**Project Summary**

The emergence of the Grid computing paradigm, especially in the scientific enterprise, raises new interest in optimizing the end-to-end performance of *data intensive distributed applications*. The *MONADD* (Middleware for Optimized Network-Aware Data Dissemination) project will develop a set of middleware components that aim to dramatically improve the performance of distributed data intensive applications. The two major MONADD approaches are to maximize the throughput of point-to-point TCP transfers, and to optimize the use of multi-server parallel downloads. A central feature in MONADD is its *network awareness*, meaning that the proposed middleware monitors the underlying network, including data sources and servers, in terms of various performance metrics, in order to make the best possible use of the available resources.

**Intellectual Merit.**
The MONADD project takes a significantly different approach in dealing with the inefficiency of the TCP transport protocol in bulk data transfers. Instead of changing or replacing TCP, a middleware component automatically determines the optimal socket buffer size that saturates the available bandwidth in a network path, while the transfer is in progress. The proposed mechanism is referred to as *SOBAS* (SOcket Buffer Auto-Sizing). The key idea in SOBAS is to limit the send window, after the transfer has saturated the available bandwidth in the path, so that the transfer does not cause self-induced congestive losses.

A second key mechanism is *MUSEPAD* (MUlti-SErver PArallel Downloads). MUSEPAD is a middleware component that coordinates the dynamic segmentation and parallel transfers of a large file from multiple servers to the same destination. MUSEPAD has some similarities with peer-to-peer (p2p) applications and CDNs, but it moves further than those systems in the direction of network awareness. MUSEPAD, running at both the client and the servers, can measure the load of the network path to each server, as well as the load of the servers, and it can detect network paths from different servers to the same client that share the same bottleneck.

As a proof-of-concept, the project will modify an existing file copying application (*bbcp*) to use the MONADD middleware. The new application (*bbcp2*) will rely on SOBAS to optimize point-to-point transfers, and on MUSEPAD to use parallel downloads from multiple servers for very large transfers. Furthermore, the project will evaluate the developed middleware experimentally over several wide-area network paths that connect various research centers and universities. Together with evaluating SOBAS and MUSEPAD, the project will compare the developed prototypes with similar applications or middleware (e.g., GridFTP, BitTorrent), as well as with new transport protocols (e.g., HS-TCP, FAST).

**Broader Impact.**
The focus area of the proposed project is data intensive distributed applications, mostly in the context of scientific and enterprise computing. The MONADD project will deliver operational middleware as well as a new data transfer application (bbcp2) to the user community. The developed middleware can gradually replace existing data transfer applications that are not network aware, or that only use a single server. The development of SOBAS will examine whether it is truly necessary to replace or modify TCP, as opposed to overcome its inefficiencies through efficient socket buffer sizing. The deployment of MUSEPAD can have a major impact on applications that distribute large data sets, by providing the means for optimized parallel transfers from multiple servers.

# 1 Introduction

The emergence of the Grid computing paradigm raises new interest in the end-to-end performance of *data intensive distributed applications*. In particular, sciences such as high-energy and nuclear physics, global weather prediction, astronomy, and bioinformatics have critical needs to share large volumes of data, already reaching the petabyte scale. Without this ability, global scientific collaborations, such as PPDG [32], BaBar [41] or LHC [28], suffer from large delays, problematic interactions, and inefficient use of resources [7].

In this project, we propose the design, development, and evaluation of *MONADD*, which stands for *Middleware for Optimized Network-Aware Data Dissemination*. MONADD is a set of middleware components that aim to dramatically improve the performance of distributed data intensive applications. The two key MONADD objectives are:

1. *Maximize the throughput of point-to-point TCP transfers.*
2. *Optimize the use of multi-server parallel downloads.*

A central feature in MONADD is its *network awareness*, meaning that the proposed middleware monitors the underlying network, including data sources and servers, in terms of various performance metrics in order to make the best possible use of the available resources [3, 43].

Typically, data intensive scientific applications run over well provisioned networks (Internet2, ESnet, GEANT, etc) built with high bandwidth links (OC-12 or higher) that are lightly loaded for most of the time. Additionally, through the deployment of Gigabit and 10-Gigabit Ethernet interfaces, congestion also becomes rare at network edges and end-hosts. With all this bandwidth, it is not surprising that Grid users expect superb end-to-end performance. However, this is often not the case. Recent measurements in Internet2 have shown that the throughput of most bulk transfers is limited to less than a few Mbps.

It is widely believed that a major reason for the relatively low end-to-end throughput is TCP. This is either due to TCP itself (e.g., congestion control algorithms and parameters), or because of local system configuration (e.g., default or maximum socket buffer size) [42]. TCP is blamed that it is slow in capturing the available bandwidth of high performance networks, mostly because of two reasons:

1. Small socket buffers at the end-hosts limit the effective window of the transfer, and thus the maximum throughput.
2. Packet losses cause large window reductions, with a subsequent slow (linear) window increase rate, reducing the transfer's average throughput.

Despite several efforts to improve or replace TCP with more efficient transport protocols [11, 20, 13, 21, 23, 26, 46], it becomes increasingly clear that it is very difficult to do so in practice. Many issues, including backward compatibility and the cost of kernel upgrades, have been a serious impediment in replacing, or even just tuning, TCP.

The MONADD project takes a significantly different approach in dealing with the inefficiencies of TCP in bulk data transfers. Instead of changing TCP, *we propose a middleware mechanism that automatically determines the optimal socket buffer size that saturates the available bandwidth in a network path, while the transfer is in progress*. The proposed mechanism is referred to as *SOBAS (SOcket Buffer Auto-Sizing)*. A first prototype of SOBAS, integrated with a simple data transfer application, has been recently described in [34]. SOBAS is based on direct measurements of the received throughput and of the corresponding RTT at the application layer. The key idea is that the send window should be limited, after the transfer has saturated the available bandwidth in the path, so that the transfer does not cause buffer overflows, i.e., *to avoid self-induced losses*.

We emphasize that *SOBAS does not require changes in TCP*, and that it can operate, in the form of middleware, with any TCP-based bulk data transfer application. Even though there have been previous attempts to perform automatic socket buffer [9, 30, 12, 38, 47], SOBAS does not require any changes in the kernel, and it can maximize the throughput of a TCP transfer [34].

Optimizing point-to-point TCP transfers with SOBAS, however, is only one direction to improve the throughput of data intensive applications. Increasingly, data intensive scientific collaborations such as HENP [32] utilize multiple servers to store and process their data. The presence of multiple servers provides another way to improve the overall application throughput, because it allows us to consider efficient server selection mechanisms, as well as schemes that optimize the segmentation of a file into multiple chunks that will be transferred from different servers in parallel.

A second key component of the MONADD project is *MUSEPAD*, which stands for *MUlti-SErver PArallel Downloads*. MUSEPAD is a middleware component that coordinates the dynamic segmentation and parallel transfers of a large file from multiple servers to the same destination. MUSEPAD is similar to peer-to-peer (p2p) applications, such as KaZaa or BitTorrent, in the use of multiple data sources for transferring different parts of a file. MUSEPAD is also similar to Content Distribution Networks (CDNs), such as Akamai, in the sense that it leverages an infrastructure of multiple servers relying on middleware to redirect a user to the best data source.

MUSEPAD, however, moves further than p2p applications and CDNs in the direction of network awareness. MUSEPAD, running at both the client and the servers, will be able to measure the load of the network path to each server, as well as the load of the servers themselves. It will also be able to detect network paths from different servers that share the same bottleneck, and avoid using those servers simultaneously. Finally, MUSEPAD will rely on statistical techniques to predict the throughput from each server [45]. Such predictions are required in order to determine how to partition a file among a number of servers so that all transfers complete at about the same time.

Together with developing middleware for SOBAS and MUSEPAD, the MONADD project will proceed in two additional directions. First, we will modify an existing file copying application (*bbcp*), developed by one of the co-PIs, to use our middleware. The new bbcp, that we refer to as *bbcp2*, will rely on SOBAS to optimize point-to-point transfers, and on MUSEPAD to use parallel downloads from multiple servers for very large transfers. bbcp2 will act as a "proof-of-concept" for the proposed middleware, but it may also become a popular application for the dissemination of large data sets in the scientific and computing communities.

Second, we will evaluate the developed middleware experimentally over several wide-area network paths that connect various research centers and universities. Such an experimental evaluation

is crucial for middleware components that rely on network awareness. Together with evaluating SOBAS and MUSEPAD, we are also going to compare our prototypes with similar applications or middleware (e.g., GridFTP, BitTorrent), as well as with new transport protocols (e.g., HS-TCP, FAST).

# 2 SOBAS

The limitations of standard TCP in utilizing high bandwidth delay paths are well understood. At the same time, modifying current TCP standard has proven to be extremely difficult. An improved throughput performance, however, can be obtained by appropriately tuning the socket buffer size for existing TCP implementations. This scheme can form a basis for middleware that uses network measurement to tune the socket buffer size and therefore optimize TCP throughput without requiring changes in the TCP stack.

## 2.1 TCP throughput and socket buffer size

In a TCP transfer the sender is allowed to have up to a certain number of transmitted but unacknowledged bytes, referred to as the *send window* $W_s$. The send window is limited by the minimum of the sender's *congestion window* $W_c$ [1], the *receive window* $W_r$, advertised by the receiver, and the size of the *send socket buffer* $B_s$. The receive window $W_r$ is limited by the receive socket buffer size $B_r$. The send window is then limited by $W_s = \min\{W_c, S\}$, where $S = \min\{B_s, B_r\}$ *is the smaller of the two socket buffer sizes*. If $T$ is the connection's RTT, the transfer's throughput is

$$R = \frac{W_s}{T} = \frac{\min\{W_c, S\}}{T} \tag{1}$$

Each link $i$ of the path $\mathcal{P}$, that the TCP transfer goes through, transmits packets with a *capacity* of $C_i$ bps. Let $\rho_i$ be the average utilization at link $i$ *prior* to the start of the TCP transfer. The *available bandwidth* $A_i$ of link $i$ is then defined as $A_i = C_i \times (1 - \rho_i)$. Adopting the terminology of [18], we refer to the link of the path $\mathcal{P}$ with the minimum available bandwidth $A = \min\{A_i\}$ as the *tight link*. The buffer size of the tight link is denoted by $B_t$. A path is *saturated* when the available bandwidth of its tight link is zero. Also, a path is *non-congested* when the packet loss rate due to congestion at its tight link is practically zero; otherwise the path is *congested*.

From Equation (1), we can view the TCP throughput as a function $R(S)$. Then, an important question is: *given a network path $\mathcal{P}$, what is the value(s) $\hat{S}$ of the socket buffer size that maximizes the TCP throughput $R(S)$?* The conventional wisdom is that the socket buffer size $\hat{S}$ should be equal to the Bandwidth Delay Product of the path, where "bandwidth" is the capacity of the path $C$, and "delay" is the RTT of the path $T$, i.e., $\hat{S} = C \times T$. Indeed, if the send window is $W_s = C \times T$, and *assuming that there is no cross traffic in the path*, the tight link becomes saturated (i.e., $A$=0) but not congested.

In practice, a network path always carries some cross traffic, and thus $A < C$. If $S > A \times T$, the TCP will saturate the tight link, build up queue at the tight link and depending on $B_t$, it may also cause packet losses. Losses, however, cause multiplicative drops in the TCP's send window, and,

potentially, throughput reductions. Thus, the amount of buffering $B_t$ at the tight link is an important factor for socket buffer sizing, as it determines the point at which the tight link becomes congested. Therefore, in a non-congested path, if a TCP connection limits its send window between $A \times T$ and $A \times T + B_t$, it will saturate the path without making it congested. In a congested path, where loss will occur irrespective of the TCP connection's send window, limiting the send window can only reduce its throughput. Therefore, in congested path the send window of the connection should be practically infinite.

### 2.2 Socket Buffer Auto-Sizing (SOBAS) mechanism

How should a socket buffer sizing scheme determine $\hat{S}$, given that it does not know a priori $A$, $T$ and $B_t$ and whether the path is congested? SOBAS is a mechanism that tries to saturate the available bandwidth of a non-congested network path, without causing a significant RTT increase. SOBAS does not require changes at the TCP protocol or implementation, and it can be deployed in the form of middleware. SOBAS does not require prior knowledge of the capacity or available bandwidth, while the RTT is measured and the presence of congestive losses are inferred directly by the application using out-of-band probing packets. SOBAS infers the point of path saturation based on periodic measurements of the received goodput at the application layer. Upon detecting path saturation, SOBAS limits its socket buffer size to the receive throughput times the RTT and therefore tries to avoid building up queues, losses and subsequent throughput reductions.

SOBAS detects the point in which the transfer has saturated the available bandwidth using two "signatures" in the receive throughput measurements: **flat-rate** and **const-rate-drop**. The **flat-rate** condition is detected when the receive throughput appears to be almost constant for some time period. The **const-rate-drop** condition occurs when SOBAS is unable to avoid self-induced losses, and it is detected as a rate drop following a short time period in which the throughput was constant.

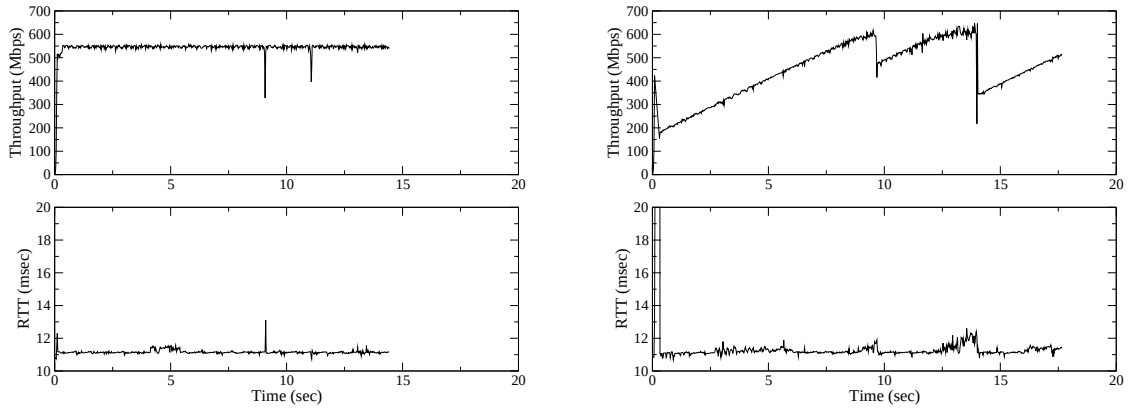
*if (***flat-rate** *or* **const-rate-drop***)*
{
*if (non-congested path)*
$S = R \times T$*; (set socket buffer size to rate times RTT)*
*else*
$S = S_{MAX}$*; (set socket buffer size to maximum value)*
}

Figure 1: Basic SOBAS algorithm.

SOBAS takes action only in non-congested paths. In paths with persistent congestion or limited available bandwidth, SOBAS will disable itself automatically, providing the same throughput as a regular TCP transfer.

Figure 2 shows an example of throughput improvement with SOBAS in a campus path at Georgia Tech. The available bandwidth in the path was about 600Mbps (at the TPC layer). Since the RTT in this path is less than 1ms, we use NISTNet[4] to create an additional delay of 10ms. Figure 2(a) and 2(b) show the time series of receive throughput and RTT for a 1GB transfer with and without SOBAS, respectively. Note that the non-SOBAS flow builds up queues, experiences throughput

reduction due to losses, and gets about 445Mbps throughput. On the other hand, SOBAS limits its socket buffer size to the point that it does not cause congestive losses, and it achieves a throughput of approximately 550Mbps.




(a) With SOBAS.

(b) Without SOBAS.

Figure 2: Example of throughput improvement with SOBAS

## 2.3 SOBAS Extensions

### 2.3.1 Dynamic SOBAS

The current implementation of SOBAS adjusts the socket buffer size only once during the transfer. This is sufficient as long as *the cross traffic is stationary*, which may be not be a valid assumption if the transfer lasts for several minutes [49]. Therefore, we propose to extend SOBAS to readjust socket buffer sizing based on the available bandwidth dynamic variations.

The dynamic version of SOBAS ("Dynamic SOBAS") would require to identify changes in the available bandwidth. A change in available bandwidth can be inferred by monitoring the received throughput and RTT without requiring external probing. A reduced throughput indicates decreased available bandwidth whereas a reduced RTT indicates increased available bandwidth. Upon the detection of a significant available bandwidth change, an appropriate socket buffer size readjustment can be performed as in the case of static SOBAS.

### 2.3.2 Parallel SOBAS in point-to-point transfers

Even in non-congested paths with stationarity cross-traffic, there can be random losses in the path. A single TCP connection with large send window can incur significant throughput reduction due to halving of the send window when it experiences such random losses. We propose to use parallel connections in conjunction with SOBAS to deal with such random losses. The main idea is, when SOBAS determines a suitable socket buffer size $\hat{S}$, instead of setting this buffer size to a single

connection, parallel SOBAS will start $N$ connections each with socket buffer size set to $\hat{S}/N$. This change will reduce the impact of a single random loss from reduction of congestion window by $\hat{S}/2$ to $\hat{S}/(2N)$.

In paths with random losses, parallel connections can clearly improve the throughput between two end-hosts. This improvement increases with the number of connections $N$. However, parallel connections introduce system overhead and raise fairness issues. For each path there is an optimal value for the number of connections $N$ that results in the maximum throughput gain. Part of our research will be to investigate how to dynamically determine the optimal value of $N$, given constraints on the maximum system overhead as well as on the fairness issue.

# 3 MUSEPAD

Optimizing the throughput of point-to-point transfers is only one direction in improving the performance of data intensive distributed applications. For very large transfers, another promising (and relatively unexplored) direction is to transfer in parallel different parts of the requested file from different servers. This approach will result in higher overall throughput as long as the bottleneck is not located at the client or at its network access link. Another requirement for the multi-server approach is, obviously, that there must be multiple servers that can store and deliver the requested file. This is increasingly the case in scientific and enterprise computing [29]. Typically, each of the collaborators or participants contributes one or more storage servers that can replicate important files or datasets for that enterprise. It is important that, in this context, the set of servers is loosely coordinated, simplifying significantly both the problems of data management and the security problems of access control and authentication. Note that this is very different than p2p systems, where any host can be a server, raising major (largely unresolved) issues regarding data integrity, unauthorized copies, and free-riders.

The idea of multi-server parallel transfers is certainly not new; for instance, see [24]. Several p2p applications, such as BitTorrent, eDonkey, Kazaa, and Morpheus, use parallel transfers from different peers [6, 10, 27]. There are major differences, however, between MUSEPAD and p2p applications. First, p2p applications do not attempt to optimize the overall throughput of parallel transfers, and they typically use very simple heuristics for file segmentation and server selection [17]. Second, p2p applications do not use network and server load measurements to select the best set of servers, or to determine how much data to transfer from each server. Third, p2p applications ignore the important issue of *shared bottlenecks* between different servers, that we will explain shortly. MUSEPAD will focus on the previous issues, leveraging the experience that resulted from p2p applications, with the objective to design an optimized and robust middleware for multi-server network aware parallel downloads.

Another related technology is that of CDNs. CDNs also use an infrastructure of multiple servers that have replicated content. However, in CDNs a client is mapped to a single server, typically the closest in terms of geographical distance or RTT [39], and so CDNs do not pursue the multi-server approach [35]. Furthermore, CDNs are mostly used for Web transfers, which are typically short, while we focus on very large transfers (hundreds of MB or more). The transfer size is crucial, as it determines both the acceptable overhead for achieving network awareness, and the potential

throughput gain using multi-server transfers.

### 3.1 MUSEPAD architecture

The MUSEPAD middleware consists of two different components: MUSEPAD-s and MUSEPAD-c, running at the servers and clients, respectively.

The role of MUSEPAD-s is to coordinate the set of machines that are used as servers for a certain collaboration or enterprise. MUSEPAD-s keeps track of which servers are currently active or available. It also keeps track of which servers maintain a copy of each file in the system. Even though the servers can be completely replicated, this does not need to be the case. In general, a file may be located only in certain servers. The file lookup operation can be performed through table lookups, if the number of stored files is not very large, or through distributed lookup techniques such as those used in p2p applications, if larger scalability is needed [5, 8]. MUSEPAD-s interacts with MUSEPAD-c when the latter needs to know which servers to contact for downloading a file. MUSEPAD-s also interacts with MUSEPAD-c to achieve the objectives of network awareness, as described next.

The role of MUSEPAD-c is twofold. First, it provides an interface to the application, hiding all the details of the multiple parallel downloads. The application simply provides a descriptor of the requested file, an identifier for the federation of servers that store that file, and a file pointer where the downloaded file should be written to. Second, MUSEPAD-c aims to maximize the overall throughput, or equivalently to minimize the transfer duration. To do so, the parallel transfers from different servers should complete at roughly the same time, meaning that *the assigned transfer size from each server should be proportional to the expected throughput from that server*.

To determine the *target set* (i.e., the servers that will be used for downloading the file), as well as the *server target range* (i.e., the file segment to be downloaded from that server), MUSEPAD-c works with MUSEPAD-s at the servers. Specifically, MUSEPAD-c first contacts MUSEPAD-s at the *authoritative server* for that client, requesting the set of servers that store the requested file. That is the initial target set. Note that the authoritative server for each client can be a configuration parameter of MUSEPAD-c.

Then, MUSEPAD-c enters the *preliminary download phase*. In that phase, MUSEPAD-c starts downloading parts of that file from all the servers of the target set, monitoring at the same time the achievable throughput from each server. The preliminary download phase lasts for only a few seconds, and it is used to predict both the average achievable throughput from each server, but also the statistical variation of those throughput measurements. Furthermore, during that phase, MUSEPAD-c performs active network measurements, in collaboration with MUSEPAD-s at each of the target servers, to detect the presence of shared bottlenecks in the paths from two or more servers to the client. The presence of shared bottlenecks has to be taken into account in estimating the throughput from each server. MUSEPAD-c also collects measurements from the target servers regarding their load. Servers that are heavily loaded or sit behind congested paths, or servers that would not add to the overall throughput due to shared bottlenecks, are excluded from the initial target set.

At the end of the preliminary download phase, MUSEPAD-c determines the target range for

each server of the (potentially reduced) target set. The target range is determined based on the predicted throughput from each server. Note that the preliminary download phase was not a waste of time for the transfer, because it contributed to the file downloading.

Finally, MUSEPAD-c will keep monitoring the resulting throughput from each server of the target set, watching for significant variations. Such variations can be caused by route changes, or by major changes in the network or server load. Upon the detection of such events, MUSEPAD-c can re-optimize the remaining of the transfer by adjusting the target set and/or the remaining target range for each server.

Figure 3 shows two clients and a set of servers during the preliminary phase. Note that even though both clients request the same file, they are mapped to different target sets for server load balancing. Figure 4, on the other hand, shows what happens after the preliminary phase. Note that the size of the target range for each server is proportional to the available bandwidth from that server to the corresponding client. Also note that the target set of the second client has been reduced from two servers to one; this can happen, for instance, because the two servers share the same bottleneck to that client.

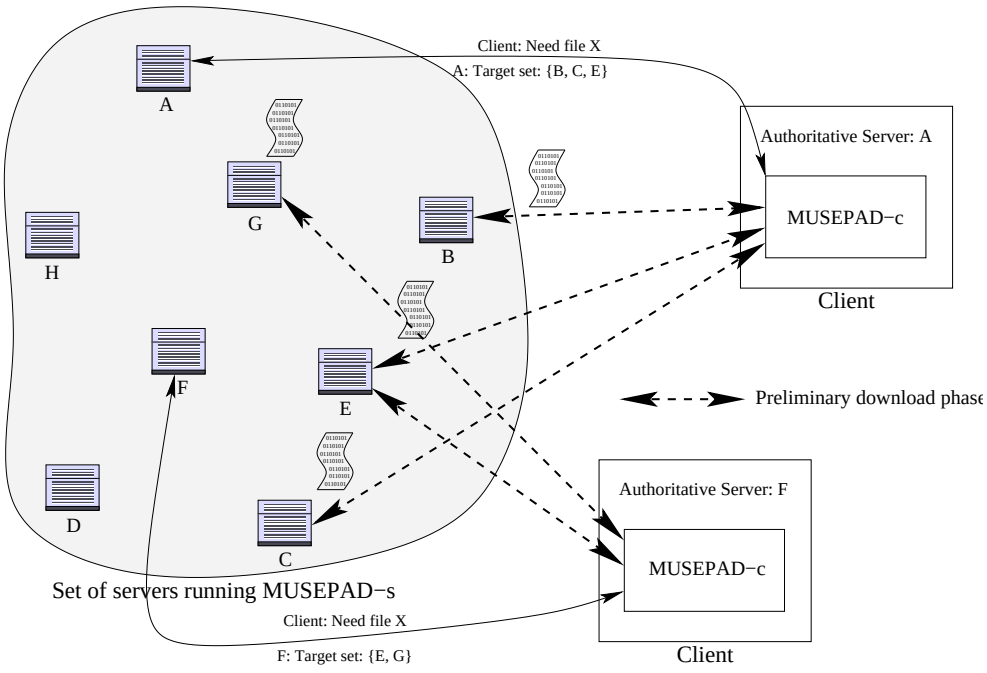



Figure 3: MUSEPAD architecture: preliminary download phase.

## 3.2 Shared bottlenecks

As mentioned earlier, including an additional server at the target set of a transfer will increase the overall throughput only if the path from that server to the client does not share the same bottleneck with another server in the target set (see figure 5(a)). Techniques to detect shared bottlenecks in network paths have been recently developed [2, 22, 37].

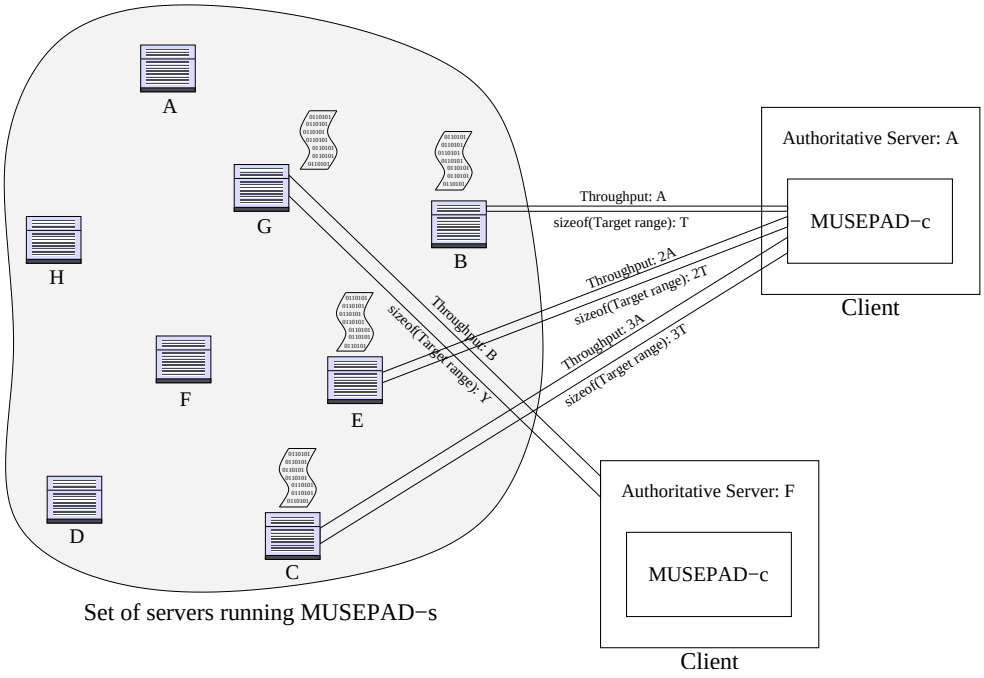
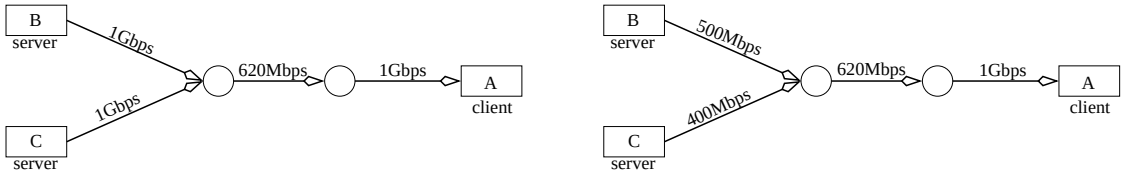



Figure 4: MUSEPAD architecture: main download phase.




(a) Shared primary bottleneck.

(b) Shared secondary bottleneck.

Figure 5: Example of two paths with either a shared primary bottleneck or a shared secondary bottleneck. Links are annotated with their available bandwidth.

In a similar scenario that we refer to as secondary bottleneck, shown in figure 5(b), the bottleneck from any of the two servers to the client is different than the bottleneck if both servers are used in parallel. The simultaneous use of those two servers can increase the aggregate throughput in this case, but not as much as if there was no sharing. Note that the aggregate throughput from a set of servers, in the presence of secondary bottlenecks, cannot be estimated simply as the available bandwidth of each of the individual paths. Instead, more elaborate estimation techniques are required that can measure the aggregate available bandwidth in a set of network paths. As part of our research on this problem, we will develop such techniques, extending our previous work on end-to-end estimation of network properties [19, 33].

# 4 Proof-of-concept: bbcp2

*bbcp* is an existing popular data transfer application developed by one of the co-PIs (Andrew Hanushevsky) at SLAC [15]. We propose to develop *bbcp2*, an extension of bbcp, to act as a proof-of-concept for the MONADD middleware.

bbcp accounts for the majority of data transferred to and from SLAC to other sites in the world. It is also used by other commercial and non-commercial institutions to improve their data dissemination throughput. bbcp is available as freeware under a Berkeley-style software license. It is an ideal starting point for our project, because it has an object-oriented design that allows for substantial modification while maintaining compatibility with previous versions. Salient features of bbcp include 3rd-party copy facilities, TCP buffer size negotiation, parallel streams, extensive monitoring capabilities via NetLogger, disk I/O optimization, restart of failed copies, port stability for external monitoring, firewall avoidance options, Quality of Service support, optional compression, optional MD5 checksumming for enhanced error detection, and extended device support (e.g., memory-to-memory). Perhaps the strongest feature of bbcp that makes it ideal for this endeavor is its peer-to-peer implementation: no server is needed to perform a copy. This allows for tests in a wide variety of environments without the need of administrative intervention. Finally, the primary developer of bbcp is a co-PI in this project.

The key extensions to bbcp2 will be to use the services provided by SOBAS and by MUSEPAD. Specifically, bbcp2 will:

1. Interact with Dynamic SOBAS to determine the optimal socket buffer size, as well as the number of parallel connections to an individual server.
2. Allow the interfacing with alternative TCP stacks and user-space UDP-based reliable transport protocols, for comparison purposes and experimental evaluation.
3. Interact with MUSEPAD to receive the addresses of the servers that will be used in parallel, as well as the amount of data to be transferred from each server.

With these enhancements, bbcp2 can be used to provide the data transport capabilities needed to effectively use statically distributed data. This would immediately make bbcp2 a powerful yet practical tool for wide-scale data dissemination. Many scientific collaborations already have the means, although not always sophisticated, to locate replicated copies of required data. bbcp2 allows these copies to be effectively, yet simply, used to substantially enhance data transfer throughput. Simplicity is critical for wide-spread use and sets the stage for more sophisticated mechanisms to be employed to enhance data dissemination applications.

In the second half of the project, bbcp2 will be enhanced to deal with *a dynamic set of distributed data sources*. The goal is to provide a general data transport mechanism that can be used by any number of peer-to-peer network architectures to improve throughput beyond that afforded by having multiple download sites. As such, bbcp2 will be enhanced to allow for real-time steering of a multi-source to single destination copy operation.

In order to partially address security concerns, we propose to capitalize on the protocol-agnostic security framework of bbcp. In order to achieve peer-to-peer functionality, bbcp merely requires

that a secure application launch-vehicle be used to start the copy operation. For instance, the widely deployed secure shell, *ssh*, can be used for general portability; or the Grid Security Infrastructure version, *ssh-gsi*, for Grid compatibility.

# 5 Experimental evaluation and comparison with similar work

Before any new transport protocols or middleware are deployed, they need careful evaluation to determine their applicability in diverse environments, including both ultra high-speed testbeds and high-speed production networks. They also need to be compared with other alternatives so recommendations can be made on their relative benefits. To be effective in that direction, we will move to pilot deployment in production applications (e.g., a large HENP experiment like BaBar). This will further help to evaluate the applicability of the MONADD middleware in the real world, wring out deployment problems, get feedback and support from scientists and regular users, and assist in getting traction for wider scale deployment.

We will work with our partners in the IEPM-BW project to setup a testbed of at least 3 paths that span a wide range of RTTs (say 10ms, 70ms and 170ms). These paths will be over production networks and will have bottlenecks of at least 622 Mbits/s (OC-12). We will setup ssh access to two high-performance ($>$ 2GHz) hosts at each end of these paths, and ensure the hosts are appropriately configured. In particular, we will ensure that appropriate maximum TCP buffers are set, install various utilities to assist in making measurements, and verify the latest versions of servers such as *IPerf*, *Pathload*, *UDT*, *ABwE*, *bbcp* and *bbftp* (we will also investigate whether the host has access to *GridFTP*).

At the SLAC end, we will procure two fast Linux hosts ($>$ 3GHz with GE interfaces) to make throughput and cross-traffic measurements from. We will select the most appropriate advanced TCP stacks and install them. The selection will be based on earlier measurements and will include criteria such as achievable throughput, fairness, stability, CPU utilization, ease of use and installation etc. We will also install UDT and file copy applications including bbcp, bbftp, and GridFTP. We will develop a methodology for the testing, including automated scripts to configure NICs, TCP stacks, and applications, start and stop applications, gather, analyze and report on the measurements.

With the above setup we will check and compare the performance of SOBAS with each existing and proposed TCP stack. We will report the incremental throughput in 5-second intervals, and observe the impact on CPU utilization and ping RTTs. We will analyze and report on stability, as well as on the effect of new transfers entering and leaving the network on existing traffic. We will also investigate the impact of cross traffic on SOBAS as well as on different TCP stacks, and the impact of traffic in the reverse path.

We will repeat the above measurements with bbcp, bbcp2, bbftp, and GridFTP. Besides providing direct feedback to our own bbcp2, we will also work with and provide feedback to the developers of the various TCP stacks and UDP reliable transports. We will evaluate and report on their performance, fairness, stability, resource utilization, and ease of integration deployment, and will appropriately encourage their development.

To understand the performance beyond the 1Gbps limit we will procure a state of the art, fast PC

running Linux. We will install an existing 10GE Network Interface Card (NIC). Tests will be made between hosts on high-performance paths on a 10GE experimental testbed involving networks such as UltraLight, the DoE UltraScienceNet, StarLight, NetherLight and CERN/DataTAG. The high speed of these links will probably mean that the disk/file system bottlenecks will limit performance to less than 10Gbps. However, the bbcp feature enabling memory to memory copies will help us to evaluate the use of bbcp2 in those environments.

As we get closer to final hardened versions of our middleware, we will extend the evaluation to utilize the 40 high speed production paths set up as part of the IEPM-BW infrastructure. This infrastructure has a wide range of bottleneck speeds from 10 Mbits/s to 900Mbits/s and sites in 9 countries. bbcp2 will be added to the set of probes used in IEPM-BW, and measurements will be made on a regular basis between a few tens of hosts on a wide range of paths. The testing will include bbcp, and bbcp2 using memory-to-memory and disk-to-disk transfers, Iperf with a standard TCP stack with one and multiple streams, an advanced TCP stack chosen from the previous measurements, a reliable UDP transport, and an available bandwidth measurement tool such as ABwE. When modified versions of bbftp and GridFTP become available, we will integrate them into the IEPM-BW toolkit and also make tests on them.

# 6 Previous related work

## 6.1 TCP modifications and replacements

Researchers have worked on improving the throughput of large TCP transfers for several years, pursuing mostly three approaches: TCP modifications [11, 20, 21, 23, 26, 46], parallel TCP transfers [40, 14], and automatic buffer sizing [9, 30, 38, 12]. Changes in TCP or new congestion control schemes, possibly with cooperation from routers [21], can lead to significant benefits for both applications and networks. However, modifying TCP has proven to be quite difficult in the last few years. Parallel TCP connections can increase the aggregate throughput that an application receives. This technique raises fairness issues, however, because an aggregate of $N$ connections decreases its aggregate window by a factor $\frac{1}{2N}$, rather than $\frac{1}{2}$, upon a packet loss. Also, the aggregate window increase rate is $N$ times faster than that of a single connection. Finally, techniques that automatically adjust the socket buffer size can be performed at the application-layer, and so they do not require changes at the TCP implementation or protocol. We have adopted the automatic socket buffer sizing approach because it can be implemented at the middleware level and it is easier to deploy than changing TCP.

Most of the advanced TCP stacks are implemented in kernel space. This requires access to the kernel sources and root privileges to build. Furthermore, the stack upgrades are not synchronized with OS patches, upgrades, etc. This makes deployment difficult; in fact, as of now none of the new TCP kernels exist in commercial OSs. Thus, there is considerable interest in efficient *user-space reliable transport protocols*. UDT is such a promising reliable transport protocol, it is built on UDP rather than TCP and runs in user space [13]. UDP implementations tend to be less efficient than TCP and use more compute cycles per bit transferred. This is a serious concern, since for the emerging 10Gbits/s paths, compute power is a gating factor to high performance.

## 6.2 File transfer/copy applications

Today, production-level high-performance file transfer applications such as bbcp, bbftp and GridFTP utilize TCP. To achieve high performance these applications provide options such as large TCP send and receive buffers/windows and parallel TCP streams. Today these performance options are set once at the start of the transfer and not modified to track path performance during the transfer. There are also UDP-based file transfer applications such as RBUDP [16] and Tsunami [44], however, they have not achieved any notable production usage in the data-intensive science and Grid communities.

## 6.3 Multiserver transfers and CDNs

A key technical advance in p2p applications, such as KaZaa/Morpheus, eDonkey/eMule, BitTorrent, and DirectConnect, has been the breaking of large media files into chunks, allowing the p2p clients to download chunks from multiple servlets in parallel. This has allowed for improving the aggregate throughput of downloading large objects. The selection of chunk sizes is driven by individual site considerations, it is often far from optimal.

A key point in the existing work on p2p systems has been that it is driven largely by an open-source community (with signal exception of KaZaa) and the primary motivation has been to get around the difficulties of large scale delivery of fat content in the face of legal troubles, bandwidth constraints, free-riding etc. Our project faces none of the above constraints and is more closely related to the problem of content distribution. Traditional CDNs arose in the context of the World Wide Web to reduce the overhead on busy Websites. The various models of CDN delivery and their effectiveness in reducing the latency perceived by the user has been examined in [25].

The Logistical Networking project [31] also enables the use of multiple sites by providing metadata to indicate the locations where the data may be found. The initial selection of sites from which to transfer the data requires a knowledge of the available bandwidth during the potential expected transfer time. Armed with such predictions it is possible to estimate how much data to transfer from each of several sites (each with potentially different available bandwidth predictions over the period of interest) so all the transfers finish at roughly the same time [36].

## 6.4 Network awareness

The Network Weather Service [48], and other related projects, make measurements of various metrics, such as server load, RTT, or loss rate, and then use various prediction techniques to estimate the throughput of a particular server and network path. One difficulty with this approach is collecting measurement data that are relevant to the source and destinations for the planned transfers. An alternative is to make the required measurements on demand. However this may require special facilities to be set up at the sources and destinations together with the privileges to use them. Another attractive alternative is to use measurements made during the transfer itself (e.g. from the application itself or if available from the underlying transport layer itself, for instance via Web100 [Web100], or even from the network devices) to adjust the transfer method and to provide estimates

for the remaining transfer time. This requires a new breed of middleware bulk-data transfer/copy utilities that are network aware.

# 7 Deliverables, distribution of work, and halfway milestone

## Deliverables

The main deliverables of the MONADD project will be:

1. Software implementation of SOBAS middleware
2. Software implementation of MUSEPAD middleware for servers
3. Software implementation of MUSEPAD middleware for clients
4. Software implementation of bbcp2
5. Research publications and technical reports describing the architecture, implementation, and performance of the developed middleware.

All software prototypes will be publicly available under the GNU licence. The prototypes will be ported to all Unix and Linux variants.

## Distribution of work

The three primary investigators and senior personnel (Dovrolis, Cottrell, Hanushevsky) are going to collaborate on every component of the project. The main responsibility of investigator, however, will be as follows:

- **Dovrolis (Georgia Tech)**, together with the graduate students and the programmer working on this project, will be primarily responsible for the design, implementation, testing, and maintenance of SOBAS and MUSEPAD.
- **Cottrell (SLAC)**, together with the graduate students working on this project, will be primarily responsible for the experimental evaluation of the produced middleware as well as for the comparison with similar middleware, applications, and transport protocols.
- **Hanushevsky (SLAC)** will be primarily responsible for the design, implementation, testing, and maintenance of *bbcp2*, and he will collaborate with Dovrolis in the design of the API for SOBAS and for MUSEPAD.

### 18th-month Milestone

A major milestone for the project is the end of the 18th month At that point, we are going to demonstrate the use of bbcp2 jointly with dynamic SOBAS in maximizing the throughput of point-to-point transfers. At the same milestone, we are going to demonstrate the basic functionality of MUSEPAD in identifying the best set of servers for each client, as well as in detecting shared network bottlenecks.

In the second phase of the project, our focus will be on the integration of bbcp2 with MUSEPAD, the evaluation of the resulting systems in wide-area networks, and the improvement of the software based on user-feedback from the first phase.

### Deployment plan

One of the co-PIs (L. Cottrell) is an active participant at the BaBar collaboration and he works closely with several researchers from the physics community (especially HENP) in both USA and Europe. In the first phase of the project, Cottrell will be instrumental in convincing physicists at SLAC, Caltech, IN2P3 (in Lyon, France), and in a few other research centers in USA and Europe to install our middleware (alpha version) in a few nodes, allowing field experimentation and providing user feedback. Additionally, we are going to experiment and evaluate the developed middleware in PlanetLab.

In the second phase of the project, we are going to make the developed middleware publicly available (beta version). Additionally, we will expand the deployment in HENP research centers, collecting feedback from a wider user community. To provide further visibility to our software, we are going to demonstrate its use and performance in some key conferences in the scientific community, in high-performance distributed computing conferences, as well as in forums such as SuperComputing.

# Facilities, Equipment, and Other Resources

## Georgia Tech

The PI C. Dovrolis is a member of the Center for Experimental Research in Computer Systems (CERCS) at Georgia Tech. CERCS has a variety of facilities located in the College of Computing and in the School of Electrical and Computer Engineering. All of these facilities are available for use by the PI and his students.

The CERCS facilities include the following systems:

- Several Intel clusters: The newest system being installed is a 16-node cluster of 8-processor Pentium Xeon nodes, networked with both Gigabit Ethernet and Lanai 7 Myrinet. Others are: 16 8-way 550MHz PIIIs interconnected by gigabit ethernet and myrinet, 40 dual 300MHz PIIs interconnected by fast ethernet, and 16 quad 200MHz PPros interconnected by fast ethernet and myrinet,
- Specialized ASAN cluster with 8 dual 1.7GHz PIVs, connected via dual gigabit ethernet cards attached to IXP1200 routers acting as network interface cards; multiple gigabit switches and additional Intel-basd machines used for experimentation with different network interface boards and software,
- Cluster of 8 Sun UltraSPARC dual-processors typically used as a traffic generator and/or for small experiments, with ATM interconnects available for experimentation,
- Sun and Intel-based server systems, including storage and compute servers,
- Over 30 SUN- and Intel-based workstations, including multimedia workstations,
- High-capacity networking switches to serve as multiprocessor interconnect fabric between all gigabit equipped computing nodes; features jumbo frame support and link aggregation to meet performance requirements,
- Access Grid node connected to other such U.S. facilities vis Internet2, performance computational resources, and
- Terabyte capacity, high-performance disk-array based system with quad processors and gigabit connectivity to support media stream and high capacity realtime data research.

In addition to the above, CAD tools running on additional high end SUN and HP platforms, board testing equipment, and specialized equipment for hardware design are located in multiple labs focused on hardware design, hardware/software co-design, and other topics in computer architecture.

All of the CoC's facilities provide switched GigE infrastructure. A high-performance multi-gigabit backbone is employed within the CoC and for connectivity to the campus network. The campus Internet connection is provided by dual 155 Mbps commodity Internet links and a 2.4 Gbps link to the Internet2 Abilene research network through the regional GigaPOP hosted at the Institute (SoX.net, the Southern Crossroads). Additional computing facilities are provided by OIT, including five public-access clusters of Dell and Apple workstations and a collection of Sun multi-processors which are treated as a unified computational resource.

## SLAC

SLAC is the home of the BaBar HENP experiment. BaBar was recently recognized as having the largest database in the world. In addition to the large amounts of data, the SLAC site has farms of compute servers with over 3000 cpus. The main (tier A) BaBar computer site is at SLAC. In addition, BaBar has major tier B computer/data centers in Lyon, France, near Oxford, England, Padova, Italy and Karlsruhe, Germany which share TBytes of data daily with SLAC. Further BaBar has 600 scientist and engineer collaborators at about 75 institutions in 10 countries. This is a very fertile ground for deployment and testing of new bulk-data transfer utilizing improved TCP stacks and Grid replication middleware. There are close ties between the SLAC investigators, the BaBar scientists and the SLAC production network engineers.

The IEPM-BW infrastructure, developed at SLAC, has in 5 countries 10 monitoring sites and about 50 monitored sites with contacts, accounts, keys, software installed etc. This provides a valuable testbed for evaluating new TCP stacks etc. The SLAC site has high speed connections (OC12 and GE) to the CENIC/Abilene and ESnet backbones. The SLAC IEPM group has a small farm of 6 high-performance network test hosts with 2.5 to 3.4GHz cpus and 10 GE Network Interface Cards (NICs). SLAC hosts network measurement hosts from the following projects: AMP, NIMI, PingER, RIPE, SCNM, and Surveyor. SLAC has two GPS aerials and connections to provide accurate time synchronization.

SLAC has an OC12 Internet connection to ESnet, and a 1 Gigabit Ethernet connection to Stanford University and thus to CalREN/Internet 2. We have also set up experimental OC192 connections to CalRENII and Level(3). The experimental connections are currently not in service, but have been successfully used at SC2000-2003 to demonstrate bulk-throughput rates from SuperComputing to SLAC and other sites at rates increasing over the years from 990 Mbits/s through 13 Gbps to 23.6 Gbps. SLAC is also part of the ESnet QoS pilot with a 3.5 Mbps ATM PVC to LBNL, and SLAC is connected to the IPv6 testbed with three hosts making measurements for the IPv6 community1. SLAC has dark fibers to Stanford University and PAIX, SLAC plans to connect at 10Gbits/s to the DoE UltraScienceNet and UltraLight testbeds later this year. The SLAC IEPM group has access to hosts with 10Gbits/s connectivity at UvA on the NetherLight network in Amsterdam, at StarLight in Chicago and CERN in Geneva. We also have also close relations with Steven Low's group at Caltech and plan to get access to their WAN-in-Lab setup for testing applications with dedicated long distance fiber loops. SLAC has been part of the SuperComputing bandwidth challenge for the last 3 years, part of the team that won the bandwidth challenge last year for the maximum data transferred, and two-time winner of the Internet2 Land Speed Record.

As part of our previous and continuing evaluations of TCP stacks, the SLAC IEPM team has close relations with many TCP stack developers (in particular the developers of FAST, H-TCP, HSTCP-LP, LTCP) and with the UDT developers.

## References


[1] M. Allman, V. Paxson, and W. Stevens. *TCP Congestion Control*, April 1999. IETF RFC 2581.

[2] A. Bestavros, J. Byers, and K. Harfoush, "Robust identification of shared losses using end-to-end unicast probes," In *Proceedings of the 8th IEEE International Conference on Network Protocols*, Nov 2000.

[3] J. Bolliger and T. Gross, "Bandwidth monitoring for network-aware applications," In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, 2001.

[4] M. Carson and D. Santay, "NIST Net - A Linux-based Network Emulation Tool," *ACM Computer Communication Review*, vol. 33, no. 3, pp. 111–126, July 2003.

[5] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like p2p systems scalable," In *Proceedings of ACM SIGCOMM*, Aug 2003.

[6] B. Choen. *Incentives Build Robustness in BitTorrent*. http://bitconjurer.org/BitTorrent/bittorrentecon.pdf, May 2003.

[7] C.Liu, L.Yang, I.Foster, and D.Angulo, "Design and evaluation of a resource selection framework for grid applications," In *Proceedings of the 11th IEEE Symposium on High-Performance Distributed Computing (HPDC-11)*, 2002.

[8] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," In *Proceedings of ACM SIGCOMM*, Aug 2002.

[9] T. Dunigan, M. Mathis, and B. Tierney, "A TCP Tuning Daemon," In *Proceedings of SuperComputing: High-Performance Networking and Computing*, November 2002.

[10] F. L. Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulie, "Clustering in Peer-to-Peer File Sharing Workloads," In *Proceedings of the 3rd International Workshop on Peer-to-Peer Computing*. Lecture Notes in Computer Science, Springer-Verlag, February 2004.

[11] S. Floyd. *HighSpeed TCP for Large Congestion Windows*, December 2003. RFC 3649 (experimental).

[12] M. K. Gardner, W.-C. Feng, and M. Fisk, "Dynamic Right-Sizing in FTP (drsFTP): Enhancing Grid Performance in User-Space," In *Proceedings IEEE Symposium on High-Performance Distributed Computing*, July 2002.

[13] R. Grossman, "UDT: An Application Level Transport Protocol for Grid Computing," In *PFLDnet 2004*, Feb 2004.

[14] T. J. Hacker and B. D. Athey, "The End-To-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network," In *Proceedings of IEEE-CS/ACM International Parallel and Distributed Processing Symposium*, 2002.

[15] A. Hanushevsky, A. Trunov, and R. L. Cottrell, " Peer-to-peer computing for secure high performance data copying ," In *Proceedings of Computing in High Energy Physics*, 2001. http://www.slac.stanford.edu/~abh/bbcp/.

[16] E. He, J. Leigh, O. Yu, and T. DeFanti, "Reliable Blast UDP: Predictable High Performance Bulk Data Transfer ," In *IEEE International Conference on Cluster Computing (CLUSTER'02)*, Sep 2002.

[17] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. A. Hamra, and L. Garces-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime," In *Proceedings of the 5th Passive and Active Measurement Workshop*, April 2004.

[18] M. Jain and C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," In *Proceedings of ACM SIGCOMM*, pp. 295–308, August 2002.

[19] M. Jain and C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 537–549, August 2003.

[20] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," In *Proceedings of IEEE INFOCOM*, March 2004.

[21] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," In *Proceedings of ACM SIGCOMM*, August 2002.

[22] D. Katabi, I. Bazzi, and X. Yang, "A passive approach for detecting shared bottlencks," In *IEEE International Conference on Computer Communications and Networks*, Arizona, 2001.

[23] T. Kelly, "Scalable TCP: Improving Performance in Highspeed Wide Area Networks," *ACM Computer Communication Review (CCR)*, April 2003.

[24] D. Kostic;, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pp. 282–297, 2003.

[25] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the Use and Performance of Content Distribution Networks," In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, OCT 2001.

[26] R. Krishnan, C. Partridge, D. Rockwell, M. Allman, and J. Sterbenz, "A Swifter Start for TCP," Technical Report BBN-TR-8339, BBN, March 2002.

[27] N. Leibowitz, M. Ripeanu, and A. Wierzbicki, "Deconstructing the kazaa network," In *Proceedings of the Third IEEE Workshop on Internet Applications*, 2003.

[28] *Large Hadron Collider Home Page* . http://lhc-new-homepage.web.cern.ch/lhc-new-homepage/.

[29] V. Mann and M. Parashar, "Middleware support for global access to integrated computational collaboratories," In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, 2001.

[30] M. Mathis and R. Reddy. *Enabling High Performance Data Transfers*, January 2003. Available at: http://www.psc.edu/networking/perf_tune.html.

[31] T. Moore, M. Beck, A. Bassi, and J. S. Plank, "The logistical backbone: Scalable infrastructure for global data grids," In *Asian Computing Science Conference 2002*, Hanoi, Vietnam, Dec 2002, Springer Verlag.

[32] *Particle Physics DataGrid* . http://www.ppdg.net/.

[33] R. S. Prasad, C. Dovrolis, and B. A. Mah, "The Effect of Layer-2 Store-and-Forward Devices on Per-Hop Capacity Estimation," In *Proceedings of IEEE INFOCOM*, 2003.

[34] R. S. Prasad, M. Jain, and C. Dovrolis, "Socket buffer auto-sizing for high-performance data transfers," *To appear in Journal of Grid Computing, Special Issue on High Performance Networking*, 2004.

[35] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," In *Proceedings of IEEE INFOCOM*, 2002.

[36] T. G. Robertazzi, "Ten reasons to use divisible load theory," *IEEE Computer*, vol. 36, no. 5, pp. 63–68, May 2003.

[37] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, , Jun 2002.

[38] J. Semke, J. Madhavi, and M. Mathis, "Automatic TCP Buffer Tuning," In *Proceedings of ACM SIGCOMM*, August 1998.

[39] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of DNS-based server selection," In *Proceedings of IEEE INFOCOM*, pp. 1801–1810, 2001.

[40] H. Sivakumar, S. Bailey, and R. L. Grossman, "PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks," In *Proceedings of SuperComputing: High-Performance Networking and Computing*, November 2000.

[41] *SLAC : BaBar*. http://www.slac.stanford.edu/BFROOT/.

[42] B. Tierney, "TCP Tuning Guide for Distributed Applications on Wide Area Networks," *USENIX & SAGE Login*, February 2001.

[43] B. L. Tierney, D. Gunter, J. Lee, M. Stoufer, and J. B. Evans, "Enabling network-aware applications," In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, 2001.

[44] *Tsunami Home page* . http://www.indiana.edu/ anml/anmlresearch.html.

[45] S. Vazhkudai and J. M. Schopf, "Predicting sporadic grid data transfers," In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11)*, 2002.

[46] R. Wang, M. Valla, M. Y. Sanadidi, and M. Gerla, "Using Adaptive Rate Estimation to Provide Enhanced and Robust Transport over Heterogeneous Networks," In *Proceedings IEEE ICNP*, November 2002.

[47] E. Weigle and W.-C. Feng, "A Comparison of TCP Automatic Tuning Techniques for Distributed Computing," In *Proceedings IEEE Symposium on High-Performance Distributed Computing*, July 2002.

[48] R. Wolski, N. Spring, and J. Hayes, "The network weather service: A distributed resource performance forecasting service for metacomputing," *Journal of Future Generation Computing Systems*, vol. 15, no. 5-6, pp. 757–768, Oct 1999.

[49] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the Constancy of Internet Path Properties," In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, pp. 197–211, November 2001.