Are P2P Data-Dissemination Techniques Viable in Today's Data Intensive Scientific Collaborations?

Samer Al Kiswany[1], Matei Ripeanu [1], Adriana Iamnitchi[2] and Sudharshan Vazhkudai[3]

[1] Electrical and Computer Engineering Department
The University of British Columbia
{samera, matei}@ece.ubc.ca

[2] Computer Science and Engineering
University of South Florida
anda@cse.usf.edu

[3] Computer Science and Mathematics Division
Oak Ridge National Laboratory
vazhkudaiss@ornl.gov

Abstract

The avalanche of data from scientific instruments and the ensuing interest among a geographically distributed user base to analyze and interpret them has pushed forward the need to efficiently disseminate data. An optimal data distribution scheme will find the delicate—and often application-specific—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 then 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 real-world data-centric collaborations.

Our results show that 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 peer-to-peer data delivery mechanisms may outperform other techniques. However a careful analysis will be required to decide the data dissemination technique as various solutions differ significantly in their generated traffic overhead and impact on competing traffic.

1. Introduction

Efficient scientific data dissemination has never been more important. A growing number of instruments and observatories are generating petabytes (10^{15} bytes) of data that needs to be analyzed by large, geographically dispersed user communities. Examples include CERN's Large Hadron Collider (LHC) experiment [1], neutron scattering at the Spallation Neutron Source (SNS)[2], or the DØ experiment at Fermi National Accelerator Laboratory [3]. Aiding in the formation of these collaborative data federations are ever increasing network capabilities including high-speed optical interconnects (e.g., TeraGrid [4], LambdaGrid [5]) and optimized bulk transfer tools and protocols (e.g., GridFTP [6], IBP [7]).

Data dissemination in such federations involves the dynamic distribution of subsets of data available at one site to one or many target locations for real-time analysis and visualization. For instance, the petabytes of data from the LHC experiment at CERN (Tier 0) are required to be distributed world-wide, across national (Tier 1) and regional (Tier 2) centers. Similarly, there is a growing interest in disseminating the data from state-of-the-art SNS experiments across other neutron centers world-wide. Another example is the hundreds of Gigabytes to Terabytes of NASA satellite hyperspectral data that need to be processed nearly in real time [8].

Two conflicting arguments compete to shape the one-to-many delivery of large-size

scientific data over well provisioned networks. On one side, there is the intuition that the well-provisioned networks are sufficient for guaranteeing good data-delivery performance: sophisticated algorithms that would adapt to unstable or limited-resource environments are superfluous and add unjustified overheads in these environments.

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

This debate motivates our study. We experimentally explore the space of solutions for one-to-many large-scale data delivery in well-provisioned environments via simulations with real-world parameters. We consider solutions typically associated with peer-to-peer applications (such as BitTorrent [9] or Bullet [10]) and evaluate them under our target scenario of large data federations. To this end, we use real, production Grid testbeds such as LCG [11], EGEE [12] and GridPP [13] in our simulations.

The contribution of this study is threefold. First, this study quantitatively evaluates and compares a set of representative data-delivery techniques applied to a realistic grid environment. The quantitative evaluation is then used to derive well-supported recommendations for choosing data-dissemination solutions and for provisioning the Grid networking 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.

Second, in addition to comparing the data dissemination solutions along multiple success metrics (e.g., time to complete the data dissemination, generated overhead, load balance) we are, to the best of our knowledge, the first study to quantitatively evaluate the fairness of these solutions and their impact on competing traffic. Our results show that the impact can be significant and suggest that this aspect bears an important characteristic when choosing the dissemination solution and the network management infrastructure (e.g., differentiated services [14]) that might be used to limit it.

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.

To derive our recommendations, we identify a relevant set of candidate solutions from different domains, build a simulator and evaluate the candidate solutions on key metrics such as time-to-delivery, generated overhead, and load balance.

The rest of this paper is organized as follows. Section 2 presents the data usage characteristics of scientific collaborations and compares them with the assumptions of peer-to-peer file-sharing systems. Section 3 surveys existing work on data dissemination solutions and describes in detail the dissemination schemes analyzed in this paper. Section 4 presents the design of our simulator while Section 5 presents our evaluation results. We summarize our findings in Section 6.

2. Data in Scientific Collaborations

Today, Grids are creating the infrastructure that enables users to dynamically distribute and share massive datasets. However, most data distribution strategies currently in place involve explicit data movement through batch jobs [15, 16] that are seldom sympathetic to changing network conditions, congestion and

latency, and rarely exploit the collaborative nature [17] 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 to data distribution orchestration can include the intelligent use of a set of strategically placed intermediary storage resources as in logistical networking [18]. Alternatively, a more dynamic approach is to split files into chunks and transfer them separately, possibly using different network paths, as in BitTorrent [9]. Yet another approach is to exploit the "orthogonal" bandwidth that might be available outside a traditional, source-rooted data distribution tree [10]. 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 and studied in the context of peer-to-peer file sharing and application-level multicast.

However, such techniques are not directly adaptable to Grid settings because of the different usage scenarios. In particular, 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 characteristics.

The **scale of data** poses unique challenges: scientific data access consists of transfers 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 [19]. This is more than 20 times larger than the 14MB average file size transferred in the Kazaa network in 2002 as reported in [20], but within the same order of magnitude with the files transferred by today's BitTorrent (mainly, movies and software distributions): Bellissimo et al. [21] report an average file size of 600MB).

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 an average data processing rate of 65MB/s [19]. However, *popularity distributions* for scientific data are more uniform than in P2P systems. 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 [21].

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 10 datasets in the DØ traces analyzed in [19] 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 atop 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 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 an unnecessary overhead and may limit the overall system performance.

All these properties (huge size transfers, well provisioned networks, more stable resources, cooperative environments) invite the question if 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 [15].

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 [22], An additional set of reasons for multicast's limited current deployment is its limited support for group management, including authorization for group creation, receiver authorization, and sender authorization, distributed address allocation, security and support for network management [23].

To provide an alternative, numerous research projects explore efficient and scalable data dissemination solutions at the application level. We have identified a number of representative techniques for our comparison. In this section we provide a classification of data distribution techniques (Section 3.1), detail the techniques we explore in this paper (Section 3.2), and present the criteria over which they are typically evaluated (Section 3.3).

3.1 Classification of Approaches

This section identifies three broad categories of data distribution techniques based on: (1) data staging; (2) data partitioning, and (3) exploiting orthogonal bandwidth. Often, deployed solutions use combinations of these techniques. These techniques are described in detail and discussed 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 more 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 a data-sharing overlays with sophisticated application-level routing performed atop the traditional Internet [24].

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 [25] or GridFTP probes [26]. Such information is then used to

make informed decisions regarding routes, so future data transfers can be accomplished in an optimized fashion from preferred locations, based on a delivery constraint schedule [27]. A logical extension is thus to use the participating sites as intermediary data staging points for more efficient data dissemination.

Additionally, a data distribution infrastructure can include a set of intermediary, strategically placed resources (as in logistical computing [18]) to stage data. In this paper we study logistical multicast [18] 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. By simulating the logistical multicast on this hypothetical topology, we aim to quantitatively evaluate its maximum attainable benefit.

3.1.2 Data Partitioning

To add flexibility, various peer-to-peer data distribution solutions split files into chunks and transfer these chunks independently (e.g., BitTorrent [9], Bullet [10], SPIDER [28] and many others). 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 being used in production systems or multicast downloads (e.g., Digital Fountain [29, 30]).

Consequently, partitioning techniques can have significant value in a data Grid collaboration setting, and this prompted us to include Bullet and BitTorrent in our study. For instance, there is a genuine need to provide application-level resilience when it comes to data transfers. Bulk data movement in the Grid is usually involves batch transfers of large files that are required to be resilient in the face of failures such as network outages or security proxy expiration. New solutions in this landscape (e.g., Globus Project' reliable file transfer service too [31], an application-level service atop GridFTP) similarly partition a large data collection into individual files and provide restart markers such that a transfer can be resumed from a failed instance.

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 efficient use of alternate network paths that may be available to speed-up data transfers. In many cases, the bandwidth available to a traditional source-routed distribution tree can be augmented using additional 'orthogonal' network paths that exist between leaf nodes.

This is the premise in a number of commercially deployed (e.g., BitTorrent [9]) or academically designed (e.g., Bullet [10]) data distribution systems. Orthogonal bandwidth tapping relies on partitioning files into chunks and, initially, sending each chunk 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 these techniques 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, which is a scarce resource. 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 have selected previously proposed solutions from each of the categories presented above. We also include other traditional, well understood techniques for comparison. This section presents a brief description of the solutions we evaluate.

Logistical multicast (LMT) [18] (as described earlier in Section 3.1.1), employs strategically placed nodes in an overlay to expedite data distribution. We evaluate an idealized version of this approach: we assume first, that logistical storage is associated with each router in our topologies, and second, we assume 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 of source-rooted distribution trees.

Application-level multicast (ALM) solutions organize participating nodes into a source-rooted distribution tree overlay [32, 33]. 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 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 [33], OMNI [34]), while others aim to integrate end-nodes without infrastructure support (Narada [22], ALMI [32]).
- *Overlay structure*. Some dissemination tree construction algorithms assume the existence of a structured overlay substrate (e.g. CAN multicasting [35], Scribe [36]).
- *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 [32]). At the other end of the spectrum, systems based on structured overlays are able to handle millions of nodes.
- *Success metrics.* 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, NICE [37], OMNI [34]).

Recently, a number of studies have proposed data dissemination algorithms targeting the Grid infrastructure. Grido [38], for example, builds a shortest path first three based on a virtual coordinates system that advise each node of its nearby neighbors. Another system, MOB [39], 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 on next page for details). MOB assumes that clustering information is available and globally known to all the nodes.

For our evaluation we have chosen a centralized solution based on global view of the topology at hand (similar to ALMI [32]) 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 time-to-finish the data transfer is often considered the main data dissemination success metric. Appendix A presents in detail our tree construction heuristic and analyzes its complexity.

SPIDER [28] 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. When SPIDER is only able to build a single tree it is equivalent to traditional IP-multicast. For our simulations, we consider an idealized IP-layer deployment.

Unlike single-tree construction algorithms, SPIDER builds a set of trees, and for each of them tries to maximize the residual bandwidth left for the other trees to be constructed. To achieve this, the construction algorithm selects the link that leaves the maximum outgoing bandwidth for its source node, among the set of candidate links to be added to the current tree.

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 [40] 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 chunks using all trees found.

Bullet [10] offers a way to exploit orthogonal bandwidth by initially distributing disjoint subsets of data on different paths of 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. The tree is used to aggregate fixed size node summaries from the leafs to the source. The source, in turn, will distribute a random subset of these summaries in a fixed size blocks down the tree. Peers use these summaries to discover at other peers in the system the content they are interested in. Further, in the pairwaise exchange between peers, the source node decides which blocks to send to the destination depending on the summary of the destination node. This push-based solution generates duplicate traffic for two reasons: first, incomplete summaries at the sender node lead to the possibility of transmitting duplicate blocks to a destination. Second, multiple sources may decide to send the same block to the destination at the same time.

BitTorrent [9] 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 chunks. BitTorrent assumes a non-cooperative environment and employs a tit-for-tat 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, the node will serve data to the peers that serve

back in return interesting data at the highest download rate.

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. We note that 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 to choose a combination appropriate for a specific application context and deployment scenario. Success metric categories can include:

- *Minimizing transfer times*: Transfer time is often a key metric in data distribution due to the need to send all data to all destination sites so that real-time processing at the end-sites can progress smoothly [41]. 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*: This involves minimizing the load on bottleneck links, the amount of duplicate data transferred, or the aggregate network 'effort' (in megabit x miles) in the distribution tree. 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.
- *Load balancing*: With the enlisting of nodes in the data dissemination effort evenly spreading the load among participants sometimes is an important goal. The load balancing metric helps us study how well different dissemination mechanisms balance 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 application, using multiple connections for a single application will strongly affect other concurrent applications using a single-TCP-connection.

4. Simulating Data Dissemination

In order to evaluate the techniques above, we have built a simulator that works at the file-block level. This section presents the set of decisions that guided our simulator design (Section 4.1), presents key details about simulating the techniques we chose to investigate (Section 4.2), discusses the scope of our simulations (Section 4.3) and finally evaluates and compares our simulator with similar simulators found in the literature (Section 4.4).

4.1 Simulator Design

We have built a high-level simulator to investigate the performance of different data distribution protocols. As in most simulators, the main tradeoff we face is between the resources allocated to simulation and the fidelity of the simulation. At one end of the possible design spectrum are packet-level simulators (such as *ns* [42]) and emulators (such as ModelNet [43] or Emulab [44]): 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 [45]. 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 (which is computationally expensive as it implies simulating each packet's propagation through a series of nodes, queues, and links) with flow-level simulation (which is requires lower computational resources as the characteristics of the network paths are computed only once for each data flow). This technique enabled simulating multicasting trees on systems with hundreds of nodes without considerably reducing the results accuracy [45].

Our simulator sits in between these two extremes. 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.* Since the data dissemination solutions we compare put a different stress on the network and, consequently, may offer better apparent performance simply by being more unfair to competing traffic, we perform our evaluation experiments in two steps. We first evaluate each dissemination solution in isolation (Sections 5.1, 5.2, and 5.3) then we compare their impact on competing traffic (Section 5.4).

4.2 The Data Dissemination Solutions Simulated

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: IP-multicast distribution (and its improvement using SPIDER heuristics) and one-to-one data delivery. 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 in hand, determines routing paths and the network 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. This is necessary due to the non-deterministic nature of these data dissemination solutions. The simulations use a default block size of 512KB (as in deployed BitTorrent systems). We have experimented with different block sizes as well but, because the block size does not have a significant impact on performance, we do not include those results here.

The simulator is composed of three main modules: routing, peering, and block transfer. As their names indicate, the routing module is responsible of 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 interested in exchanging blocks 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.

4.3 The Scope of the Simulation Study

In order to focus on the objectives of our study and on the particularities of our target environment, we limit the axes over which we vary parameters in our study to the strictly required ones. This does not impact the validity of our results, since we are making the most optimistic 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 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).

4.4 Simulator Evaluation

Although we designed our simulator to faithfully simulate the selected strategies and its performance was thus a secondary concern, it is important to evaluate our simulator performance and understand its limitations. Consequently, this section evaluates the simulator performance.

We have generated a set of Waxman topologies using BRITE [46] 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 with a 2.8 GHz processor and 1GB of memory.

The simulator for IP-multicasting, ALM, LM tree, SPIDER and separate transfers from the source to every node, simulates the high-level deterministic protocol behavior. Simulating one of these solutions on

topologies with few thousands end-nodes can be obtained in a few minutes.

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 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 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 1 and Figure 2 present the time required to simulate data dissemination with Bullet and BitTorrent. It is clear from the analysis presented in Table 1, that the BitTorrent simulation has a higher complexity. This is due to the more complex peering (tit-for-tat) and block selection (rarest-first) policies used in BitTorrent. Practically, the simulator can simulate a network of few hundreds of nodes and few thousands of blocks (a typical setting in today's scientific collaboration systems) in few hours.

Although we have not focused on simulation performance (for example our simulator is implemented in Python), compared to other simulators described in literature, our simulator performs well. For instance, Bharambe et al. [47] 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. [48] 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.

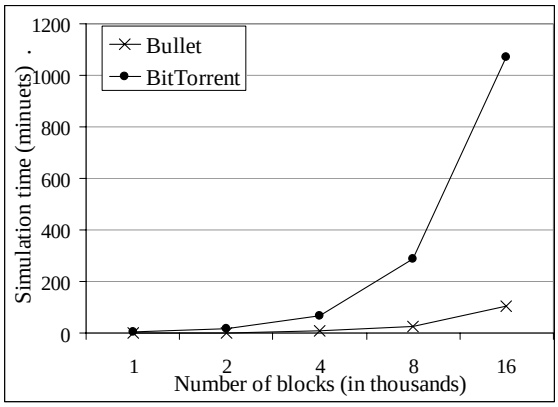

Figure 1: Simulation time for a 25 nodes topology and 1GB file.

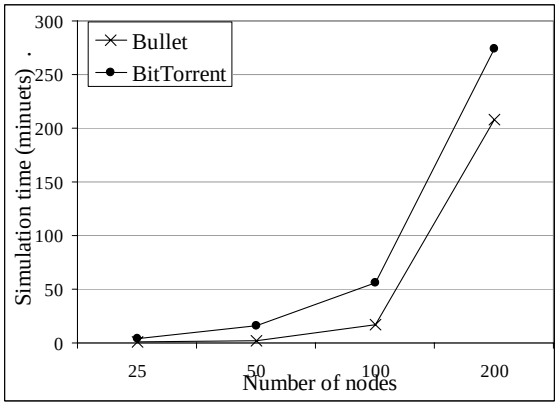

Figure 2: Simulation time for disseminating a 1 GB file (divided into 2000 blocks).

5. Simulation Results

We use the physical network topologies of three real-world grid testbeds LCG [11], EGEE [12] and GridPP [13] (Figure 3, Figure 4, and Figure 5). The LCG topology incorporates 121 sites connected through 10Gbps core links. EGEE and GridPP are smaller but have similar characteristics.

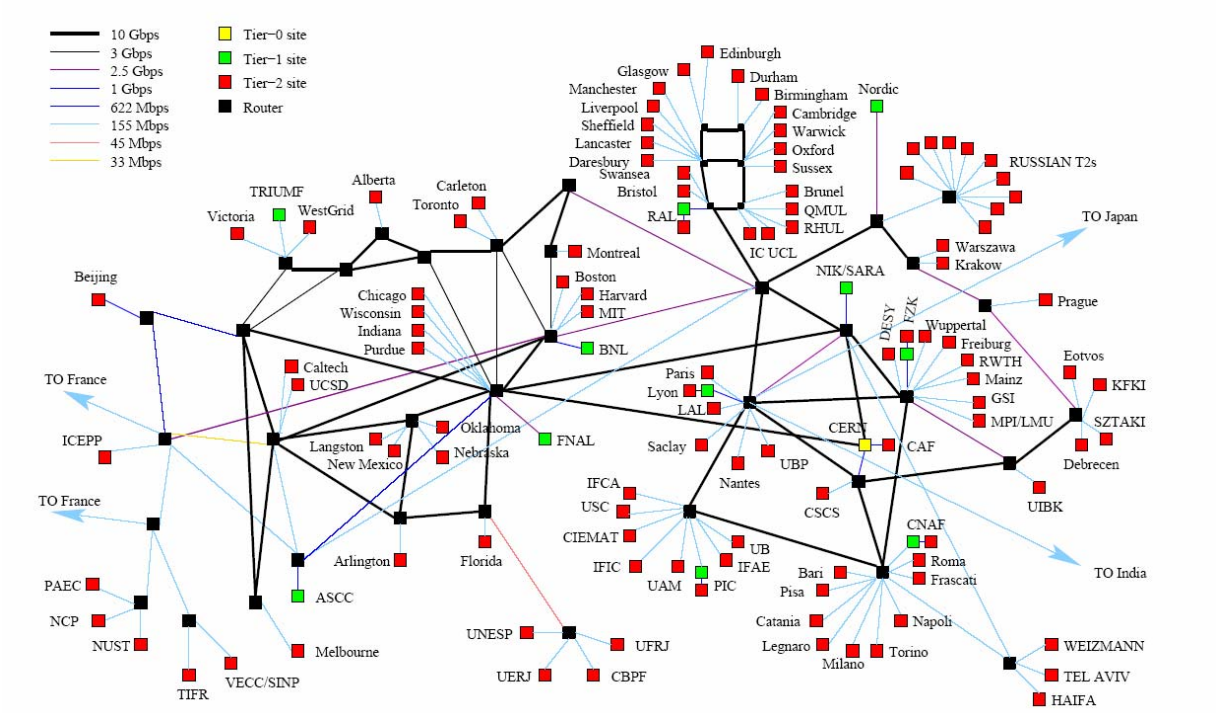

Figure 3: LCG topology. In all simulations the data dissemination source is the CERN node. (Source [11])

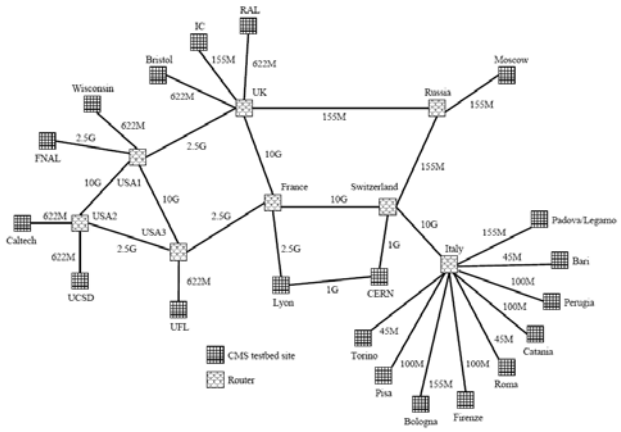

Figure 4: EGEE topology. In all simulations the data dissemination source is the CERN node. (Source: [49])

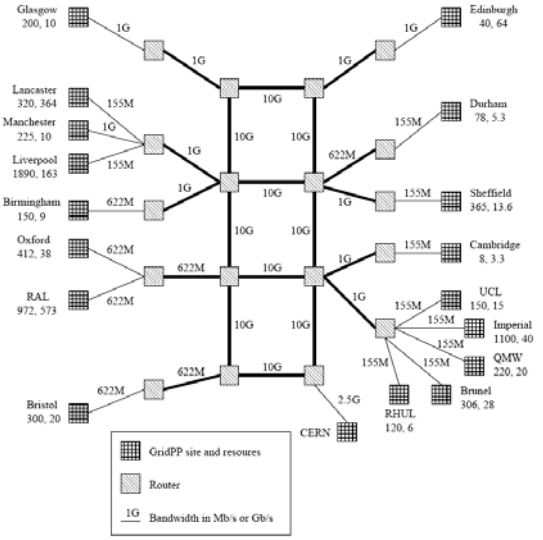

Figure 5 GridPP topology. In all simulations the data dissemination source is the CERN node (Source: [13]).

Additionally, to increase the confidence in our results, we have generated two other sets of Waxman topologies using BRITE [46]. 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 200 nodes total). As these additional

experiments only 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 have chosen these configurations as they offer optimal performance in the real-world topologies we modeled in our experiments (details about how we reached this conclusion are presented in Section 5.5).

In the following subsections, we present our simulation results that compare the techniques we study. We present the data dissemination time in Section 5.1, protocol overhead in Section 5.2, load balancing characteristics in Section 5.3, and fairness to competing traffic in Section 5.4. Section 5.5 presents an experimental validation of our choices of protocols parameters for Bullet and BitTorrent.

5.1 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 of the number of destinations that have completed the file transfer.

Figure 6, Figure 7, and Figure 8 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 or SPIDER is poor. This is because these multicasting schemes do not include buffering at intermediate points in the network and limit their 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 partially a result of the bandwidth distribution in these two 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 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.

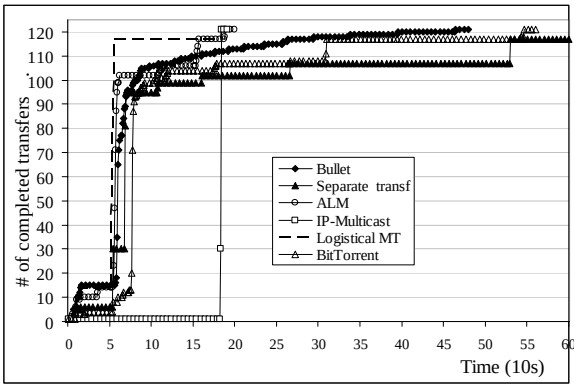

Figure 6: Number of destinations that have completed the file transfer for the original LCG topology (Separate transfer technique finishes the transfer at 730s).

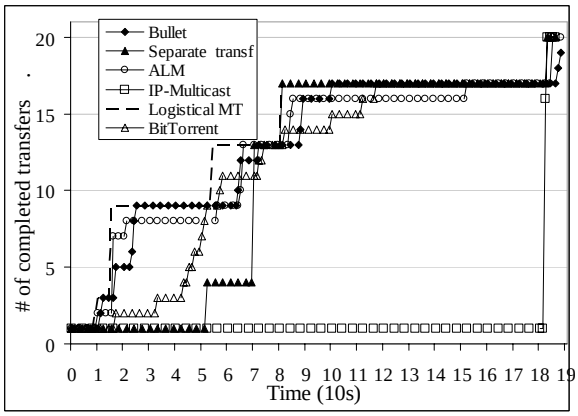

Figure 7: Number of destinations that have completed the file transfer for the original EGEE topology.

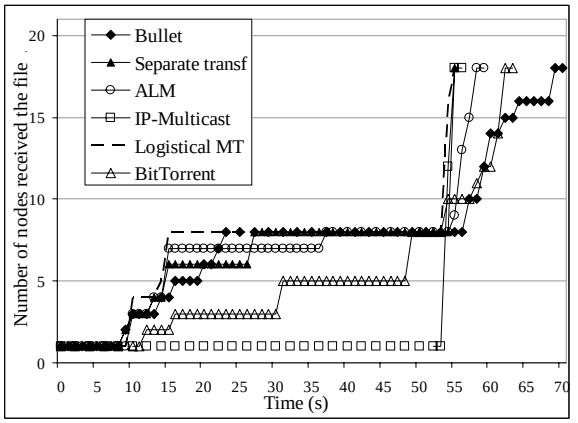

Figure 8: Number of destinations that have completed the file transfer for the original GridPP topology

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.

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 9, Figure 10, and Figure 11, 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.

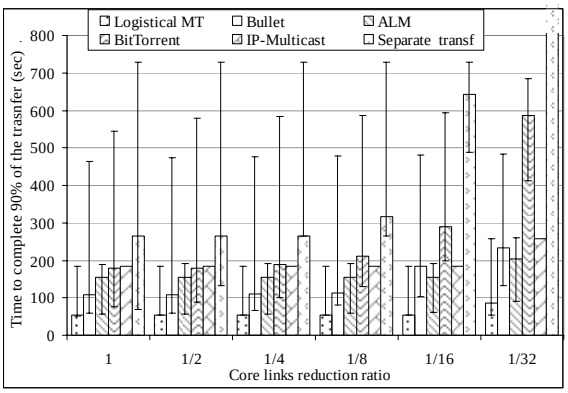

Figure 9: Time to finish the transfer to 90% of the nodes for the original LCG topology and the topology with reduced core bandwidth. The lower error bar indicates the time to complete the transfer for 50% of the nodes while the top error bar indicates the time to complete the transfer for the last node. Separate transfer's technique finishes in 2280 seconds on the topology with core bandwidth reduced to 1/32 (not presented here for clarity).

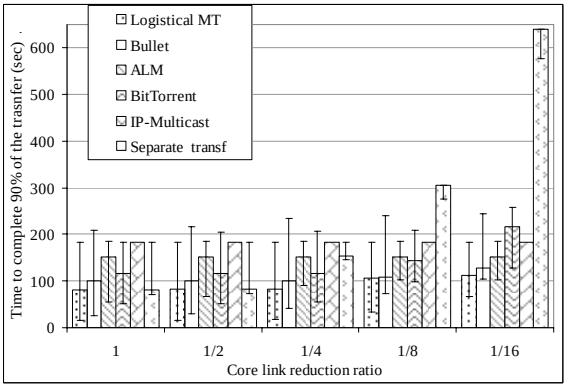

Figure 10: Time to finish the transfer to 50%, 90%, and all nodes for the EGEE topology – original and reduced core bandwidth.

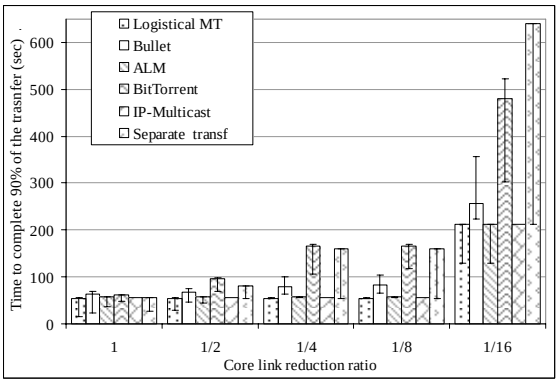

Figure 11: Time to finish the transfer to 50%, 90%, and all nodes for the GridPP topology – original and reduced core bandwidth..

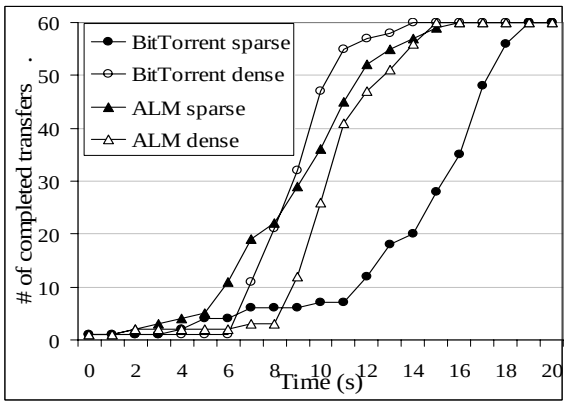

Figure 12: Number of destinations that have completed the file transfer with two generated topologies. The dense topology has four times more links in the core with four times less average bandwidth per link.

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, for our topologies 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 generate two sets of topologies in which the aggregate core bandwidth is maintained constant but the number of core links is varied. Figure 12

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 12, BitTorrent performance is better with more links in the core while ALM performance slightly degrades. The 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 specific to peer-to-peer systems).
- Bullet and BitTorent 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.

5.2 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 13 and Figure 14 show the generated useful and overhead traffic for each protocol for the original LCG and EGEE topology. We define as *useful* the data traffic that remains after excluding all link-level duplicates. Note that the volume of useful traffic differs between the protocols, since different schemes map differently on the physical layer.

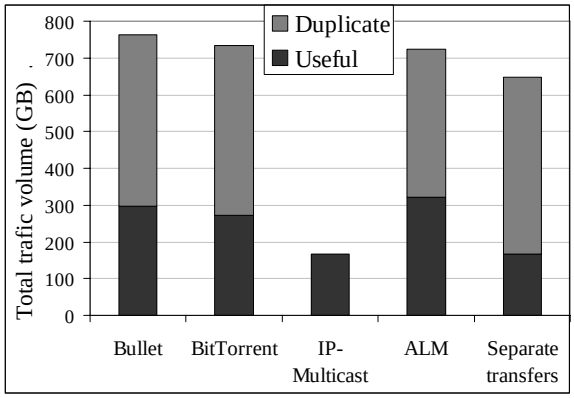

Figure 13: Overhead of each protocol on LCG Topology.

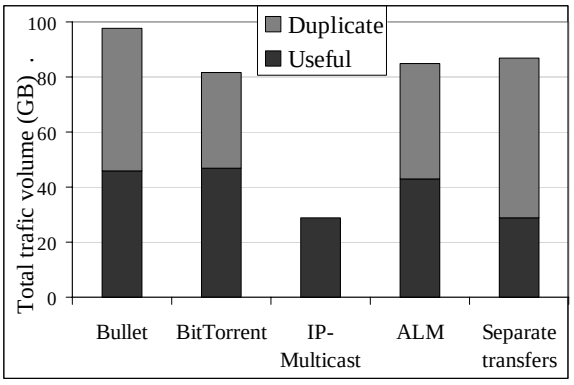

Figure 14: Overhead of each protocol on EGEE Topology.

The following observations can be made based on Figure 13 and Figure 14 (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.

Summary. Application-level multicasting generates significant overheads (up to four times more compared to optimal IP-level optimal 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 IP-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.

5.3 Load Balance

Another metric to evaluate the performance of data dissemination schemes is load balancing. 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 balancing. Here, 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.

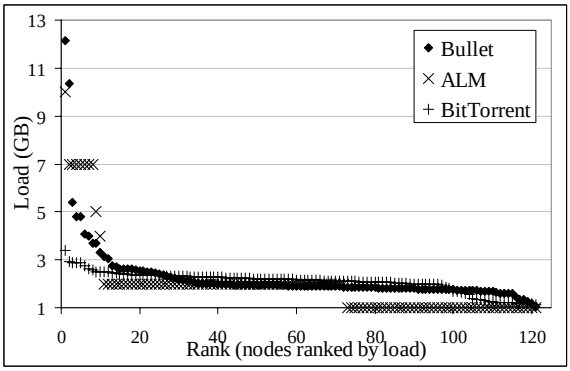

Figure 15: Load balancing for ALM, BitTorrent and Bullet. Nodes are ranked in decreasing order of their load (LCG topology).

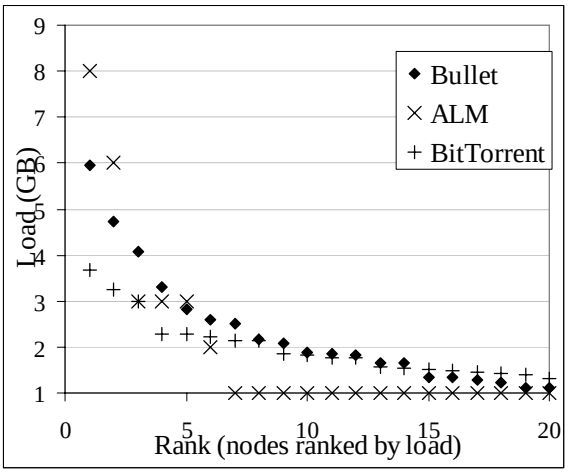

Figure 16: Load balancing for ALM, BitTorrent and Bullet. Nodes are ranked in decreasing order of their load (EGEE Topology).

Figure 15 and Figure 16 present the load balancing performance of the remaining techniques: ALM, BitTorrent, and Bullet. These results are obtained for LCG and EGEE topologys, but again, the relative order of these techniques in terms of load balance does not change across all our experiments.

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 every destination. Additionally, BitTorrent offers the best load balance among application-level solutions.

5.4 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.

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 [50], 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.

While different application-layer data dissemination solutions do have different impact levels on competing traffic, we are not aware of any related work analyzing their fairness. 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 will have the highest impact on competing traffic. For instance, in FastReplica [51], 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 17 and Figure 19 shows. The maximum link stress generated by Bullet and BitTorrent can be as high as 70 and 23 on the LCG and EGEE topology, respectively, as Figure 18 and Figure 20 show. 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.

Finally, the technique of sending the file from the source to every destination through separate channels is the worst in terms of fairness to competing traffic (considering that all generated unicast traffic is part of the same application), since the unicast connection from the source to every destination will pass through the same links around the source.

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

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

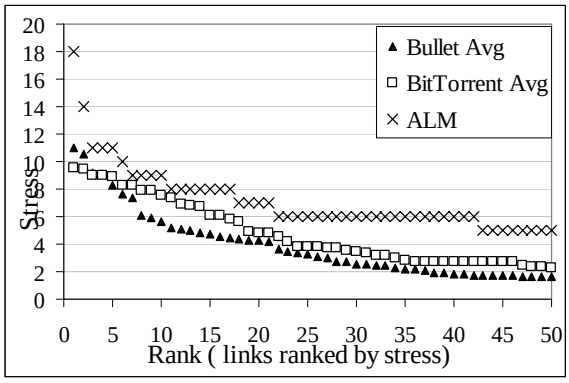

Figure 17: Link stress distribution of BitTorrent and Bullet over the LCG topology. The plot presents average link stress for the most stressed 50 links.

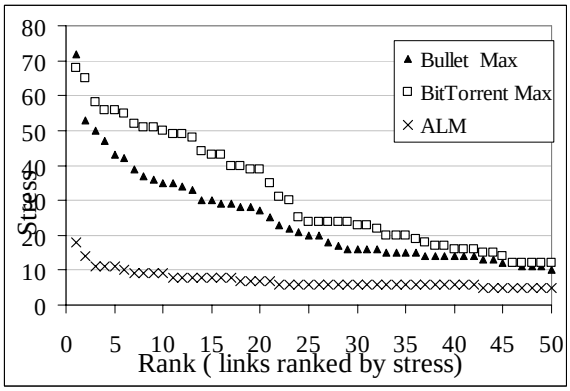

Figure 18: Link stress distribution of BitTorrent and Bullet over the LCG topology. The plot presents maximum link stress for the most stressed 50 links.

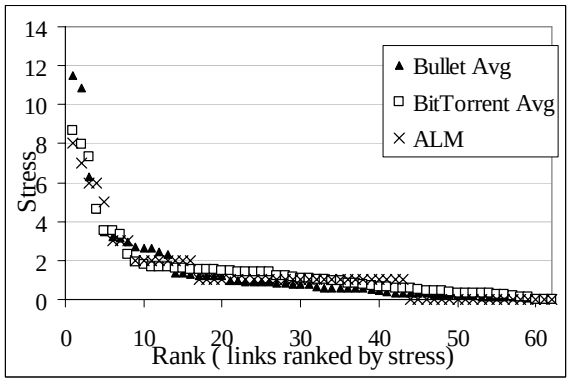

Figure 19: Link stress distribution of BitTorrent and Bullet over the EGEE topology. The plot presents the average link stress.

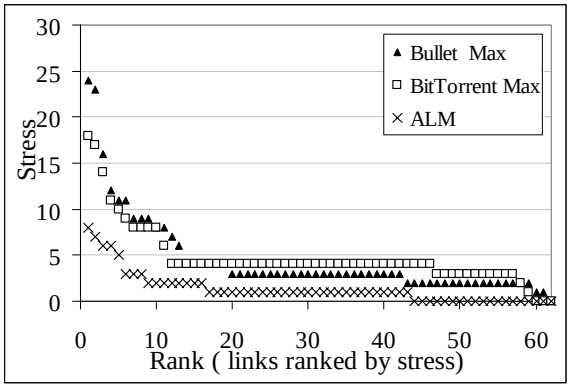

Figure 20: Link stress distribution of BitTorrent and Bullet over the EGEE topology. The plot presents the maximum link stress.

5.5 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 vary the number of peering relationships.

We find that for our topologies, configuring Bullet with two peers (in the three topologies) and BitTorrent with four peers in EEGE and GridPP topologies and eight peers in LCG topology (incidentally the default configurations for BitTorrent deployments) 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 provides the fastest data dissemination. Increasing the number of peers results in larger generated overheads. Configuring BitTorrent with four peers in EGEE and GridPP topologies and with eight peers in LCG topology provides the fastest data dissemination, increasing the number of peers provides better load balancing without generating additional overhead. Regarding fairness to competing traffic, increasing the number of peers in Bullet and BitTorrent reduces fairness around the nodes with high access bandwidth, as these nodes will obtain the complete file first in the data dissemination process and will continue to serve a large number of nodes.

6. 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 P2P solutions for one-to-many data dissemination offer?

Our simulation-based experimental investigation on seven solutions drawn from Internet data delivery and peer-to-peer networks 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 P2P 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 file-delivery 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, 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 P2P-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.

In short, the P2P 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 for example) P2P 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.

7. 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. *The D0 Experiment, Fermi National Laboratory*, http://www-d0.fnal.gov. 2002.
4. Catlett, C., *The TeraGrid: A Primer*. 2002, www.teragrid.org.
5. Brown, M., *Blueprint for the Future of High-Performance Networking.* Communications of the ACM (CACM), 2003. **46**(11): p. 30-77.
6. Allcock, W., et al. *Protocols and Services for Distributed Data-Intensive Science*. in *Advanced Computing and Analysis Techniques in Physics Research (ACAT)*. 2000: AIP Conference Proceedings.
7. Bassi, A., et al., *The Internet Backplane Protocol: A Study in Resource Sharing.* Future Generation Computing Systems, 2003. **19**(4): p. 551-561.
8. Plaza, A., et al., *Commodity cluster-based parallel processing of hyperspectral imagery.* Journal of Parallel and Distributed Computing, 2006. **66**(3): p. 345–358.
9. Cohen, B., *BitTorrent web site:* http://www.bittorrent.com. 2005.
10. Kostic, D., et al. *Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh*. in *SOSP'03*. 2003. Lake George, NY.
11. 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.
12. *Enabling Grids for E-sciencE Project*. 2006.
13. Britton, D., et al. *GridPP: Meeting the Particle Physics Computing Challenge*. in *UK e-Science All Hands Conference*. 2005.
14. Blake, S., et al., *An Architecture for Differentiated Services*. 1998, IETF.

15. Terekhov, I., et al. *Distributed data access and resource management in the D0 SAM system*. in *IEEE International Symposium on High Performance Distributed Computing*. 2001.
16. Wang, F., et al. *File System Workload Analysis For Large Scientific Computing Applications*. in *NASA/IEEE Conference on Mass Storage Systems and Technologies (MSST 2004)*. 2004.
17. Iamnitchi, A., M. Ripeanu, and I. Foster. *Small-World File-Sharing Communities*. in *Infocom 2004*. 2004. Hong Knog.
18. Beck, M., et al. *Logistical Networking: Sharing More Than the Wires*. in *Active Middleware Services Workshop*. 2000. Norwell, MA.
19. Iamnitchi, A., S. Doraimani, and G. Garzoglio. *Filecules in High-Energy Physics: Characteristics and Impact on Resource Management*. in *HPDC 2006*. 2006. France.
20. Gummadi, K.P., et al. *Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload*. in *SOSP'03*. 2003. Lake George, NY.
21. 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.
22. Chu, Y.-h., et al., *A Case for End System Multicast.* IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast, 2002. **20**(8).
23. Diot, C., et al., *Deployment Issues for the IP Multicast Service and Architecture.* IEEE Network: Special Issue on Multicasting, 2000. **14**(1).
24. Touch, J.D., *Overlay Networks.* Computer Networks, 2001. **36**(2001): p. 115-116.
25. 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.
26. 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.
27. 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.
28. Ganguly, S., et al. *Fast Replication in Content Distribution Overlays*. in *IEEE INFOCOM*. 2005. Miami, FL.
29. Byers, J.W., et al. *A Digital Fountain Approach to Reliable Distribution of Bulk Data*. in *SIGCOM*. 1998.
30. Byers, J., et al. *Informed Content Delivery Across Adaptive Overlay Networks*. in *SIGCOMM2002*. 2002. Pittsburg, PA.
31. *Globus Toolkit - Reliable File Transfer Service (www-unix.globus.org/toolkit/reliable_transfer.html)*. 2003.
32. Pendarakis, D., et al. *ALMI: An Application Level Multicast Infrastructure*. in *USITS'01*. 2001.
33. Jannotti, J., et al. *Overcast: Reliable Multicasting with an Overlay Network*. in *4th Symposium on Operating Systems Design and Implementation (OSDI 2000)*. 2000. San Diego, California.
34. Banerjee, S., et al., *OMNI: an efficient overlay multicast infrastructure for real-time applications.* Computer Networks: The International Journal of Computer and Telecommunications Networking, 2006. **50**(6).
35. Ratnasamy, S., et al. *Application-Level Multicast Using Content-Addressable Networks*. in *Third International COST264 Workshop on Networked Group Communication*. 2001.
36. Castro, M., et al., *Scribe: A large-scale and decentralized application-level multicast infrastructure*. IEEE Journal on Selected Areas in Communication (JSAC), 2002. **20**(8).
37. Banerjee, S., B. Bhattacharjee, and C. Kommareddy. *Scalable Application Layer Multicast*. in *SIGCOMM2002*. 2002. Pittsburgh, PA.
38. Das, S., et al. *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.
39. 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.
40. Izmailov, R. and S. Ganguly. *Fast Parallel File Replication in Data Grid*. in *Future of Grid Data Environments workshop, GGF - 10*. 2004. Berlin, Germany.
41. 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.
42. *The Network Simulator - ns-2*. 2006.
43. Vahdat, A., et al. *Scalability and Accuracy in a Large-Scale Network Emulator*. in *OSDI*. 2002.
44. White, B., et al. *An Integrated Experimental Environment for Distributed Systems and Networks*. in *OSDI*. 2002. Boston, MA.
45. 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.
46. Medina, A., et al. *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.
47. Bharambe, A.R., C. Herley, and V.N. Padmanabhan. *Analysing and improving a BitTorrent network's performance mechanisms*. in *The 25rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2006)*. 2006. Barcelona, Spain.
48. 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.
49. Cameron, D.G., et al., *Analysis of Scheduling and Replica Optimisation Strategies for Data Grids Using OptorSim*. Journal of Grid Computing, 2004. **2**(1): p. 57-69.
50. Yang, Y.R. and S.S. Lam. *Internet Multicast Congestion Control: A Survey*. in *ICT 2000*. 2000. Acapulco, Mexico.
51. 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.

8. 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 $O(n^3)$. And 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
```