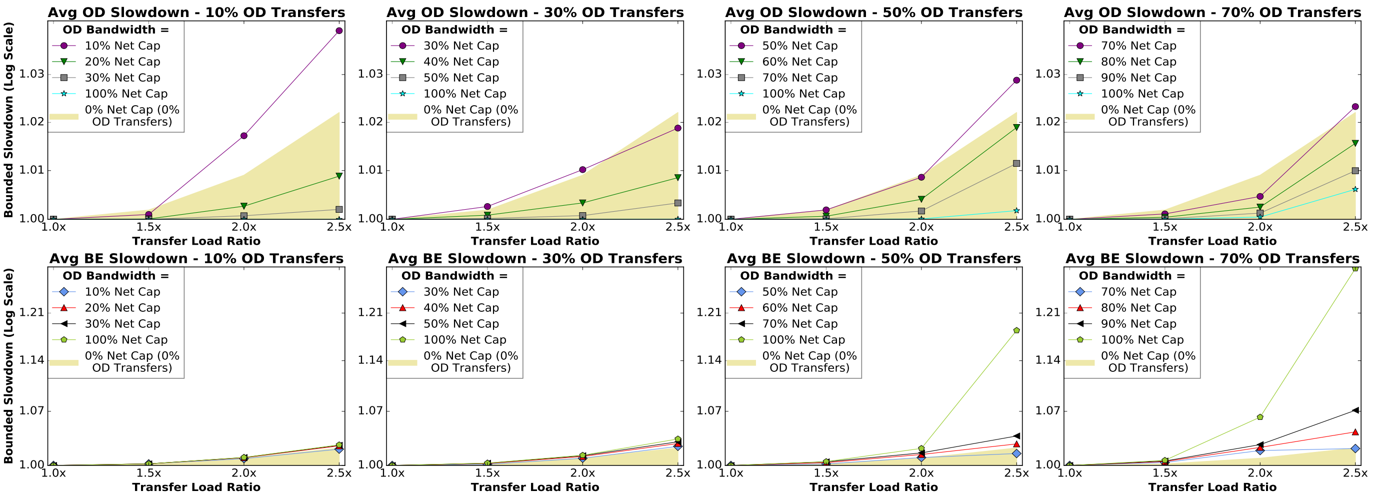


Figure 4: Average Bounded Slowdown for $trace_2$.

algorithm offers improved performance for OD transfer, but it comes at the expense of BE transfer performance.

OD bandwidth is negatively correlated with OD slowdown and positively correlated with BE slowdown, meaning that the higher the OD bandwidth, the lower the OD slowdown, but the higher the BE slowdown. In our experiments, as long as the bandwidth allocated to BE transfers is comparable to the proportion of BE jobs, such as when there are 50% BE transfers and they are allocated 30% of the total bandwidth, the average BE slowdown is less than 2.0x even when the transfer load is doubled. Conversely, in simulations where OD transfers are allocated 100% of the bandwidth, and thus BE transfers are only given bandwidth when it became available, OD transfers have zero increase in slowdown, but BE transfers have significantly higher slowdown. Thus, it is necessary to consider the requirements and relative importance of OD and BE transfers before making a tradeoff between OD and BE performance. If the goal is to minimize OD slowdown, it makes sense to allocate 100% of the bandwidth to OD transfers; however, if both OD and BE slowdowns are important, it is necessary to strike a more equal balance between OD and BE transfer bandwidths. Based on our experiments, a % OD bandwidth equal to the % of OD transfers + 10% or 20% results in a reasonable balance between OD and BE slowdowns.

As shown in Figures 7 and 8, in our control experiments when 100% of transfers are BE, the max OD slowdown is under 12.0 even under a 2x load. Note that we present the results for only $trace_1$ and $trace_2$ as the results of $trace_3$ and $trace_4$ are similar to that of $trace_1$. When the transfers are categorized into OD and BE, the max slowdown for BE becomes significantly higher when BE transfers are only given the leftover bandwidth (OD Bandwidth = 100% of available bandwidth) and there are 50% or more OD transfers. However, when OD transfers are only allowed to use 10% or 20% more than their proportion, the max slowdown for OD transfers is under 5.0, and the max slowdown for BE transfers is under 18.0.

Although there is a correlation between transfer load ratios and bounded slowdown values (higher transfer load ratios, results in higher bounded slowdown values), our results indicate that it is possible to increase the transfer load without significantly affecting transfer slowdown. For example, in the control experiments, when the transfer load is doubled and the network capacity fixed at the current level, the averaged slowdown for the transfers is under 1.5x, and under our algorithm with 50% OD transfers and

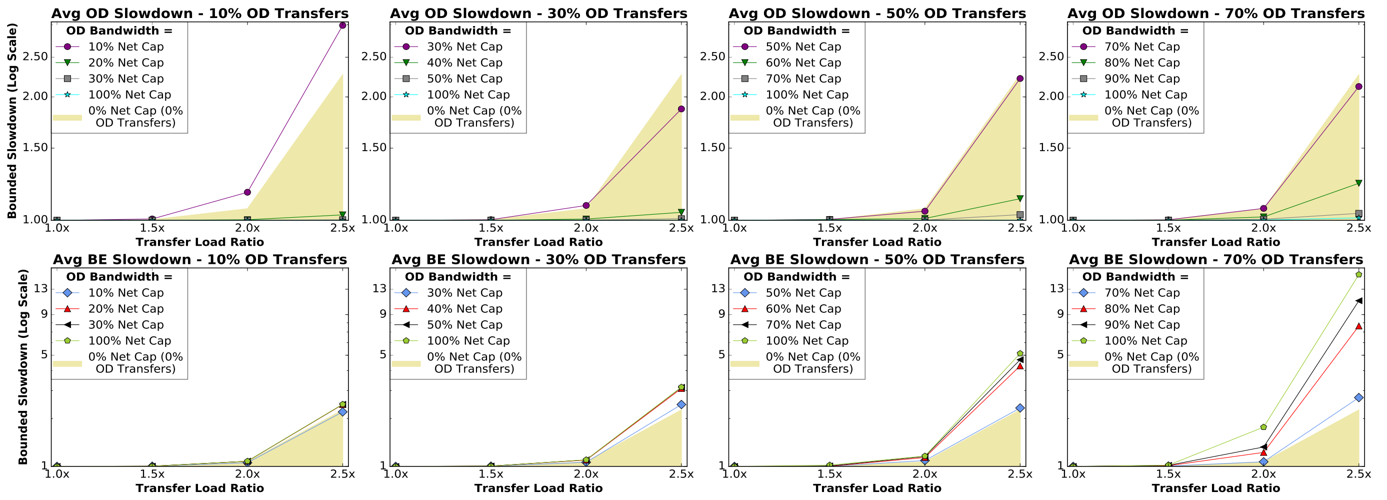


Figure 5: Average Bounded Slowdown for $trace_3$.

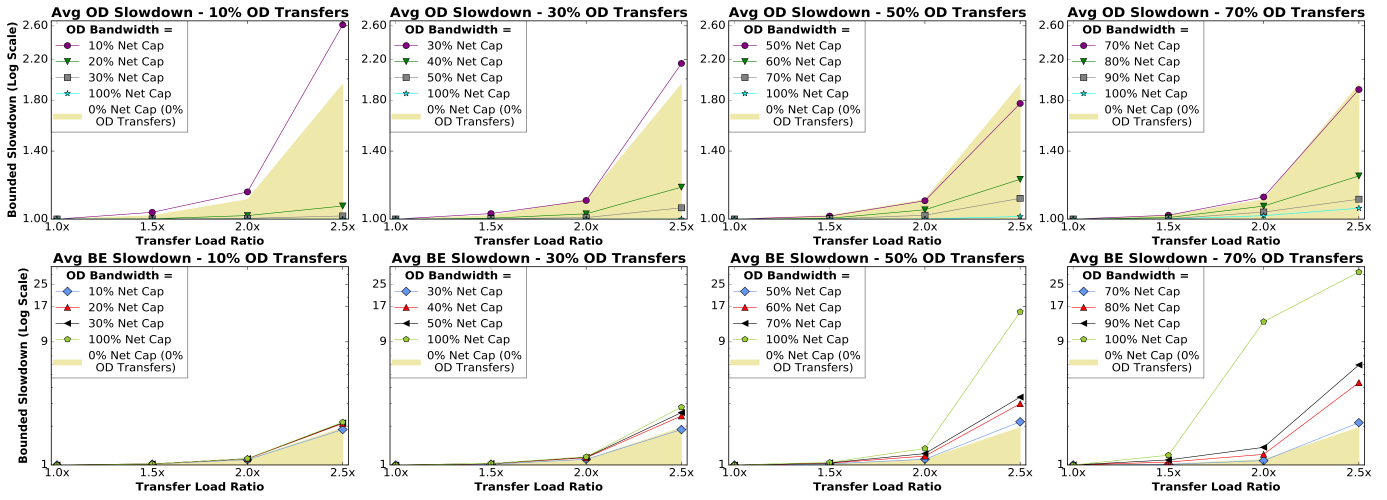


Figure 6: Average Bounded Slowdown for $trace_4$.

50% of BE transfers, OD transfers experience negligible average slowdown and BE transfer average slowdown is under 2x.

Furthermore, simulations from the different trace logs resulted in a wide range of slowdown values. In particular, the average slowdowns for $trace_1$ are significantly higher than for other trace logs, which is likely the result of $trace_1$ having the highest mean to peak ratio (0.24). Thus, proportionally it has the least amount of additional bandwidth available for extra network traffic. This hypothesis is supported by $trace_2$, which has significantly lower slowdowns than the other trace logs and also has the lowest mean to peak ratio (0.10) meaning it relatively has the most bandwidth available for additional network traffic.

5.5. *Coefficient of Variation*

As shown in Figure 2, there is a large gap between the mean and peak demand in the trace logs used in our experiments, which is common for science networks. Of the four network demand graphs in Figure 2, the bottom two plots ($trace_3$ and $trace_4$) are relatively similar in terms of mean and peak demand; however, comparing these two trace logs raises an interesting inconsistency. Although $trace_3$ has a higher mean demand, peak demand, and mean to peak ratio than $trace_4$ (Mean: 2.5 Gbps vs. 1.7 Gbps. Peak: 11.2 Gbps vs. 10.6 Gbps. Mean to peak ratio: 0.22 vs. 0.16), in our experiments $trace_3$ had lower OD and BE slowdown values. Comparing the network demand graphs for the two trace logs in Figure 2 show noticeable differences; while the graph for $trace_3$ is relatively centered around the mean, the graph for $trace_4$ has a much thicker and more jagged line indicating rapid and frequent variations in network demand.

To quantify the difference in network demand variation between the trace logs, we computed concurrency and throughput coefficient of variation metrics. To do so, we divided the 24-hour simulation period into 1 minute intervals, and computed the number of transfers that occurred during each of the intervals (concurrency intervals), and the total transfer throughput during each of the intervals (throughput intervals). Then, to calculate the coefficient of variation metrics, we used the following equations:

$$\text{Coefficient of Variation}_{concurrency} = \frac{Std.Dev.(\text{Concurrency Intervals})}{mean(\text{Concurrency Intervals})} \tag{5}$$

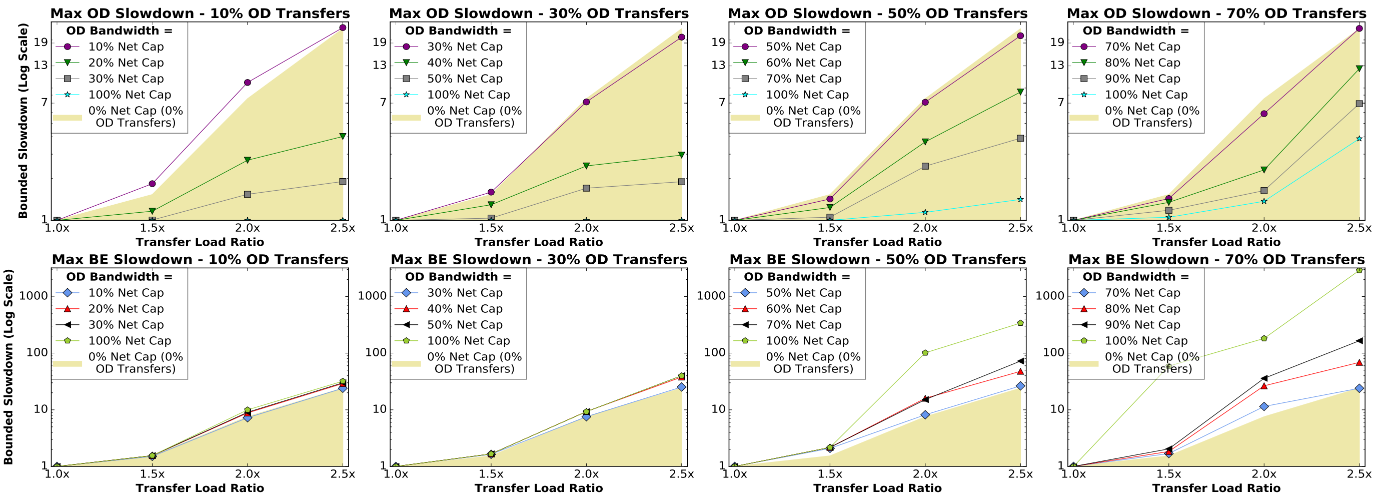


Figure 7: Max Bounded Slowdown for $trace_1$.

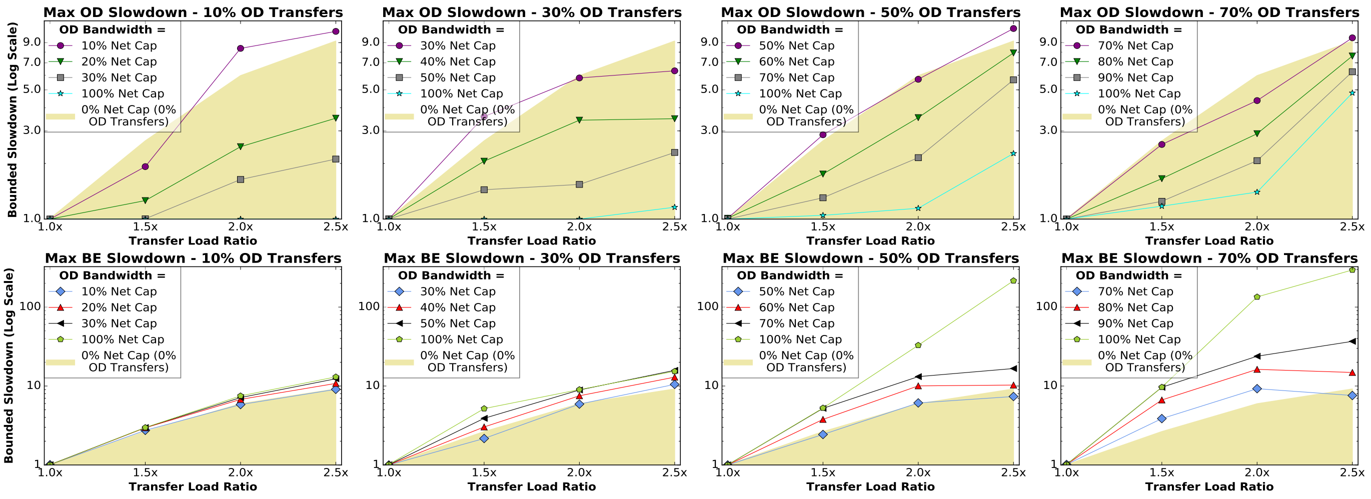


Figure 8: Max Bounded Slowdown for $trace_2$.

$$\text{Coefficient of Variation}_{Throughput} = \frac{Std.Dev.(\text{Throughput Intervals})}{mean(\text{Throughput Intervals})} \quad (6)$$

Concurrency and throughput coefficient of variations for $trace_3$ were 0.67 and 0.69 respectively, and concurrency and throughput coefficient of variations for $trace_4$ were 0.87 and 1.07 respectively. We suspect that as the network demand increases, the number of concurrent or overlapping transfers also increases, which raises slowdown values since there are more transfers competing for bandwidth. However, the inverse is not necessarily true, since if the network demand drops below the available network bandwidth, all transfers will have a slowdown of 1, which is the lowest possible slowdown value, and so even if the network demand continues to decrease, slowdown values will remain at 1. Since $trace_4$ has a higher variation in concurrency and throughput, it has more moments of low and high network demand in comparison to $trace_3$, which has a network demand that remains more consistently around the mean. The higher variation and frequent moments of high network demand result in higher slowdown values and therefore worse performance in $trace_4$ than in $trace_3$, even though the mean to peak ratio is higher in $trace_3$.

6. Related Work

Several studies have developed methods of using traffic engineering to improve network performance and efficiency, including locally optimal intradomain traffic engineering [15], globally optimal software driven WAN [16], and differentiating files transfers based on priority level [17]. In particular, studies have explored ways to reduce the adverse efforts of high-rate, bursty traffic associated with large scientific data transfers. In [18], researchers used traffic-engineered paths and isolated queues to accomplish this. Our research differentiates itself from previous work by addressing the gap between peak and mean network loads.

Other research has shown that dynamic link width management can be an effective way to reduce energy costs without negatively affecting sensitive applications involving on-demand network transfers [19]. However, this approach involves overprovisioning and building a network to meet the peak demand, and then turning off network links when not in use. This differs from our research which minimizes the need to overprovision the network by using network traffic scheduling to reduce the gap between the peak and the mean throughput.

Our research has applications outside of managing traffic on science networks, especially situations that involve a large disparity between the peak and average demand for some resource. For example, electric grids are known to exhibit high variability in demand and overall low efficiency. Several studies have examined how different policies and pricing models can help reduce the peak-to-mean ratio in smart electrical grids [20]. Public cloud computing is another active area of research related to improved resource allocation without overprovisioning, and studies have suggested a number of different price models that can be used to effectively and fairly reduce the peak demand [21], [22].

7. Conclusions

We presented a study to motivate measures to reduce the huge gap between peak and average loads in research and education networks. Using real world logs from production GridFTP servers, we simulated high data transfer loads by keeping the network capacity at current levels and studied the performance of the data transfers. We showed that current network capacity can handle up to 2x the current load with minimal impact to the data transfers, when the peak load is 5x or more than the average load. We also showed that when the transfers are categorized into on-demand and best-effort, with preferential treatment for on-demand transfers, the impact on the transfers that really need on-demand service can be made negligible (while at the same time keeping the impact on best-effort transfers minimal).

8. Acknowledgments

This work was supported in part by the U.S. Department of Energy under contract number DEAC02-06CH11357.

9. References

[1] "Internet2 IP backbone capacity augment practice," http://www.internet2.edu/policies/ip-backbone-capacity-augment-practice/.

[2] G. Bell and M. Ernst, "HEP Community Summer Study 2013 Computing Frontier: Networking," in *Snowmass'2013*.

[3] K. Bergman, V. Chan, D. Kilper, I. Monga, G. Porter, and K. Rauschenbach, "Scaling terabit networks: Breaking through capacity barriers and lowering cost with new architectures and technologies," *NSF Workshop*, 2013.

[4] "ESnet Strategic Plan," http://www.es.net/assets/Uploads/ESnet-Strategic-Plan-March-2-2013.pdf.

[5] "Biological and Environmental Research Network Requirements, Nov. 2012," http://www.es.net/assets/pubs_presos/BER-Net-Req-Review-2012-Final-Report.pdf.

[6] "Basic Energy Sciences Network Requirements Workshop, September 2010 - Final Report," http://www.es.net/assets/Uploads/BES-Net-Req-Workshop-2010-Final-Report.pdf.

[7] "High Energy & Nuclear Physics Net. Req. Review, Aug. 2013," http://www.es.net/assets/Papers-and-Publications/HEP-NP-Net-Req-2013-Final-Report.pdf.

[8] "LIGO Data Replicator," http://www.lsc-group.phys.uwm.edu/LDR/.

[9] D. Williams *et al.*, "Earth System Grid: Enabling access to multi-model climate simulation data," *Bulletin of American Meteorological Society*, vol. 90, no. 2, 2009.

[10] "Belle-II Experiment Network Requirements, Oct. 2012," http://www.osti.gov/scitech/servlets/purl/1171367.

[11] S. Habib, V. Morozov, H. Finkel, A. Pope, K. Heitmann, K. Kumaran, T. Peterka, J. Insley, D. Daniel, P. Fasel, N. Frontiere, and Z. Lukić, "The universe at extreme scale: Multi-petaflop sky simulation on the bg/q," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 4:1–4:11. [Online]. Available: http://dl.acm.org/citation.cfm?id=2388996.2389002

[12] I. Monga, C. Guok, W. E. Johnston, and B. Tierney, "Hybrid networks: lessons learned and future challenges based on esnet4 experience," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 114–121, May 2011.

[13] Internet2, "Layer 2 services," http://www.internet2.edu/products-services/advanced-networking/layer-2-services/, accessed: 2016-05-04.

[14] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong, "Theory and Practice in Parallel Job Scheduling," in *Proceedings of the Job Scheduling Strategies for Parallel Processing*, ser. IPPS '97. London, UK, UK: Springer-Verlag, 1997, pp. 1–34. [Online]. Available: http://dl.acm.org/citation.cfm?id=646378.689517

[15] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 118–124, Oct. 2002.

[16] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving High Utilization with Software-driven WAN," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 15–26. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486012

[17] R. Kettimuthu, G. Agrawal, P. Sadayappan, and I. Foster, "Differentiated Scheduling of Response-Critical and Best-Effort Wide-Area Data Transfers," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2016, pp. 1113–1122.

[18] Z. Yan, C. Tracy, M. Veeraraghavan, T. Jin, and Z. Liu, "A network management system for handling scientific data flows," *Journal of Network and Systems Management*, vol. 24, no. 1, pp. 1–33, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10922-014-9336-2

[19] K. Kant, "Power control of high speed network interconnects in data centers," in *INFOCOM Workshops 2009, IEEE*, April 2009, pp. 1–6.

[20] H. K. Nguyen, J. B. Song, and Z. Han, "Demand side management to reduce peak-to-average ratio using game theory in smart grid," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, March 2012, pp. 91–96.

[21] N. Nasiriani, C. Wang, G. Kesidis, B. Urgaonkar, L. Y. Chen, and R. Birke, "On fair attribution of costs under peak-based pricing to cloud tenants," in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2015 IEEE 23rd International Symposium on*, Oct 2015, pp. 51–60.

[22] M. Mattess, C. Vecchiola, and R. Buyya, "Managing peak loads by leasing cloud infrastructure services from a spot market," in *High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on*, Sept 2010, pp. 180–188.