BEYOND MUSIC SHARING: AN EVALUATION OF PEER-TO-PEER DATA DISSEMINATION TECHNIQUES IN LARGE SCIENTIFIC COLLABORATIONS

by

SAMER AL KISWANY

B.Sc. Jordan University of Science and Technology, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

December 2007

ABSTRACT

The avalanche of data from scientific instruments and the ensuing interest from geographically distributed users to analyze and interpret it accentuates the need for efficient data dissemination. An optimal data distribution scheme will find the delicate balance between conflicting requirements of minimizing transfer times, minimizing the impact on the network, and uniformly distributing load among participants. We identify several data distribution techniques, some successfully employed by today's peer-to-peer networks: staging, data partitioning, orthogonal bandwidth exploitation, and combinations of the above. We use simulations to explore the performance of these techniques in contexts similar to those used by today's data-centric scientific collaborations and derive several recommendations for efficient data dissemination.

Our experimental results show that the peer-to-peer solutions that offer load balancing and good fault tolerance properties and have embedded participation incentives lead to unjustified costs in today's scientific data collaborations deployed on over-provisioned network cores. However, as user communities grow and these deployments scale, peer-to-peer data delivery mechanisms will likely outperform other techniques.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor, Dr. Matei Ripeanu. This work would not have been possible without his guidance. His insight, support and enthusiasm are very much appreciated.

I wish to acknowledge my collaborators Adriana Iamnitchi - at University of South Florida (USF) and Sudharshan Vazhkudai at Oak Ridge National Laboratory (ORNL), for their collaboration and helpful technical advice.

I would like to thank Sathish Gopalakrishnan for his effort in reviewing this work and providing helpful comments and technical advice.

Sincere thanks go to my dear friend Enad Mahmoud, whom help and support was key in passing the most critical moments in my graduate study. Thanks also go to Rafed Al Masri, Hani Khasawneh, and Mohammad Malkawi, for their friendship, and continues encouragement.

I am grateful to the Networked Systems research group students, Elizeu Santos-Neto, Abdullah Gharaibeh, Armin Bahramshahry, Hesam Ghasemi, and David Boen, for their friendship, support, and insightful comments.

I wish also to acknowledge my teachers and constant supporters at the Computer Engineering Department at Jordan University of Science and Technology (JUST). Special thanks go to Professor Sameer Bataineh, Dr. Omar Al-jarrah, and Dr. Mohammad Al-rousan, for their support and honest advice through my student life. Thanks also go to Dr. Moad Mowafi, Dr. Fayez Idris, Dr. Fahed Awad, Dr. Ali Shatnawi, Dr. Abdulla Bataineh, Dr. Taisir Eldos, and Dr. Saleh Abdel-hafeez.

Through my graduate study I faced exceptionally challenging situations; it was impossible to continue my graduate study without the encouragement and support of few people. First and foremost on this list are my parents, Ishaq and Nabilah, my wife, Dima, and my brothers, Khaled and Bilal. Their unconditional love and confidence in my abilities are deeply appreciated.

Last, but certainly not least, I am grateful to Dr. Deepa Kundur, and Dr. Scott Pike, at Texas A&M University (TAMU), for their help and support through the hardest times in my graduate study. Their help and patience are deeply appreciated.

To My Father, Ishaq
My Mother, Nabilah
My Wife, Dima

1. Introduction

Today's Grids provide the infrastructure that enables users to dynamically distribute and share massive datasets. A growing number of instruments and observatories are generating petabytes (10^{15} bytes) of data that need to be analyzed by large, geographically dispersed user communities, requesting now more than even efficient data-dissemination solutions. Examples of data-intensive collaborations include the European Council for Nuclear Research (CERN) Large Hadron Collider (LHC) experiment [1], neutron scattering at the Spallation Neutron Source (SNS) [2], the DØ experiment at Fermi National Accelerator Laboratory or the Southern California Earthquake Center earthquake simulators – TeraShake [3] and CyberShake [4]. Enabling the formation of these collaborative data federations are ever increasing network capabilities including high-speed optical interconnects (e.g., TeraGrid [5], LambdaGrid [6]) and optimized bulk transfer tools and protocols (e.g., GridFTP [7], IBP [8]).

However, most data distribution strategies currently in place involve explicit data movement through batch jobs [9, 10] that are seldom sympathetic to changing network conditions, congestion and latency, and rarely exploit the collaborative nature [11] of modern-day science.

At the same time, peer-to-peer file sharing and collaborative caching efficiently exploit patterns in users' data-sharing behavior. For instance, one approach is to split files into blocks and transfer them separately, possibly using different network paths, as in BitTorrent [12]. Another approach is to exploit the orthogonal bandwidth that might be available outside a traditional, source-rooted data-distribution tree [13]. Aforementioned techniques

offer several benefits such as increased throughput, good use of network resources, and resilience in the face of link and node failures. Furthermore, such techniques have been deployed [12] and studied [14, 15] in the context of peer-to-peer file sharing and application-level multicast.

However, such techniques may not be directly adaptable to Grid settings because of the different usage scenarios, workloads, or resource properties. For example, an important usage scenario is the dynamic distribution of subsets of data available at one site to one or many target locations for real-time analysis and visualization. This is, for example, the processing mode for TeraBytes of National Aeronautics and Space Administration (NASA) satellite hyperspectral data that need to be processed in near real time [16]. Another example is the concurrent visualization approach in visualizing large simulations or experiments [17]. In this approach the intermediate results of large scale simulation are sent for online analysis and visualization. Concurrent visualization has three main advantages: 1) it enables the scientist to view the current state of the long running simulation and facilitates run time monitoring and steering, 2) it allows higher temporal resolution since it obviates the storage space requirement, 3) it is the only choice for some time critical systems like hurricane tracking and forecasting. This approach is adopted in NASA supercomputer massively parallel forecast model [17] often used to forecast the progress of hurricanes, consequently online analysis and visualization of the simulation results is critical.

Two conflicting arguments compete for designing one-to-many delivery of large-size scientific data over well provisioned networks. On one side, there is the intuition that well-provisioned networks are sufficient for guaranteeing good data-delivery performance:

sophisticated algorithms (such as in peer-to-peer systems) that adapt to unstable or limited-resource environments are superfluous and add unjustified overheads.

The flip side is the argument that advanced data dissemination systems are still required as the high data volumes and the relatively large collaborations create contention and bottlenecks on shared resources. Additionally, even if contention for shared resources is not a serious concern, the question remains whether networks are over provisioned and thus advanced data dissemination techniques induce unnecessary costs.

This debate motivates our study. We explore experimentally the solution space for one-to-many large-scale data delivery via simulations with real-world parameters. We consider solutions typically associated with peer-to-peer applications (such as BitTorrent [12] or Bullet [13]) and evaluate them in the large data federations scenario. To this end, we use three real topologies of production Grid testbeds in our simulations: LCG [18], EGEE [19] and GridPP [20].

1.1. Contributions

The contribution of this study is threefold. First, this study quantitatively evaluates and compares a set of representative data-delivery techniques applied to realistic grid environments. The quantitative evaluation is then used to derive well-supported recommendations for choosing data-dissemination solutions and for provisioning the Grid network infrastructure. Further, our study contributes to a better understanding of the performance tradeoffs in the data-dissemination space. To the best of our knowledge, this is the first, head-to-head comparison of alternative data dissemination solutions using multiple performance metrics: time to complete the dissemination of data, generated overhead, and load balance.

Second, in addition to comparing the data dissemination solutions along multiple success metrics, we provide, to the best of our knowledge, the first quantitative evaluation of the fairness of these solutions and their impact on competing traffic. Our results show that this impact can be significant and suggest that fairness should be an important factor when choosing a dissemination solution and the right network management infrastructure.

Third, a byproduct of this study is a simulation framework that can be used to explore optimal solutions for specific deployments and can be extended for new dissemination solutions.

1.2. Summary of Findings

Data dissemination in science grids environments is characterized by relatively small collaborations (tens to hundreds of participating sites), large data files to transfer, well-provisioned networks, and collaborative participants.

Our simulation-based experimental investigation on seven solutions selected from a set of successful Internet data delivery and peer-to-peer deployed systems shows the following: First, the Grid deployments we studied are over-provisioned, which leaves little space for the intelligent techniques to show their strength. Secondly, application-level schemes such as BitTorrent, Bullet and application-level multicast perform best in terms of dissemination time. However, they introduce high-traffic overheads. Thirdly, the naive solution of separate data transfers from source to each destination performance drops dramatically when the available bandwidth in the network core decreases. In such cases, adaptive peer-to-peer like techniques that are able to exploit multiple paths existing in the physical topology can offer good performance.

1.3. Research Publications

This work resulted in two refereed publications, and one journal submission. These publications were written in collaboration with Adriana Iamnitchi, at University of South Florida, and Sudharshan Vazhkudai, at Oak Ridge National Laboratory. The following is the list of the three publications:

- Samer Al-Kiswany, Matei Ripeanu, Adriana Iamnitchi, and Sudharshan Vazhkudai, "Beyond Music Sharing: An Evaluation of Peer-to-Peer Data Dissemination Techniques in Large Scientific Collaborations", Submitted to the Journal of Grid Computing.
- Samer Al-Kiswany, Matei Ripeanu, Adriana Iamnitchi, and Sudharshan Vazhkudai, "Are P2P Data-Dissemination Techniques Viable in Today's Data-Intensive Scientific Collaborations?", The 13th International Euro-Par Conference, European Conference on Parallel and Distributed Computing, August 28-31, 2007, Rennes, France. (acceptance rate 89/333 = 27%)
- Samer Al-Kiswany and Matei Ripeanu, "A Simulation Study of Data Distribution Strategies for Large-scale Scientific Data Collaborations", 20th IEEE Canadian Conference on Electrical and Computer Engineering, Vancouver, BC, April 2007.

Despite the short publication period, this work is cited by two research projects: GridTorrent [21] and scheduling data intensive bag of tasks [22], and been listed as a recommended reading by a third research group [23].

1.4. Thesis Structure

The rest of this thesis is organized as follows. Chapter 2 presents the data usage characteristics of scientific collaborations and compares them with the assumptions of

peer-to-peer file-sharing systems. Chapter 3 surveys existing work on data dissemination and describes in detail the dissemination schemes this study analyzes. Chapter 4 discusses the different evaluation approaches for comparing a set of dissemination techniques, and presents a survey of related work, Chapter 5 presents the design of our simulator and Chapter 6 presents our evaluation results. We summarize our findings in Chapter 7.

2. Data in Scientific Collaborations

Three key differences make it difficult to predict the behavior of adaptive techniques employed in peer-to-peer systems when applied to scientific data federations: scale of data, data usage characteristics, and resource availability.

The **scale of data** poses unique challenges: scientific data access consists of transfer of massive collections (TeraBytes), comprising of hundreds to thousands of GigaByte sized files. For instance, of the more than one million files accessed in DØ between January 2003 and May 2005, more than 5% are larger than 1GB and the mean file size is larger than 300MB [24]. This is more than 20 times larger than the 14MB average file size transferred in the Kazaa network in 2002 as reported in [25], but within the same order of magnitude with the files currently transferred by BitTorrent (mainly, movies and software distributions): Bellissimo *et al.* [26] report an average file size of 600MB. Another example is the TeraShake [27] earthquake simulations, A single simulation may generate 47 terabytes of data that is analyzed and visualized using TeraGrid [5].

Usage of data in scientific communities is of a different *intensity* compared to other communities. For example, the 561 scientists part of the DØ project processed more than 5PB of data between January 2003 and May 2005, which translates to accessing more than 1.13 million distinct data files and a sustained data processing rate of 65MB/s [24]. Additionally, *popularity distributions* for scientific data are more uniform than in peer-to-peer systems with a significant impact on caching effectiveness. For example, while in DØ a file is requested by at most 45 different users, a BitTorrent file can be requested by thousands of users or more [26].

Another difference in data usage is *co-usage*: often, in scientific environments, files are used in groups and not individually. Taking the high-energy physics project DØ as a case study again, each data analysis job accessed on average 108 files, with a maximum of more than 20,000. The need for simultaneous access to multiple files stresses the problems brought up by the large file size, requesting transfers of data collections in the order of TB. For example, the largest ten datasets in the DØ traces analyzed in [24] are between 11 and 62 TB.

Finally, **resource availability** in Grids poses smaller challenges than in peer-to-peer networks. Computers stay connected for longer, with significantly lower churn rate and higher availability due to hardware characteristics and software configurations.

At the same time, data federations are overlays built on top of well-provisioned (sometimes over-provisioned) network links (e.g., TeraGrid) as opposed to the commercial Internet. In particular, network cores are well provisioned, often with multiple Gbps links.

Yet another difference in resource availability is that in scientific collaborations resource sharing is often enforced by out-of-band means, such as agreements between institutions or between institutions and research funding agencies. For this reason, mechanisms that enforce participation, such as the tit-for-tat scheme in BitTorrent, may impose unnecessary overheads and may limit the overall system performance.

All these properties (huge size transfers, well provisioned networks, more stable resources, cooperative environments) invite the question whether peer-to-peer data distribution strategies will result in tangible gains on the well-endowed network infrastructures on which today's Grids are deployed. A careful study is necessary to derive

recommendations for constructing and provisioning future testbeds and choosing efficient dissemination approaches to support scientific collaborations.

3. Data Distribution: Solutions and Metrics

The naive solution for data dissemination is to set up an independent transfer channel between each data source and destination pair. Although this technique is clearly not the most efficient and overloads the data source, it is often adopted in current deployments [9].

A second well-understood solution is to use IP multicast. However, despite significant efforts, IP multicast is not widely deployed as it requires new routing hardware, and faces challenging problems providing reliability, congestion and flow control [28]. An additional set of reasons for IP multicast's limited deployment is its limited support for group management, including authorization for group creation, receiver authorization, and sender authorization, distributed address allocation, and support for network management [29].

To provide an alternative, numerous research projects have explored data dissemination solutions at the application level. This section provides a classification of data distribution techniques (Section 3.1), details the representative techniques we have selected to explore in depth in this thesis (Section 3.2), and presents the criteria over which data dissemination solutions are typically evaluated (Section 3.3).

3.1. Classification of Approaches

This section identifies three broad categories of techniques used in application-level data dissemination systems: data staging, data partitioning, and orthogonal bandwidth harnessing. Existing data dissemination solutions often use combinations of these techniques. The rest of this section describes these techniques in detail in the context of our target environment.

3.1.1. Data Staging

With data staging, participating nodes are used as intermediate storage points in the data distribution solution. Such an approach is made feasible by the emergence of network overlays. For instance, it is becoming increasingly common practice in the Internet community for application-specific groups to build collaborative networks, replete with their application-level routing infrastructure. This is based on the premise that sophisticated applications are better aware of their resource needs, deadlines, and associated constraints and can thus perform intelligent resource allocation and workload/data transfer scheduling. In this vein, peer-to-peer file sharing systems can be viewed as data-sharing overlays with sophisticated application-level routing performed on top of the traditional Internet [30].

Similarly, in scientific data-analysis communities, user collaboration patterns and shared interest in data lead to a 'natural' way to structure an overlay. In data grids, data staging is a trend encouraged by the increasing significance of application-level tuning of large transfers. For instance, collaborating sites often gather intelligent routing information through the use of Network Weather Service [31] or GridFTP probes [32]. Such information is then used to make informed decisions regarding routes, such that data transfers can be executed in an optimized fashion based on a delivery constraint schedule [33]. A logical extension is thus to use the participating sites as intermediary *data staging* points for more efficient dissemination.

Additionally, a data distribution infrastructure can include a set of intermediary, strategically placed resources (as in logistical computing [34] [35]) to stage data. In this thesis we study idealized version of logistical multicast [36] as the representative exponent of this class of solutions. We simulate an idealized, optimal IP-level logistical multicast

infrastructure that includes infinite buffering capabilities associated with all the intermediate routers. This idealization aims to quantify the maximum benefits data staging can offer when used in isolation.

3.1.2. Data Partitioning

To add flexibility, various peer-to-peer data distribution solutions split files into blocks and transfer these blocks independently (e.g., BitTorrent [12], Bullet [13], SPIDER [37] and many other systems). Much like the aforementioned application-level routing, this approach allows applications a greater degree of control over data distribution. Further, it enables application-level error correction: for example, in the case of downloading a file from multiple replicas, partitioning can be coupled with erasure coding to achieve fault tolerance. Several of these techniques are used in production systems (e.g., Digital Fountain [38, 39]).

Partitioning techniques can have significant value in a data-grid collaboration setting. For instance, there is a genuine need to provide application-level resilience when it comes to data transfers. Bulk data movement in the Grid usually involves transfers of large files that are required to be resilient in the face of failures such as network outages or security proxy expiration. This prompted us to include two representative systems that use data partitioning in our study: Bullet [13] an academia developed data dissemination solution and the widely popular BitTorrent file sharing protocol [12].

3.1.3. Orthogonal Bandwidth Exploitation

Once a basic file partitioning mechanism is in place, it can then be used to exploit orthogonal bandwidth. Thematic here is the use of alternate network paths to speed-up data transfers. The reason is that, in many cases, the bandwidth available to a traditional source-

routed distribution tree can be augmented using additional 'orthogonal' network paths that exist between its interior and leaf nodes.

This is the premise in a number of commercially deployed or academically designed data distribution systems. Orthogonal bandwidth tapping relies on partitioning files into blocks and, initially, sending each block to a different peer with the intent that peers would then form pair-wise relationships and acquire from each other the data they are missing. Such an approach works as a means both to exploit the residual bandwidth available at the peer and, more importantly, to employ alternate network routes than would not have been available in a single source distribution scenario. Many peer-to-peer networks owe much of their success to such optimizations (e.g., BitTorrent).

Intuitively, it appears that what we described so far will offer commensurate gains when applied to Grid data collaborations. However, several of these optimizations are designed to work in a naturally competitive environment such as the Internet, where peers contend for bandwidth. One question we address is how this intuition translates when the bandwidth is plentiful, as is the case with modern data collaborations with heavily provisioned networks.

3.2. Candidate Solutions for Evaluation

For our experimental study, we selected previously proposed solutions from each of the categories presented above. We also include other traditional, well-understood techniques as a base for our comparison. The following subsections present a brief description for each of the solutions we evaluate.

3.2.1. Logistical Multicast

Logistical multicast (LMT) [36] (as described earlier in Section 3.1.1), depends on the logistical networking approach [34]. Logistical networking infrastructure incorporates strategically placed nodes in an overlay to expedite data distribution; these strategically placed nodes may serve as routers and switches in addition to providing storage services [35]. Internet Backplane Protocol [8, 40] is the protocol that implements the storage service primitives (for instance, allocating storage space, storing, and reading) that the applications can use to access the logistical storage space. LMT [36] uses topology information to construct a multicasting tree from a single source to multiple destinations using the intermediate logistical nodes. The intermediate nodes are used to buffer and stage the data to the destinations. The multicasting tree is described in a form of a schedule of data transfers between the tree nodes. The topology information used in constructing the tree is gathered by network measurement modules [41] or services already deployed in the system, for instance bandwidth measurement between PlanetLab nodes [42, 43] .We evaluate an idealized version of this approach: we assume that logistical storage is associated with each router in our topologies, and that intermediary storage nodes have infinite storage capacity. With these idealizations, the logistical multicast version we evaluate offers an upper bound for the performance of data dissemination solutions based on source-rooted distribution trees.

3.2.2. Tree Based Application-Level Multicast

Tree based application-level multicast (ALM) solutions organize participating nodes into a source-rooted overlay tree used for data dissemination [44, 45]. Each node maintains information about the other nodes in the tree it is connected to. Data routing algorithms are

trivial as data is simply passed down the tree structure. Since, in our case, participating nodes are end-nodes with an interest in long-term data storage, recovering lost blocks and flow control can be simply implemented for each tree branch.

What differentiates various ALM solutions is the algorithm used to build and maintain the distribution tree. These algorithms can be classified based on multiple criteria: the ownership of the participating resources, their approach to decentralization, their use of a structured or unstructured overlay, and the performance metric that is optimized.

- *Resource ownership and infrastructure.* Some systems rely on strategically placed infrastructure proxies to support the construction of their data distribution trees (Overcast [45], OMNI [46], application level SPIDER [37]), while others aim to integrate end-nodes without infrastructure support (Narada [28], ALMI [44]).
- *Overlay structure.* Some dissemination tree construction algorithms assume the existence of a structured overlay substrate (e.g. CAN multicasting [47], Scribe [48], SplitStream [49]).
- *Centralized vs. distributed tree construction.* For small and medium scale systems centralized tree construction and management algorithms based on full system view have been designed (e.g., ALMI [44]). At the other end of the spectrum, systems based on structured overlays are able to handle millions of nodes.
- *Optimization function.* While some dissemination tree construction algorithms strive to provide the highest possible bandwidth, other algorithms aim to minimize the resulting overheads in terms of additional message delay or generated network traffic (e.g. Narada [28], NICE [50], OMNI [46]), or try to load balance the dissemination effort between the overlay nodes [49].

Recently, a number of studies have proposed data dissemination algorithms targeting the Grid infrastructure. Grido [51], for example, builds a shortest-path-first tree based on a virtual coordinates system that advise each node of its nearby neighbors. Another system, Multicast Optimizing Bandwidth (MOB) [52], adopts a hierarchical approach; nodes are organized into clusters, and intra-cluster transfers are preferred to inter-cluster transfers to reduce overheads. The nodes exchange data within the cluster and between the clusters in a BitTorrent like approach (see BitTorrent description in Section 3.2.5). MOB assumes that clustering information is available and globally known to all the nodes.

For our evaluation we choose a centralized solution based on global topology view (similar to ALMI [44]) appropriate for the scale we target and offering near optimal trees. Our algorithm constructs a bandwidth-optimized ALM tree without assuming strategically placed proxy nodes or the presence of a structured overlay substrate. The reason to choose a bandwidth-optimized tree construction is that the data transfer time-to-finish is often considered the main data dissemination success metric in our environment. Appendix A presents in detail the tree construction heuristic and analyzes its complexity.

3.2.3. SPIDER

SPatial Indirection for Path Diversity for Expedited Replication (SPIDER) [37] offers a set of heuristics that enables fast content distribution by building multiple source-rooted trees (assuming global views). This way, SPIDER can exploit existing orthogonal bandwidth. This technique can be used at the application as well as at lower network layers.

SPIDER builds a set of trees, and tries to maximize the aggregate bandwidth of all the constructed trees. SPIDER builds one tree at a time in a non-greedy fashion. Meaning that SPIDER does not try to maximize the constructed tree bandwidth as the case in single tree

construction algorithms and some multi-tree constructions algorithms (for instance, FPFR [53] which is described next). In constructing a tree SPIDER tries to maximize the residual bandwidth left for the other trees to be constructed. For this, the construction algorithm selects from a set of candidates the link that leaves the maximum outgoing bandwidth for its source node. Appendix B details the SPIDER tree construction algorithm.

A number of other algorithms are based on the same principle of building a set of source-rooted trees to exploit the orthogonal bandwidth available. For example, Fast Parallel File Replication (FPFR) tool [53] constructs multiple, source-rooted multicasting trees by repeatedly using depth-first search to find a tree spanning all hosts. For each tree, bandwidth as high as the bottleneck bandwidth is "reserved" on all links used in the tree. The search for new trees continues until no more trees spanning all hosts can be found. The data to be distributed is then multicast in fixed-size blocks using all trees found.

For our simulations, we consider a scenario where SPIDER algorithm is used at network layer, which offers an upper bound for solutions based on multiple source-rooted trees. Note that, in this case, when SPIDER is able to build only a single tree it is equivalent to traditional IP-multicast.

3.2.4. Bullet

Bullet [13] offers a way to exploit orthogonal bandwidth by initially distributing disjoint subsets of data on different paths of a distribution tree (we use the ALM-built tree in this study). After this step, nodes pair up and exchange missing blocks to complete the file distribution. Bullet also depends on the source rooted tree in exchanging the control messages, more precisely, to aggregate fixed size node content summaries from leafs to the source node. The source, in turn, distributes a random subset of these summaries in a fixed

size blocks down the tree. Peers use these summaries to discover the blocks they are interested in at other peers in the system. Further, in the pairwaise exchange between peers, the exchange initiator decides which blocks to send to the destination depending on the summary of the destination node. This push-based solution generates duplicate traffic since incomplete summaries at the initiator node lead to the possibility of transmitting duplicate blocks to a destination.

3.2.5. BitTorrent

BitTorrent [12] is a popular data distribution scheme that exploits the upload bandwidth of participating peers for efficient data dissemination. Participating nodes build transitory pair-wise relationships and exchange missing file blocks. BitTorrent employs two main algorithms in its operation: chocking, and rarest fist algorithms. The chocking algorithm selects the set of peers to reciprocate with. The chocking algorithm keeps track of download rate provided by other peers, and periodically selects a set of peers offering the highest download rate to reciprocate with (send blocks to). Moreover, periodically it selects one peer optimistically. This optimistic selection (unchocking) allows nodes to discover other peers, and helps the new joining nodes to obtain their first blocks. The rarest first algorithm is a simple algorithm for selecting the next block to download from the peers. Through this approach the node will try to download first the block that is least replicated among its neighbors. This approach minimizes the possibility of losing the rare blocks and tries to uniformly distribute all the blocks for better performance.

BitTorrent assumes a non-cooperative environment and employs a tit-for-tat (through the chocking algorithm) incentive mechanism to discourage free riders. Additionally, nodes are selfish: each node selects its peers to minimize its own time to acquire content,

disregarding the overall efficiency of the data distribution operation. Consequently, a node will serve data to the peers that serve back in return useful blocks at a high rate.

Other solutions. Finally, to offer a basis for comparison, we also simulate IP-multicast and the naive approach of using independent transfers from the source to each destination.

3.3. Success Metrics

Multiple categories of success metrics can be defined for most data management problems. The relative importance of these metrics is dependent on the application context. Thus, no data distribution solution is optimal for all cases and a careful evaluation of various techniques is required for choosing a solution appropriate for a specific application context and deployment scenario. Performance objectives include:

- *Minimizing transfer times.* Transfer time is often a key metric for data dissemination due to the need to send all data to all destinations so that real-time processing at the end-sites can progress smoothly [54]. The focus typically is on minimizing the average, median, N^{th} percentile, or the highest transfer time to destination.
- *Minimizing the overall impact on the network.* For advanced, dynamic data dissemination techniques that build sophisticated distribution trees and exploit all available network routes, it is vital to evaluate their overall impact and their impact on bottleneck links. A success metric tied to the network effort might involve minimizing the load on bottleneck links, the amount of duplicate data transferred, or the aggregate network 'effort' (in megabit x mile transferred) in the distribution tree.
- *Load balance.* With the enlisting of end-nodes in the data dissemination effort, evenly spreading the load among participants becomes an important goal. The load balance

metric evaluates how well each dissemination mechanism balances load among participating nodes.

- *Fairness* to other concurrent transfers may be an important concern depending on the lower-layer network and the protocols used. Fairness is especially important considering that most of today's networked applications are TCP friendly. In this case, since TCP aims to provide a fair share of the available bandwidth to each data flow, using multiple flows for a single application will strongly affect other concurrent applications operating in a single flow mode.

4. Evaluation Approaches

Three different approaches are possible to evaluate the set protocols described in Section 3.2, namely: Analytical, deployment, and simulation. The next three sections detail these approaches, and present a survey of related work.

4.1. Analytical Evaluation

The analytical approach tries first to create a model for the data dissemination it then analyzes the model using mathematical theorems to prove the properties and estimates the performance of the dissemination technique. The defined model should define: how the nodes are connected, the nodes characteristics, and any dissemination technique specific properties.

Different analytical models have been proposed to study different file distribution protocols in real-life scenarios. Biersack *et al.* [55] analyze the performance of optimal application-level tree file-distribution-algorithms as will as a parallel tree structure. In case of a single tree the authors conclude that as the number (k) of children per node increases, the percentage of the nodes uploading data decreases. This is because only one of k peers will be uploading data, and interior node must upload the entire file k times. The analysis also shows that the parallel trees algorithm outperforms the single tree algorithm. Through their analysis, they assume simplifying assumptions that affect the accuracy of the results. They assume a homogeneous network where all the nodes have the same upload and download bandwidth, and the network core is over provisioned to sustain $n.k$ concurrent flows where n is the number of nodes and k is the number of peers per node.

Qiu and Srikant [56] present a fluid model of BitTorrent-like networks, in which they confirm BitTorrnet scalability (that is the average download time is not dependent on the node arrival rate), and that the rarest-first-policy is efficient in uniformly distributing the file blocks.

For analytical studies however, as for any modeling exercise, the main tradeoff is between the complexity of the model and its ability to capture all system details. Consequently, it is often necessary to simplify, sometimes unrealistically, the model in order to make the analysis computationally tractable. In a large dynamic system like a data dissemination system it is unrealistic to assume a homogenous set of nodes, infinite capacity network core, or to ignore the traffic competition in the network core. Without these assumptions, on the other side, analytical models that accurately model heterogeneity, physical network topologies, and network contention quickly become intractable.

4.2. Deployment-based Evaluation

The deployment approach depends on deploying the unmodified implementation of the techniques under study on a testbed of comparable size and capabilities with the target system. Considering this study, the deployment approach suggests setting up the candidate techniques on a testbed like PlanetLab [42] and evaluate the performance of the techniques under real workloads.

This approach is plausible since the reported results are result of unmodified technique implementation on a real-life large-scale testbed. But this approach is not suitable for all the success metrics we are interested into, mainly due to the presence of uncontrolled effects found in such testbeds. Considering this project target success metrics presented in Section 3.3, it is impossible to fairly compare the performance of the candidate techniques on a

testbed like PlanetLab. This is because these solutions differ in the way they perform in the presence of concurrent traffic. For instance, assume that we are interested in comparing two protocols over a simple testbed of two nodes: source and destination. The first technique opens a single data channel to the destination, while the second opens n data channels to the destination. Given that most of these techniques and the other applications on the internet use TCP or TCP-friendly protocol, the performance of these two techniques is not comparable if there is a concurrent competing traffic between the two nodes. The reason is that the second approach may perform better simply due opening n connections to the destination, and hence being less fair in sharing the link bandwidth with other applications, and not because of the technique's mechanisms.

To the best of our knowledge there is no deployment-based study that compares a set of dissemination techniques. Instead, a number of measurement-based studies have been conducted to evaluate the performance of deployed systems. These studies depend on log files collected by a central server in the system, or on statistics collected by modified nodes participating in the dissemination network.

Pouwelse *et al.* [57] present a measurement-based study of BitTorrent system. Their measurement software is composed of two main components. The first component monitors a selected tracker (The central repository server in BitTorrent dissemination network), its mirrors and .torrent file servers (servers serving the exchanged-file metadata). The second component tracks the nodes participating in a certain file exchange. While their study confirmed BitTorrent ability to deal with flash crowds, the necessity for decentralizing the tracker components in the system, and to add incentives for seeding, it does not present file-

distribution related measurements as average download time, load balancing, and network stress.

Izal *et al.* [58] present a similar study. In their work, they analyze two log files for a popular-file torrent; the first log file was generated by file tracker, while the second log file was generated through running a peer node to participate in this same torrent. Their finding confirms BitTorrent ability to scale well and to handle flash crowds in real life scenarios, while providing reasonable download rate. However, this study is limited in scope. First, it depends only on single tracker log file for a single file exchange. Second, the client log file is generated through using a client connected to a 10Mb connection, which hinders the generality of the results. Finally, due to the limited information found in log files, no results are reported regarding load balancing between peers or protocol overhead.

4.3. Simulation-based Evaluation

The simulation approach like the analytical approach starts with modeling the target system. Unlike the analytical approach the model in the simulation approach can capture the system important details without worrying about how easy it is to analyze the model mathematically. The amount of details modeled is affected by: the target success metrics, the target accuracy level, and the amount of resources available for the simulation. At one end of this approach lies the faithful simulation of the target technique, for instance using the technique unmodified implementation in a packet level simulator like *ns* [59]. On the other end lies the high level simulation of the technique, such as simulating the technique high level data flow over a network of infinite capacity.

Few simulation studies attempt to compare different protocols. These studies generally focus on the effect of protocol mechanisms and parameters on the protocol performance.

Bharambe *et al.* [15] study the effect of the BitTorrent different mechanisms on the system performance. While their findings agree with our simulation results, their simplifying assumptions leave some unanswered questions. For example, in their network model, the authors model only the uplink and downlink links in the network without considering the links in the core of the network. While this assumption could be justified by considering the over provisioned connections found in most networks cores (so the bottleneck links are only on the network edges), it is unclear if this is still the case with multiple flows sharing the same limited number of core links. In addition, the study does not consider the protocol's overhead, nor it addresses the issue of fairness to other competing traffic, which we believe are key metrics to consider when selecting a protocol for scientific collaboration system.

While BitTorrent attracted researchers' attention due to its wide deployment, other data dissemination systems have not attracted similar attention. In order to evaluate the techniques above, we have built a simulator that works at the file-block level. We believe that our simulator enables a first study to directly compare a multitude of dissemination protocols. A detailed discussion of our model and simulation approach is presented in Chapter 5.

5. Simulating Data Dissemination

In order to evaluate the techniques above, we have built a simulator that works at the file-block level. This chapter presents the set of decisions that guided our simulator design (Section 5.1), presents key details about simulating the techniques we chose to investigate (Section 5.2), discusses the scope of our simulations (Section 5.3) and evaluates and compares our simulator with similar simulators in the literature (Section 5.4).

5.1. Simulator Design

We have built a high-level simulator to investigate the performance of different data distribution protocols. As for most simulators, the main tradeoff we face is between the resource volume allocated to simulation and the fidelity of the simulation. At one end of the possible design spectrum are packet-level simulators (such as *ns* [59]) and emulators (such as ModelNet [60] or Emulab [61]): they require significant hardware resources but model application performance faithfully by running unmodified application code and simulating/emulating network transfers at the IP-packet level. At the other end of the spectrum are high-level simulators that abstract the application transfer patterns and employ only coarse network modeling [62]. For example, a commonly used approach is to model the Internet as having limited capacity access links and infinite bandwidth at the core. Another example is replacing packet-level simulation (computationally expensive as it implies simulating each packet's propagation through a series of router queues and network links) with flow-level simulation. This requires lower computational resources as the characteristics of the network paths are computed once per data flow. Flow-level simulations

enabled simulating multicasting trees with hundreds of destinations without considerably reducing the results accuracy [62].

Our simulator sits in between the two extremes above. At the application level the granularity is file-block transfer, a natural choice since many of the data dissemination schemes we investigate use file blocks as their data management unit. At the network simulation level, while we do not simulate at the packet level, we do simulate link level contention between application flows.

In addition, our simulator design and implementation is guided by the following set of decisions:

- *Ignore control overheads.* For our target scenario, i.e., distribution of large files, the generated control traffic is orders of magnitude lower than useful payload. Similarly, the additional delay incurred while waiting for control commands and synchronization on control channels is minimal compared to actual data transfer delays, especially given that that control messages overlap or are often piggybacked. As a result, we do not attempt to estimate control channel overhead and do not model the delay it introduces.
- *Use of global views.* Our simulator uses a global view of the system in order to hide algorithmic details that are not relevant to our investigation. Thus, following our high-level simulation objective, the simulator replaces decentralized configuration algorithms (e.g., for building application-level dissemination trees) with their centralized alternatives that use global views. As a consequence, the performance of centralized versions we simulate using global views is an upper bound of the performance of original distributed versions.

- *Isolated evaluation.* The data dissemination solutions we compare put a different stress on the network and, consequently, when evaluated in a competitive environment, may offer better apparent performance simply by being more unfair to competing traffic. To overcome this problem, and enable fair, head-to-head evaluation, we perform our evaluation experiments in two steps. We first evaluate each dissemination solution in isolation (Sections 6.2, 6.3, and 6.4) then we compare their impact on competing traffic (Section 6.5).

5.2. Simulating Data Dissemination Techniques

We experiment with the four solutions for data dissemination described in Section 3.2: application-level multicast (ALM), BitTorrent, Bullet, and logistical multicasting. To provide intuition about their efficiency, we compare them with two base cases. First, we consider the base case of IP-multicast distribution (and its improvement using SPIDER heuristics). IP-multicast, although not guaranteed to always offer the minimal transfer times, is optimal in terms of network usage and node load-balance. SPIDER tries to build multiple IP multicast trees in order to optimally exploit the bandwidth through different paths from the source to destinations; consequently, SPIDER performs identically to IP multicasting on sparse topologies where it can build only one tree.

The second base case evaluates the naïve (yet popular) data dissemination solution where the source sends a copy of the file separately to each node. For this case, the simulator uses the best IP path to send data from the source to each destination.

For these solutions as well as for all tree-based solutions, the simulator analyses the topology at hand, determines the routing paths and the flow contention at the physical link level and estimates data transfer performance.

For the most complex protocols, Bullet and BitTorrent, the simulator models each block transfer independently. This is necessary due to the non-deterministic nature of these data dissemination solutions. The simulations use a default block size of 512 KB (as in deployed BitTorrent systems [12, 63]). We experimented with multiple block sizes but since the block size does not have a significant impact on performance, we do not include these results in here.

The simulator is composed of three main modules: routing, peering, and block transfer. As their names indicate, the routing module is responsible for running the routing protocol for all internal nodes in the topology; the peering module is responsible for constructing peering relationships between nodes according to the protocol specification; and, finally, the block transfer module uses the information provided by the two other modules to transfer the blocks between peers using the paths provided by the routing module while accounting for link level contention. The peering module uses a global view: every node is fully informed about the content of every other node, which slightly improves Bullet and BitTorrent performance.

In more detail, after routing paths between nodes are selected (using a shortest path algorithm), the simulation works in rounds for the two dissemination solutions that use temporary peering between nodes, i.e., Bullet and BitTorrent: in each round, first, the peering algorithm is executed, adding or deleting new pairs of nodes that exchange data. At this stage, with these two pieces of information (the set of pairs of nodes exchanging data in the next round and the routing paths between them), network contention is simulated on each physical link and the number of blocks to be transferred between each pair of nodes is found. Next, the set of blocks to be exchanged are selected, and finally the blocks are

simulated to propagate between the peers. Note that, while the routing module is invoked only once at the beginning of the simulation, the peering and the block transfer modules are invoked in every cycle, thus driving the overall simulation speed.

5.3. The Scope of the Simulation Study

In order to focus on the objectives of our study, we limit the axes over which we vary parameters to the strictly required ones. This does not impact the validity of our results, since we are making the same assumptions uniformly for all solutions we compare. As a result:

- Consistent to our controlled deployment environment assumption, we do not attempt to quantify the impact of node and link volatility.
- We do not quantify the impact of imperfect information (we compare instantiations of algorithms that use global, complete system views where required).
- We do not investigate the scalability of these schemes (though all have been shown to work well at the scale of today's Grid deployments).

5.4. Simulator Evaluation

We designed our simulator with strong emphasis on accuracy and less on simulation performance. This section presents the complexity of the simulation for the most compute intensive strategies, evaluates the simulator performance in a practical setting, and compares the simulator performance with that of other simulators used for similar studies.

The simulator for IP-multicasting, ALM, logistical multicasting, SPIDER and independent transfers from the source to every node, simulates the high-level deterministic

protocol behavior. Consequently it is less complex than BitTorrent and Bullet simulators, which use block level simulation.

While Bullet and BitTorrent simulators use the same routing module, each has a complex peering and block transfer module reflecting the protocol's characteristics. Since these are the most complex protocols we simulate, they limit the size of the physical topologies we can explore. Table 5.1 details the complexity of each module for these two protocols.

Module	Bullet	BitTorrent
Routing	$O(E^3*L)$	$O(E^3*L)$
Peering	$O(E^2*B*Log(B))$	$O(E^2*B.Log(B)+E^3)$
Block Transf.	$O(E*P*B)$	$O(E*P^2+E*P*Log(N))$

Table 5.1. Table 1. The complexity of Bullet and BitTorrent protocol's modules. Notations: E - the number of end nodes; L - the number of links in the physical network topology; B - the number of file-blocks; P - the number of peers per node.

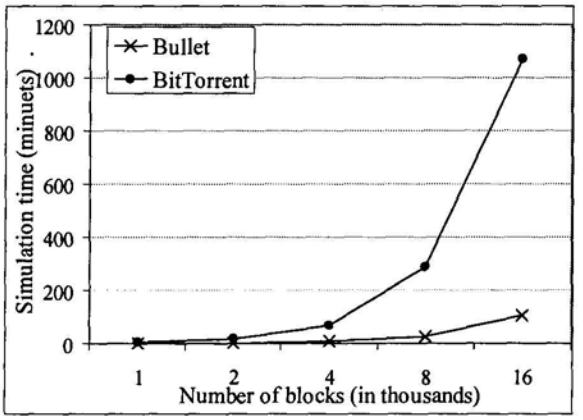

Figure 5.1. Simulation time for a 25 nodes topology and 1GB file.

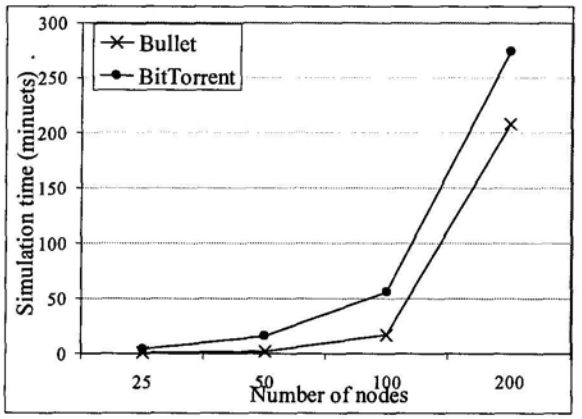

Figure 5.2. Simulation time for disseminating a 1 GB file (divided into 2000 blocks).

Table 5.1 shows that BitTorrent simulation has a higher complexity. This is a result of the complex peering (tit-for-tat) and block selection (rarest-first) policies used. Figure 5.1 and Figure 5.2 present the time required to simulate data dissemination with Bullet and BitTorrent.

To evaluate the simulator performance in real settings we generated a set of Waxman topologies using BRITE [64] with different number of nodes and simulated the distribution of 1 GB files with different number/size of blocks. All simulations were executed on a system with an Intel P4@2.8 GHz processor and 1GB of memory.

The scale of our experiments is limited by the scale of BitTorrent and Bullet simulations. The other simulated protocols are much faster: simulating one of these solutions on topologies with few thousands end-nodes can be obtained in a few minutes. For BitTorrent and Bullet the simulator can simulate the distribution of a file split into a few thousands of blocks to few hundreds of nodes (a typical setting in today's scientific collaboration systems) in few hours.

Although we have not focused on performance (e.g., we have used Python for the simulator implementation), our simulator performs well compared to others described in

literature. For instance, Bharambe *et al.* [15] present results for simulating a network with 300 simultaneously active nodes and 100 MB file of 400 blocks, without incorporating physical topologies and consequently not simulating network contention. Similarly, Gkantsidis *et al.* [65] present a simulation results for a topology of 200 nodes and a file split in 100 blocks. They use a simplified network topology model with infinite core capacity and bandwidth constraints only on access links. Further their simulations are simplified by using overlay topologies that are computed offline.

6. Simulation Results

This chapter presents the results of our comprehensive simulation study. We detail our experimental setup in Section 6.1 and present simulation results that compare, along multiple success metrics, the techniques we study. We present the data dissemination time in Section 6.2, protocol overhead in Section 6.3, load balancing characteristics in Section 6.4, and fairness to competing traffic in Section 6.5. Section 6.6 presents an experimental validation of our choices of protocols parameters for Bullet and BitTorrent.

6.1. Experimental Setup

We use the physical network topologies of three real-world grid testbeds LCG [18], EGEE [19] [66] and GridPP [20]. The LCG topology incorporates 121 sites connected through up to 10Gbps core links. EGEE and GridPP are smaller and have similar characteristics to LCG in terms of network core bandwidth and access link to core link bandwidth ratio.

Additionally, to increase the confidence in our results, we generated two other sets of Waxman topologies using BRITE [64]. These two sets have the same number of intermediate and end-nodes and constant overall bandwidth, but they differ in the density of network links in the core. Comparing results on these two sets of topologies gives a more direct measure of the degree to which various proposed protocols are effective in exploiting network path diversity. The third generated set has a higher number of intermediate and end-nodes (around 500 nodes total). As these additional experiments mainly validate our simulation results, we do not provide their detailed description in here.

All simulations explore the performance of distributing a 1GB file over the different topologies. Bullet and BitTorrent are configured to work with two and four peers, respectively. We chose these configurations as they offer optimal performance in the topologies we modeled in our experiments (details about how we reached this conclusion are presented in Section 6.6).

6.2. Performance: File Transfer Time

As discussed in Section 3.3, depending on the application context, the performance focus can be on minimizing the average, median, N^{th} percentile, or the highest transfer time to destination. To cover all these performance criteria, for each data dissemination technique we present the evolution, in time, for the number of destinations that have completed the file transfer.

Figure 6.1, Figure 6.2, and Figure 6.3 present this evolution for the original LCG, EGEE, and GridPP topologies, respectively. Despite the different experimental results for these topologies, the following observations are common:

- IP-multicast and Logistical Multicast are the best solutions to deliver a file to the slowest node as they optimally exploit the bandwidth on bottleneck links. Spider is not presented here because it does not build more than one dissemination tree and thus it is, for these three topologies, equivalent to IP-multicast.
- Intermediate progress with IP-multicast. The explanation is that IP-multicast does not include buffering at intermediate points in the network and limits its data distribution rate to the rate of the bottleneck link.
- Logistical Multicast is among the first to complete the file dissemination process and also offers one of the best intermediate progress performance. This is a result of ability to store

data at intermediate routers as well as a result of the bandwidth distribution in these topologies: the bottlenecks are the site access links and not the links at the core of the network. As a result, Logistical Multicast is able to push the file fast through the core routers that border the final access link and thus offer near optimal intermediate distribution times.

- Application-level multicast (ALM), Bullet and BitTorrent perform worse but comparable to Logistical Multicast both in terms of finishing time as well as intermediate progress. They are able to exploit the plentiful bandwidth at the core and their performance is limited only by the access link capacity of various destination nodes.
- As expected, the naïve technique of distributing the file through independent streams to each destination does not offer any performance advantage. Surprisingly, however, on these over-provisioned networks, its performance is competitive with that of other methods.

The surprisingly good performance of parallel independent transfers in these topologies clearly indicates that the network core is over-provisioned. Even with all nodes pairing up and exchanging data at the full speed of their access links, core links are far from being fully used.

We are interested in exploring the performance of data dissemination techniques at various core-to-access link capacity ratios for the following two reasons. First, if the core is over-provisioned, we would like to understand how much bandwidth (and eventually cost) can be saved by reducing the core capacity without significantly altering the data dissemination performance. Second, we aim to understand whether independent transfers perform similarly well when compared to the more sophisticated techniques under different

network conditions. Stated otherwise, *we aim to quantify the performance gains (in terms of dissemination time) that complex data dissemination techniques offer when operating on less-endowed infrastructures.*

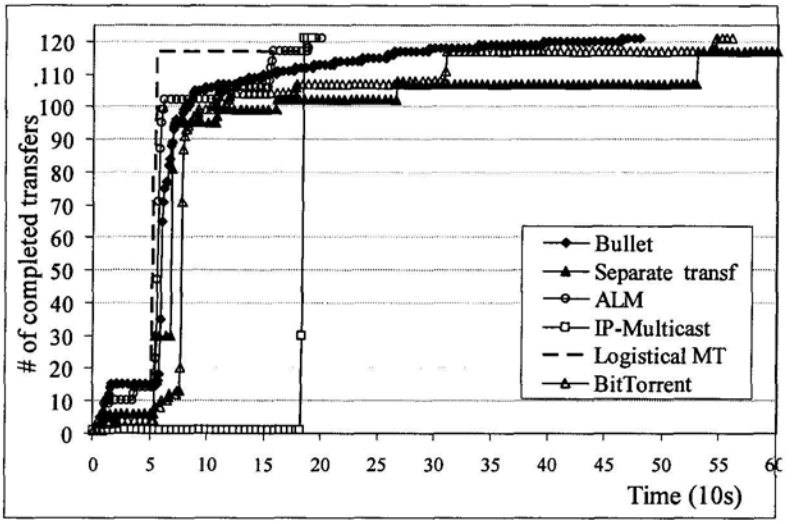

Figure 6.1. Number of destinations that have completed the file transfer for the original LCG topology (Separate transfer technique finishes the transfer at 730s, not presented in the plot for better readability).

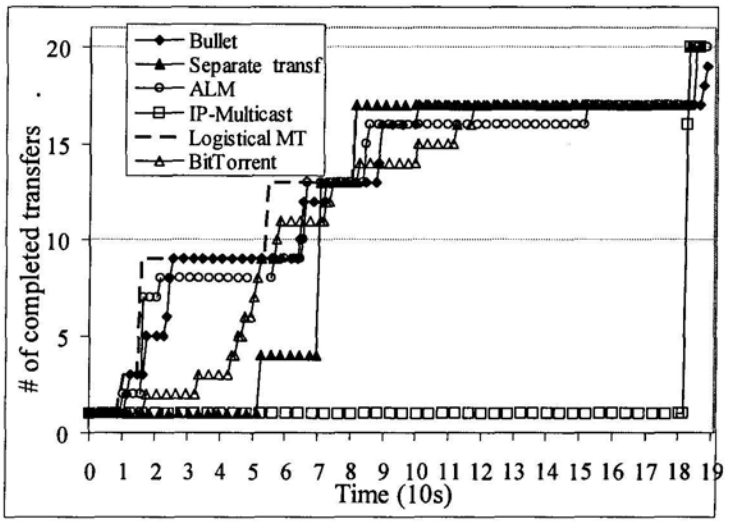

Figure 6.2. Number of destinations that have completed the file transfer for the original EGEE topology.

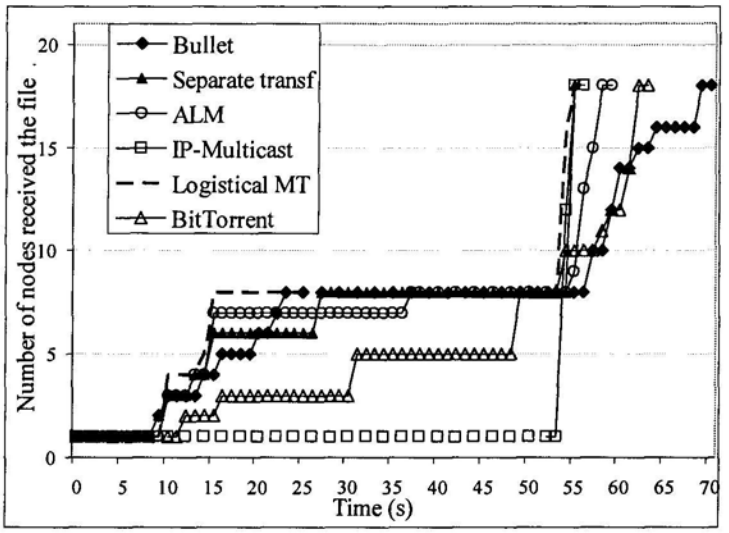

Figure 6.3. Number of destinations that have completed the file transfer for the original GridPP topology.

With these two goals in mind we ran the same simulations on a set of hypothetical topologies. These topologies are similar to the original LCG, EGEE and GridPP topologies except that the bandwidth of the core links (the links between the core routers) is 1/2, 1/4, 1/8, 1/16 or 1/32 of the original core link bandwidth.

Figure 6.4, Figure 6.5, and Figure 6.6, present the time to complete the transfer to 90% of the nodes (as well as to 50% and all nodes using the error bars) for the LCG, EGEE, and GridPP topologies, respectively, with different core link bandwidth. We summarize our observations below.

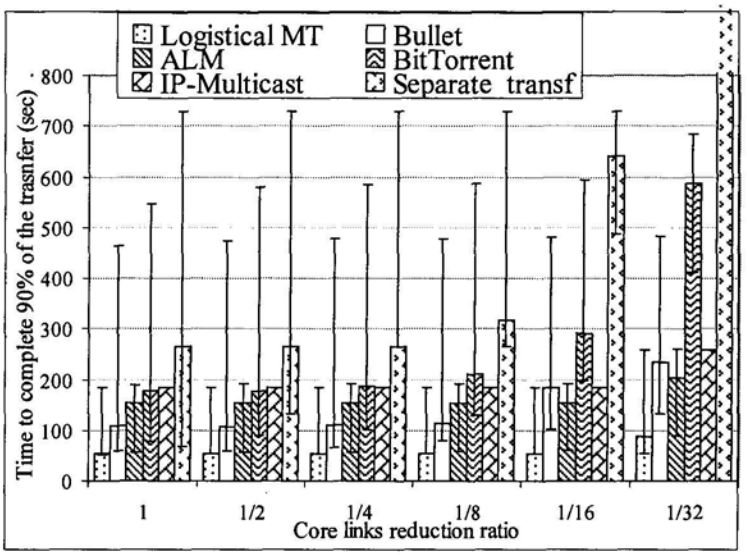

Figure 6.4. Time to finish the transfer to 90% of the nodes for the original LCG topology and the topology with reduced core bandwidth. The lower error bar indicates the time to complete the transfer for 50% of the nodes while the top error bar indicates the time to complete the transfer for the last node. Separate transfer's technique finishes in 2280 seconds on the topology with core bandwidth reduced to 1/32 (not presented here for clarity).

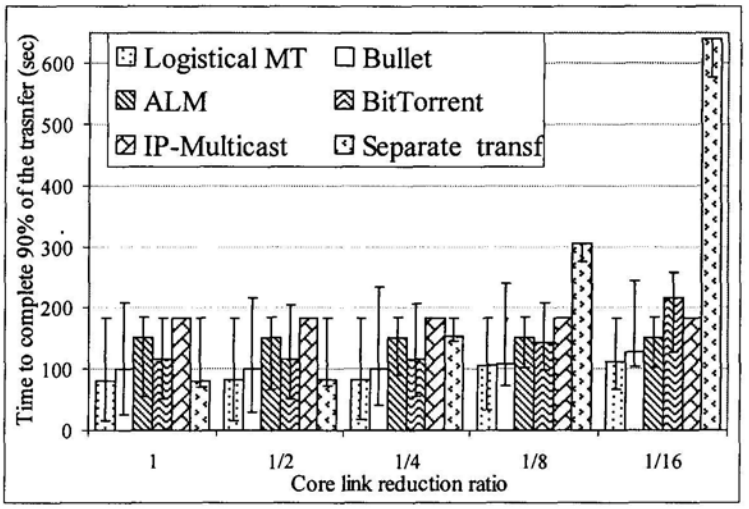

Figure 6.5. Time to finish the transfer to 50%, 90%, and all nodes for the EGEE topology – original and reduced core bandwidth.

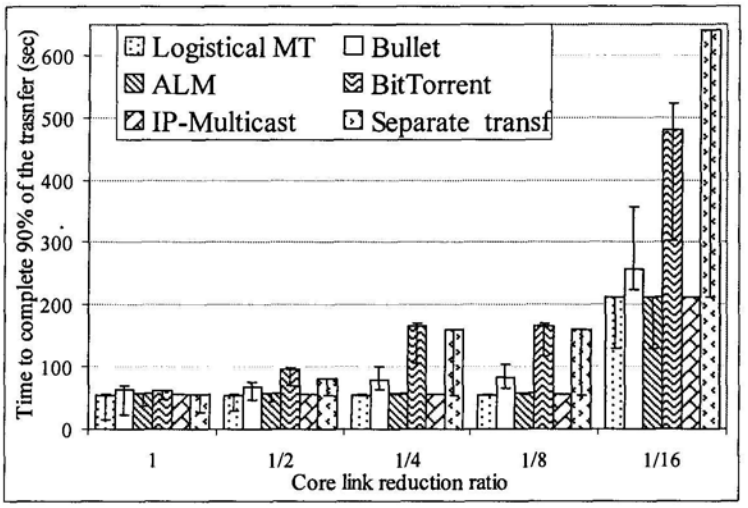

Figure 6.6. Time to finish the transfer to 50%, 90%, and all nodes for the GridPP topology – original and reduced core bandwidth.

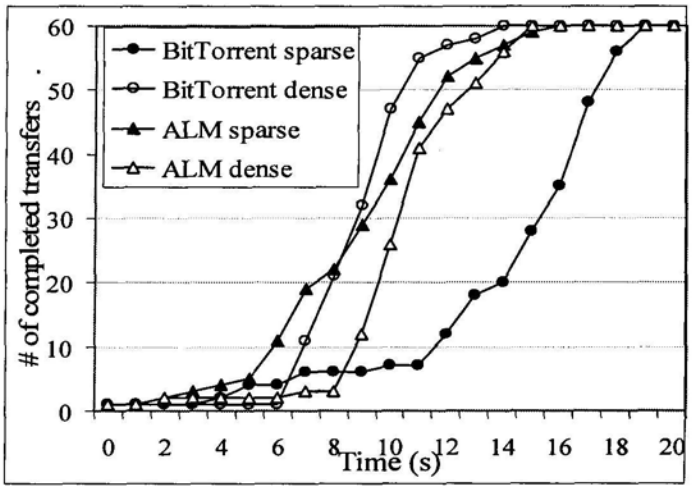

Figure 6.7. Number of destinations that have completed the file transfer with two generated topologies. The dense topology has four times more links in the core with four times less average bandwidth per link.

Most importantly, we observe that the performance of the parallel independent-transfers technique degrades much faster than the performance of any other technique when the bandwidth in the core decreases. Additionally, the performance of the more sophisticated dissemination schemes does not degrade significantly when reducing the core capacity. This

is testament to their ability to exploit orthogonal bandwidth. Furthermore, it is an indication that similar performance can be obtained at lower network core budgets by employing sophisticated data distribution techniques.

In addition, based on results obtained with reduced core capacity, we observe the following. First, ALM and Logistical Multicast offer good intermediate progress, while their completion time, limited by a bottleneck link, is similar to simple IP-multicast. Second, in addition to offering good intermediate progress, Bullet and BitTorrent offer good completion time by exploiting orthogonal bandwidth.

To further investigate the ability to exploit alternate network paths, we have generate two sets of topologies in which the aggregate core bandwidth is maintained constant but the number of core links is changed. Figure 6.7 compares the intermediate progress of the BitTorrent and ALM protocols on these two topologies: the 'dense' topology has four times more links in the core (and four times lower average core link bandwidth). As shown in Figure 6.7, BitTorrent performance is better with more links in the core while ALM performance slightly degrades. These results underline BitTorrent ability to exploit all available transport capacity. Bullet shows similar behavior.

Summary. Three key conclusions can be derived from the above simulation results:

- In the real Grid deployments analyzed, networks appear to be over-provisioned and, in these conditions, even naïve algorithms perform well.
- The group of application-level schemes such as Bullet, BitTorrent and, ALM are initially within the same ballpark compared to others. Because Bullet and BitTorrent generate high overheads (discussed in the next section), ALM performance starts to dominate for more constrained cores. We note, however, that Bullet and BitTorrent have other additional

intrinsic properties (e.g., tolerance to node failures) that make them attractive in different scenarios (e.g., high churn conditions specific to peer-to-peer systems).

- Bullet and BitTorrent are more efficient in exploiting the orthogonal bandwidth available between the participating nodes, being thus more capable to cope with different topologies and adapt to dynamically changing workloads.

6.3. Overheads: Network Effort

A second important direction to compare data dissemination solutions is evaluating the overhead they generate.

The traditionally used method to compare overheads for tree-based multicast solutions is to compare maximum link stress (or link stress distributions); where link stress is defined as the number of identical logical flows that traverse the link. However, this metric is irrelevant for Bullet or BitTorrent as these protocols dynamically adjust their distribution patterns and, therefore, link stress varies continuously during data dissemination.

For this reason, we propose a new metric to estimate overheads. We estimate the volume of duplicate traffic that traverses each physical link and aggregate it over all links in the testbed. While individual values of this metric are not relevant in themselves, they offer interesting insights when comparing distinct protocols.

Figure 6.8 shows the generated traffic (labeled as useful or overhead) for each protocol for the original LCG. We define as *useful* the data traffic that remains after excluding all link-level packet duplicates. Note that the volume of useful traffic differs between the protocols since different schemes map differently on the physical topology.

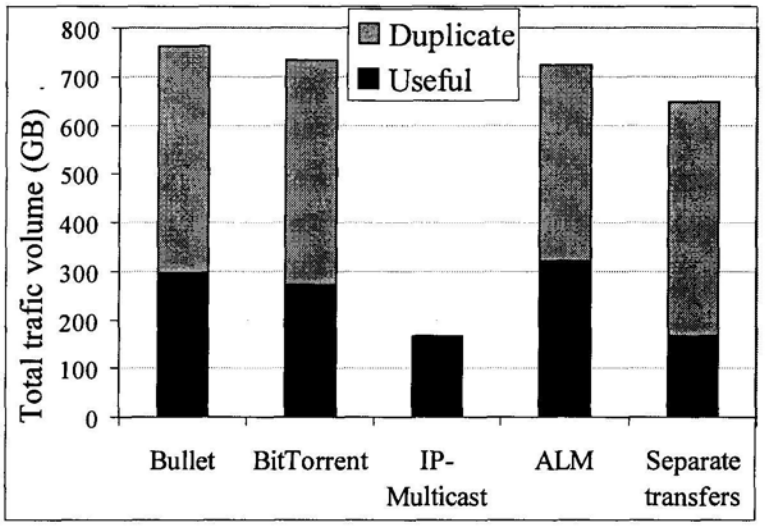

Figure 6.8. Overhead for each protocol on the LCG topology.

The following observations can be made based on Figure 6.8 (and can be generalized, as there is little variance across various topologies). First, as expected, IP-layer solutions do not generate any duplicates and thus are optimal in terms of total traffic.

Second, Bullet, BitTorrent and ALM require significantly higher network effort even without considering the duplicates. This is the result of node pairing relationships in these schemes that pay little consideration to the nodes location in the physical network topology.

In considering duplicate traffic, Bullet emerges as the largest bandwidth consumer. This is because Bullet uses approximate representations of the set of blocks available at each node and the upload decision is made at the sender node depending on the receiver content summary. False negatives on the approximate data representations thus generate additional traffic overhead. BitTorrent generates slightly smaller overheads as nodes employ exact representations (bitmaps) to represent the set of blocks available locally.

ALM trees also introduce considerable overhead as the tree construction algorithm is optimized for high-bandwidth dissemination and ignores nodes' location in the physical topology.

Finally, one observation applies equally to all application level techniques studied: the overhead grows with the size of the topology. For example, while for the EEGE topology the *ratio* of duplicate traffic is between 43% (for BitTorrent) and 66% (for separate transfers) it grows to 55% (for ALM) and 74% (for separate transfers) for the LCG topology.

Summary. Application-level data dissemination solutions generate significant overheads (the generated traffic volume is up to four times larger than that generated by optimal IP-level solutions). The reason is that application-level techniques base their dissemination decisions on application level metrics rather than on node topology location. Consequently, traffic often does not use optimal network paths and the same block of data travels multiple times the same physical link or is sent multiple times through the core through different network paths.

6.4. Load Balance

Another metric to evaluate the performance of data dissemination schemes is load balance. To this end, we estimate the volume of data processed (both received and sent) at each end-node. Obviously, network-layer techniques (e.g., IP-multicast, SPIDER, logistical multicast) that duplicate packets at routers or storage points inside the network, will offer ideal load balance. Using these, each end-node will optimally receive/send a minimal amount of data.

At the other end of the spectrum, sending data through independent connections directly from the source will offer the worst load balance as the source load is directly proportional to the number of destinations.

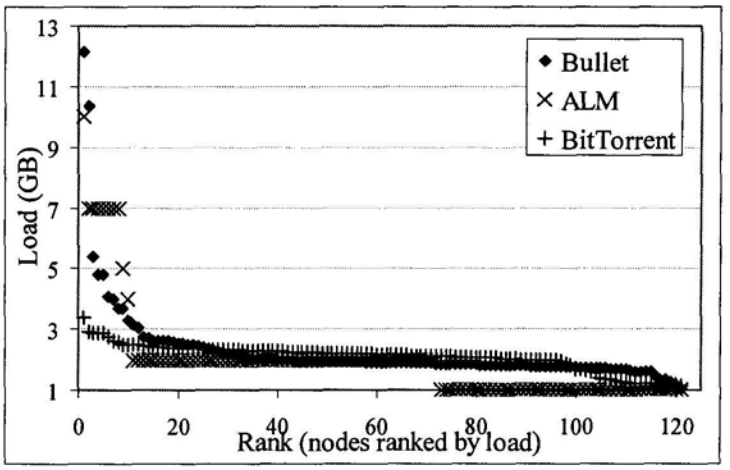

Figure 6.9. Load balancing for ALM, BitTorrent and Bullet. Nodes are ranked in decreasing order of their load (LCG topology).

Figure 6.9 presents the load balancing performance of the remaining techniques: ALM, BitTorrent, and Bullet for the LCG topology. For the other topologies the relative order of these techniques in terms of load balance does not change, thus we do not present these results here.

ALM has the worst load balance among the three solutions as it tends to increase the load on the nodes with ample access-link bandwidth. Of the remaining two, BitTorrent offers slightly better load balancing than Bullet due to its tit-for-tat mechanism that implicitly aims to evenly spread data dissemination efforts.

Summary. Application-level solutions offer better load balance than the naive solution of sending the data through separate channels from the source to each destination. Additionally, BitTorrent offers the best load balance among application-level solutions.

6.5. Fairness to Competing Traffic

While all the application layer protocols we analyze use TCP or a TCP-friendly congestion control scheme for data exchanges between each individual pair of nodes, they differ in their impact on the network and on the competing traffic. The reason is that some of these dissemination schemes generate a large number of network flows, sometimes mapped randomly over the network topology, thus having the potential to intensely stress bottleneck links and, consequently, impact the network flows generated by other applications. We are not aware of any related work analyzing this impact, and implicitly the fairness of data dissemination techniques.

Our evaluation of fairness is complicated by the fact that, unlike for unicast traffic, for single-source-multiple-destinations traffic there is no commonly accepted fairness definition. Even for IP-multicast, although fairness has been studied for many years [67], there is still no general consensus on what should be the relative fairness between multicast and unicast traffic.

In general, with multicast traffic, multiple bandwidth allocation policies are possible. For example, on one side, at the individual physical link level, a multicast session might deserve more bandwidth than a TCP connection as it serves multiple receivers. On the other side, however, it is also reasonable to argue that a multicast session should not be given more bandwidth than individual TCP connections, in order not to penalize competing TCP connections that share a portion of the path with the multicast session.

Different data dissemination solutions that work at the application-layer do have different impact on competing traffic. For example, at one end of the spectrum, a logistical multicast scheme with intermediary storage nodes placed close to network routers will be

similar in impact to IP-multicast. At the other end of the spectrum, solutions that create a distribution tree for each participating node have the highest impact on competing traffic. For instance, in FastReplica [68], the source node divides the file into *n* equal blocks (where *n* equals the number of participating nodes) and sends each block to one of the participating nodes. After receiving the first block from the source, each node opens *n-1* separate channels and sends the block to every other participating node. This solution creates a number *(n-1)*(n-2)* channels simultaneously and is clearly unfair to competing traffic.

From the possible set of metrics to estimate the impact of competing traffic we choose link stress distribution. The higher the number of flows a data dissemination scheme maps on a physical link, the higher its impact on competing traffic. This impact is non-negligible, as Figure 6.10 presenting the *average* link stress distribution for the LGC topology shows. In fact, the *maximum* link stress generated by Bullet and BitTorrent can be significantly higher; as high as 70 on the LCG topology, as Figure 6.11 shows. This implies that if a unicast transfer shares its bottleneck link with a link on which Bullet or BitTorrent generates such stress, its allocated bandwidth is drastically reduced.

ALM tends to stress more the links around the nodes with high access link bandwidth, since these nodes are favored to have many children nodes in the bandwidth optimized trees. Similarly, the technique of sending the file from the source to each destination through independent channels is the worst in terms of fairness to competing traffic since the generated stress on the access link of the source is proportional with the number of destinations

This is not a behavior particular to grids, of course. In fact, the generous network provisioning in the topologies we analyze masks the problems raised by the lack of fairness. However, we believe that presenting fairness metrics is germane to our evaluation.

Summary. While IP multicast offers ideal fairness, application-level solutions have a high impact on competing traffic. Overall BitTorrent and Bullet are fairer than ALM, since our ALM tree construction favors the nodes with high bandwidth leading to more connections around these nodes.

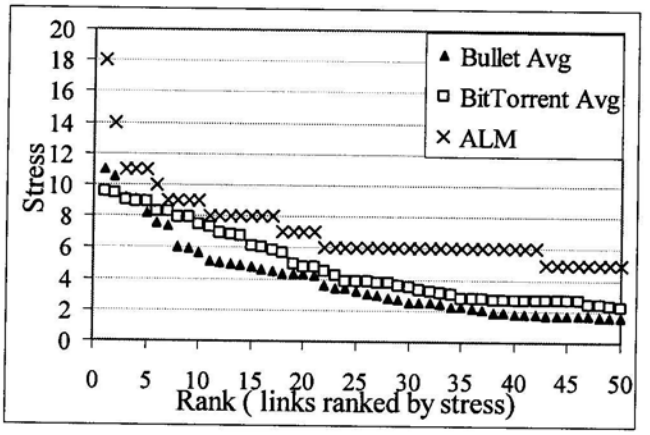

Figure 6.10. Average link stress distribution of BitTorrent and Bullet over the LCG topology. The plot presents average link stress for the most stressed 50 links.

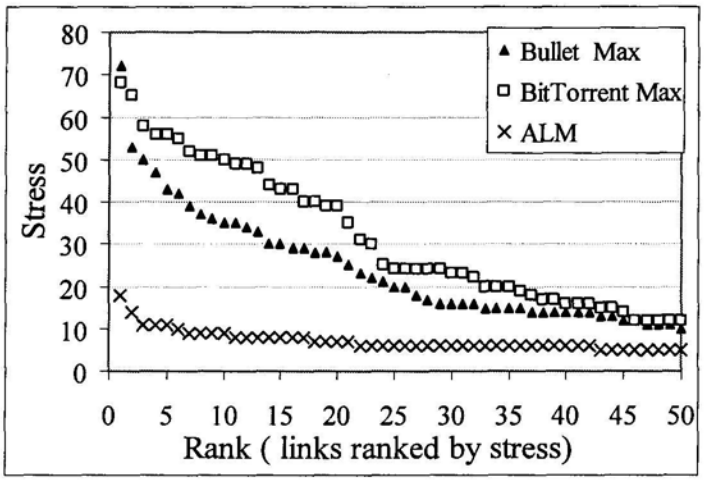

Figure 6.11. Maximum link stress distribution of BitTorrent and Bullet over the LCG topology. The plot presents maximum link stress for the most stressed 50 links.

6.6. *The Effect of the Number of Peers in Bullet and BitTorrent*

The number of 'peering relationships' (i.e., the number of nodes each node exchanges data with) is a configuration parameter specific for Bullet and BitTorrent. To understand and make sure we configure these two protocols optimally for our environments we have experimented with different configurations for the number of peering relationships.

We find that, for our topologies, configuring Bullet with two peers and BitTorrent with four peers in EEGE and GridPP topologies (incidentally the default configurations for BitTorrent deployments) and eight peers in LCG topology, provides the fastest data dissemination. We note, however, that the differences in dissemination speed among various configurations are minor compared to the differences among dissemination schemes.

The generated traffic volume remains constant for BitTorrent, while it linearly, and non-trivially, increases with the number of peers for Bullet. This is a consequence of exchanging probabilistic summaries in Bullet as opposed to accurate, though slightly larger, summaries in BitTorrent. We estimate that, for large files, the additional control overhead

required to provide accurate summaries in Bullet will be entirely compensated by lower duplicate traffic.

In terms of load balancing, the Bullet configuration with two peers provides the best load balancing, while a larger than the default four peers (as in our experiments) would improve load balancing for BitTorrent.

In terms of fairness to competing traffic, increasing the number of peers generally results in reduced fairness on links around nodes with high access bandwidth, since these nodes get data sooner in the replication process and serve more collaborating peers.

Summary. For our topologies, configuring Bullet with two peers and BitTorrent with four peers in EEGE and GridPP topologies, and eight peers in LCG topology, provides the fastest data dissemination. Increasing the number of peers provides better load balancing without generating additional overheads. Additionally, increasing the number of peers in Bullet and BitTorrent reduces fairness around the nodes with high access bandwidth, as these nodes obtain the complete file first in the data dissemination process and continue to serve a large number of nodes.

7. Summary

This study focuses on the problem of disseminating large data from one source to multiple destinations in the context of today's science grids. Data dissemination in these environments is characterized by relatively small collaborations (tens to hundreds of participating sites), large data files to transfer, well-provisioned networks, and collaborative participants.

The objective of this study was to provide an experimentally-supported answer to the question: Given the characteristics of deployed grids, what benefits can peer-to-peer solutions offer for one-to-many data dissemination?

Our simulation-based experimental investigation on seven solutions selected from a set of successful Internet data delivery and peer-to-peer deployed systems shows the following:

- Some of today's Grid testbeds are over-provisioned. In this case, the deployment is scalable with the size of the user community, and peer-to-peer solutions that adapt to dynamic and under-provisioned networks do not bring significant benefits. While they improve load balancing, they add significant overheads and, more importantly, do not offer significant improvements in terms of distribution time.
- Application-level schemes such as BitTorrent, Bullet and application-level multicast perform best in terms of dissemination time. However, they introduce high-traffic overheads, even higher than independent parallel transfers. On the other hand, BitTorrent and Bullet are designed to deal with dynamic environment conditions, a feature which might be desirable in some scenarios.
- The naive solution of separate data transfers from source to each destination yields

reasonable performance on well-provisioned networks but its performance drops dramatically when the available bandwidth decreases. In such cases, adaptive peer-to-peer like techniques that are able to exploit multiple paths existing in the physical topology can offer good performance on a network that is less well provisioned.

To summarize, the peer-to-peer solutions that offer load balancing, adaptive data dissemination, and participation incentives lead to unjustified costs in today's scientific data collaborations deployed on over-provisioned network cores. However, as user communities grow and these deployments scale (as already seen in the Open Science Grid [69], for example) peer-to-peer data delivery mechanisms will outperform other techniques.

In any case, network provisioning has to progress hand-in-hand with improvements and the adoption of intelligent, adaptive data dissemination techniques. In conjunction with efficient data distribution techniques, appropriate network provisioning will not only save costs while building/provisioning collaborations, but also derive optimal performance from deployed networks.

References

1. LHC, *The Large Hardon Collider, http://lhc-new-homepage.web.cern.ch/lhc-new-homepage/.* 2002.
2. *The Spallation Neutron Source*. 2006.
3. *TeraShake http://epicenter.usc.edu/cmeportal/TeraShake.html*. 2007.
4. *CyberShake http://epicenter.usc.edu/cmeportal/CyberShake.html*. 2007.
5. Catlett, C., *The TeraGrid: A Primer*. 2002, www.teragrid.org.
6. Brown, M., *Blueprint for the Future of High-Performance Networking.* Communications of the ACM (CACM), 2003. **46**(11): p. 30-77.
7. Allcock, W., A. Chervenak, I. Foster, C. Kesselman, and S. Tuecke. *Protocols and Services for Distributed Data-Intensive Science*. in *Advanced Computing and Analysis Techniques in Physics Research (ACAT)*. 2000: AIP Conference Proceedings.
8. Bassi, A., M. Beck, T. Moore, J.S. Plank, M. Swany, R. Wolski, and G. Fagg, *The Internet Backplane Protocol: A Study in Resource Sharing.* Future Generation Computing Systems, 2003. **19**(4): p. 551-561.
9. Terekhov, I., R. Pordes, V. White, L. Lueking, L. Carpenter, H. Schellmant, J. Trumbo, S. Veseli, M. Vranicar, and S. White. *Distributed data access and resource management in the D0 SAM system*. in *IEEE International Symposium on High Performance Distributed Computing*. 2001.
10. Wang, F., Q. Xin, B. Hong, S.A. Brandt, E.L. Miller, D.D.E. Long, and T.T. Mclarty. *File System Workload Analysis For Large Scientific Computing Applications*. in *NASA/IEEE Conference on Mass Storage Systems and Technologies (MSST 2004)*. 2004.
11. Iamnitchi, A., M. Ripeanu, and I. Foster. *Small-World File-Sharing Communities*. in *Infocom 2004*. 2004. Hong Knog.

12. Cohen, B., *BitTorrent web site: http://www.bittorrent.com*. 2005.
13. Kostic, D., A. Rodriguez, J. Albrecht, and A. Vahdat. *Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh*. in *SOSP'03*. 2003. Lake George, NY.
14. Guo, L., S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. *Measurements, Analysis and modeling of BitTorrent-like systems*. in *ACM SIGCOMM Internet Measurement Conference*. 2005. New Orleans, LA.
15. Bharambe, A.R., C. Herley, and V.N. Padmanabhan. *Analysing and improving a BitTorrent network's performance mechanisms*. in *The 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2006)*. 2006. Barcelona, Spain.
16. Plaza, A., D. Valencia, J. Plaza, and P. Martinez, *Commodity cluster-based parallel processing of hyperspectral imagery.* Journal of Parallel and Distributed Computing, 2006. **66**(3): p. 345–358.
17. Ellsworth, D., C. Henze, B. Green, P. Moran, and T. Sandstrom, *Concurrent Visualization in a Production Supercomputer Environment.* EEE Transactions on Visualization and Computer Graphics, 2006. **12**(5): p. 997-1004.
18. Doyle, A.T. and C. Nicholson. *Grid Data Management: Simulations of LCG 2008*. in *Computing in High Energy and Nuclear Physics, CHEP '06*. 2006. Mumbai, India.
19. *Enabling Grids for E-sciencE Project*. 2006.
20. Britton, D., A.J. Cass, P.E.L. Clarke, J.C. Coles, A.T. Doyle, N.I. Geddes, J.C. Gordon, R.W.L. Jones, D.P. Kelsey, S.L. Lloyd, R.P. Middleton, S.E. Pearce, and D.R. Tovey. *GridPP: Meeting the Particle Physics Computing Challenge*. in *UK e-Science All Hands Conference*. 2005.
21. Kaplan, A., G.C. Fox, and G.v. Laszewski. *GridTorrent Framework: A High-performance Data Transfer and Data Sharing Framework for Scientific Computing*. in *Workshop on Grid Computing Portals and Science Gateways - Supercomputing '07*. 2007. Reno Nevada.

22. Briquet, C., X. Dalem, S. Jodogne, and P.-A.d. Marneffe. *Scheduling data-intensive bags of tasks in P2P grids with bittorrent-enabled data distribution.* in *workshop on Use of P2P, GRID and agents for the development of content networks - HPDC '07.* 2007.

23. *Chyouhwa Chen webpage - National Taiwan University of Science and Technology* http://cchen1.csie.ntust.edu.tw/students/master2007.htm. 2007.

24. Iamnitchi, A., S. Doraimani, and G. Garzoglio. *Filecules in High-Energy Physics: Characteristics and Impact on Resource Management.* in *HPDC 2006.* 2006. France.

25. Gummadi, K.P., R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, and J. Zahorjan. *Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload.* in *SOSP'03.* 2003. Lake George, NY.

26. Bellissimo, A., P. Shenoy, and B.N. Levine, *Exploring the Use of BitTorrent as the Basis for a Large Trace Repository*, University of Massachuttes-Amherst.

27. Cui, Y., R. Moore, K. Olsen, A. Chourasia, P. Maechling, B. Minster, S. Day, Y. Hu, J. Zhu, A. Majumdar, and T. Jordan. *Enabling Very-Large Scale Earthquake Simulations on Parallel Machines.* in *International Conference on Computational Science.* 2007.

28. Chu, Y.-h., S.G. Rao, S. Seshan, and H. Zhang, *A Case for End System Multicast.* IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast, 2002. **20**(8).

29. Diot, C., B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, *Deployment Issues for the IP Multicast Service and Architecture.* IEEE Network: Special Issue on Multicasting, 2000. **14**(1).

30. Touch, J.D., *Overlay Networks.* Computer Networks, 2001. **36**(2001): p. 115-116.

31. Wolski, R., *Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service*, in *Proc. 6th IEEE Symp. on High Performance Distributed Computing.* 1997: Portland, Oregon.

32. Vazhkudai, S., J. Schopf, and I. Foster. *Predicting the Performance of Wide-Area Data Transfers*. in *16th International Parallel and Distributed Processing Symposium (IPDPS 2002)*. 2002. Fort Lauderdale, FL.

33. Vazhkudai, S., S. Tuecke, and I. Foster. *Replica Selection in the Globus Data Grid*. in *IEEE International Conference on Cluster Computing and the Grid (CCGRID2001)*. 2001. Brisbane, Australia.

34. Beck, M., T. Moore, J.S. Plank, and M. Swany. *Logistical Networking: Sharing More Than the Wires*. in *Active Middleware Services Workshop*. 2000. Norwell, MA.

35. Plank, J.S., A. Bassi, M. Beck, T. Moore, D.M. Swany, and R. Wolski, *Managing Data Storage in the Network*. IEEE Internet Computing, 2001. **5**(5): p. 50–58.

36. Zurawski, J., M. Swany, M. Beck, and Y. Ding. *Logistical Multicast for Data Distribution*. in *Workshop on Grids and Advanced Networks*. 2005. Cardiff, UK.

37. Ganguly, S., A. Saxena, S. Bhatnagar, S. Banerjee, and R. Izmailov. *Fast Replication in Content Distribution Overlays*. in *IEEE INFOCOM*. 2005. Miami, FL.

38. Byers, J.W., M. Luby, M. Mitzenmacher, and A. Rege. *A Digital Fountain Approach to Reliable Distribution of Bulk Data*. in *SIGCOM*. 1998.

39. Byers, J., J. Considine, M. Mitzenmacher, and S. Rost. *Informed Content Delivery Across Adaptive Overlay Networks*. in *SIGCOMM2002*. 2002. Pittsburg, PA.

40. Plank, J.S., M. Beck, W. Elwasif, T. Moore, M. Swany, and R. Wolski. *The Internet Backplane Protocol: Storage in the Network*. in *NetStore99: The Network Storage Symposium*. 1999. Seattle, WA.

41. Castro, R., M. Coates, M. Gadhiok, R. King, R. Nowak, E. Rombokas, and Y. Tsang, *Maximum likelihood network topology identification from edge-based unicast measurements*. ACM SIGMETRICS, 2002.

42. Chun, B., D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, *PlanetLab: An Overlay Testbed for Broad-Coverage Services.* ACM Computer Communications Review, 2003. **33**(3).

43. *Planetlab Iperf Data.* http://jabber.services.planet-lab.org/php/iperf. 2007.

44. Pendarakis, D., S. Shi, D. Verma, and M. Waldvogel. *ALMI: An Application Level Multicast Infrastructure*. in *USITS'01*. 2001.

45. Jannotti, J., D.K. Gifford, K.L. Johnson, M.F. Kaashoek, and J.W. O'Toole. *Overcast: Reliable Multicasting with an Overlay Network.* in *4th Symposium on Operating Systems Design and Implementation (OSDI 2000)*. 2000. San Diego, California.

46. Banerjee, S., C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, *OMNI: an efficient overlay multicast infrastructure for real-time applications.* Computer Networks: The International Journal of Computer and Telecommunications Networking, 2006. **50**(6).

47. Ratnasamy, S., M. Handley, R.M. Karp, and S. Shenker. *Application-Level Multicast Using Content-Addressable Networks*. in *Third International COST264 Workshop on Networked Group Communication*. 2001.

48. Castro, M., P. Druschel, A.-M. Kermarrec, and A. Rowstron, *Scribe: A large-scale and decentralized application-level multicast infrastructure.* IEEE Journal on Selected Areas in Communication (JSAC), 2002. **20**(8).

49. Castro, M., P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. *SplitStream: High-Bandwidth Multicast in Cooperative Environments*. in *SOSP'03*. 2003. Lake George, NY.

50. Banerjee, S., B. Bhattacharjee, and C. Kommareddy. *Scalable Application Layer Multicast*. in *SIGCOMM2002*. 2002. Pittsburgh, PA.

51. Das, S., A. Nandan, M.G. Parker, G. Pau, and M. Gerla. *Grido An Architecture for a Grid-based Overlay Network*. in *International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine 2005)*. 2005. FL, USA.

52. Burger, M.d. and T. Kielmann. *MOB: zero-configuration high-throughput multicasting for grid applications*. in *16th international symposium on High performance distributed computing (HPDC)*. 2007. California, USA.

53. Izmailov, R. and S. Ganguly. *Fast Parallel File Replication in Data Grid*. in *Future of Grid Data Environments workshop, GGF - 10*. 2004. Berlin, Germany.

54. Allen, M.S. and R. Wolski. *The Livny and Plank-Beck Problems: Studies in Data Movement on the Computational Grid*. in *SuperComputing 2003 (SC2003)*. 2003. Phoenix, Arizona.

55. Biersack, E.W., P. Rodriguez, and P. Felber. *Performance analysis of peer-to-peer networks for file distribution*. in *QofIS'04, Fifth International Workshop on Quality of Future Internet Services*. 2004.

56. Qiu, D. and R. Srikant. *Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks*. in *ACM SIGCOMM*. 2004. Portland, OR, USA.

57. Pouwelse, J.A., P. Garbacki, D.H.J. Epema, and H.J. Sips. *The Bittorrent P2P File-Sharing System: Measurements and Analysis*. in *Int'l Workshop on Peer-to-Peer Systems (IPTPS)*. 2005.

58. Izal, M., G. Urvoy-Keller, E.W. Biersack, P. Felber, A.A. Hamra, and L. Garc´es-Erice. *Dissecting BitTorrent: Five Months in a Torrent's Lifetime*. in *PAM*. 2004.

59. *The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/*. 2006.

60. Vahdat, A., K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. *Scalability and Accuracy in a Large-Scale Network Emulator*. in *OSDI*. 2002.

61. White, B., J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. *An Integrated Experimental Environment for Distributed Systems and Networks*. in *OSDI*. 2002. Boston, MA.

62. Huang, P., D. Estrin, and J. Heidemann. *Enabling Large-scale simulations: selective abstraction approach to the study of multicast protocols*. in *Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. 1998. Montreal, Canada.

63. *Azureus*, in http://azureus.sourceforge.net/. 2007.

64. Medina, A., A. Lakhina, I. Matta, and J. Byers. *BRITE: An Approach to Universal Topology Generation.* in *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS '01.* 2001. Cincinnati, Ohio.

65. Gkantsidis, C. and P.R. Rodriguez. *Network coding for large scale content distribution.* in *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005).* 2005. Miami, FL.

66. Cameron, D.G., A.P. Millar, C. Nicholson, R. Carvajal-Schiaffino, K. Stockinger, and F. Zini, *Analysis of Scheduling and Replica Optimisation Strategies for Data Grids Using OptorSim.* Journal of Grid Computing, 2004. **2**(1): p. 57-69.

67. Yang, Y.R. and S.S. Lam. *Internet Multicast Congestion Control: A Survey.* in *ICT 2000.* 2000. Acapulco, Mexico.

68. Cherkasova, L. and J. Lee. *FastReplica : Efficient Large File Distribution within Content Delivery Networks.* in *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems.* 2003. Seattle, Washington.

69. *Open Science Grid,* http://www.opensciencegrid.org/. 2007.

Appendix A: ALM Tree Construction Algorithm

We designed an ALM tree construction algorithm using a global view of the nodes and the topology. The ALM construction algorithm presented in the below starts with a tree containing only the source node. Every cycle the algorithm selects the best child (the end node that is reachable with the highest bandwidth) and adds it to the tree.

In the main while loop, we find the best child (the one with the highest bandwidth path) for every node currently found in the tree through the procedure *FindBestChildNotInTheTree(n),* which finds the best child (if any) through building the shortest path tree rooted at n on the physical topology.

After finding all best children for all the nodes currently in the tree, BestNode procedure simply selects the one with the highest bandwidth. The new node is added to the tree and the loop continues until all nodes are added to the tree.

The complexity of simple implementation of this approach is $O(n^3)$. It can be optimized to run in $O(n(l+n)\log n)$, where n is the number of nodes in the system, and l is the number of links in the network.

```
Nodes ←{all end nodes}
Tree ← {sourceNode}
Nodes ← nodes - {sourceNode}
While nodes ≠ϕ
   Candidates ← {}
   For every node n ∈Tree:
      newCandidate=
         FindBestChildNotInTheTree (n)
      Candidates=Candidates ∪
         {newCandidate}
   EndFor
   newLeaf= BestNode(Candidates)
   Tree = Tree ∪ newLeaf
   Nodes ←nodes - {newLeaf}
EndWhile
```

Appendix B: SPIDER Tree Construction Algorithm

Instead of trying to maximize the bandwidth of the constructed tree, SPIDER tree construction algorithm tries to maximize the residual bandwidth after constructing the tree. The residual bandwidth facilitates building more trees and exploiting perpendicular bandwidth.

Figure B.1 lists the SPIDER tree construction algorithm. The algorithm starts by adding the source (say node 0) to the list of nodes that have been added to the current tree (*InTree)*. Next, for each node k in the *InTree* list, the algorithm finds the maximum outgoing bandwidth arc to nodes outside the *InTree* list. For each of node k already in the tree, the algorithm computes the leftover outgoing bandwidth E_k, if its maximum outgoing edge is added to the current tree. The edge which leaves the maximum outgoing bandwidth for its source node is added to the tree and its destination node is added to the *InTree* list. These steps are repeated until all nodes have been added to the tree. Then, the bandwidth on all edges of the tree is reduced to the amount of bottleneck bandwidth. The algorithm is executed on the remaining graph until no more trees are found.

Input:
N - Set of Nodes – S is the source
b_{ij} - Bandwidth between all pairs $i, j \in N$
E_n - Outgoing Access Capacity of $n \in N$

Output:
T - Set of Trees

Algorithm:
1. $T \leftarrow \phi$
2. Do
3. $CurrentTree \leftarrow \phi$
4. $InTree \leftarrow \{S\}$
5. While $InTree \neq N$ do
6. For each $n \in InTree$ do
7. $M_n \leftarrow$ Edge with $max_j\{b_{nj}\}$
8. $R_n \leftarrow E_n$ - {Bandwidth of M_n}
9. End For
10. $x \leftarrow$ Node with $max_j\{R_j\}$
11. $CurrentTree \leftarrow CurrentTree \cup M_x$
12. $InTree \leftarrow InTree \cup \{destination\ of\ M_x\}$
13. End While
14. For each edge e in $CurrentTree$
15. $b_e \leftarrow b_e$ - {bottleneck in $CurrentTree$}
16. End For
17. $T \leftarrow T \cup CurrentTree$
18. End Do
19. Return T

Figure B.1. SPIDER tree construction algorithm (source [37])