J Grid Computing (2009) 7:91–114
DOI 10.1007/s10723-008-9113-0

Beyond Music Sharing: An Evaluation of Peer-to-Peer Data Dissemination Techniques in Large Scientific Collaborations

Samer Al-Kiswany · Matei Ripeanu · Adriana Iamnitchi · Sudharshan Vazhkudai

Received: 8 October 2007 / Accepted: 27 November 2008 / Published online: 16 December 2008

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. A suitable 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.

Keywords Data dissemination · Application level multicast · Peer-to-peer · Performance evaluation

S. Al-Kiswany (✉) · M. Ripeanu
Electrical and Computer Engineering Department,
The University of British Columbia,
Vancouver, Canada
e-mail: samera@ece.ubc.ca

M. Ripeanu
e-mail: matei@ece.ubc.ca

A. Iamnitchi
Computer Science and Engineering,
University of South Florida,
Tampa, FL, USA
e-mail: anda@cse.usf.edu

S. Vazhkudai
Computer Science and Mathematics Division,
Oak Ridge National Laboratory,
Oak Ridge, TN, USA
e-mail: vazhkudaiss@ornl.gov

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 generate petabytes (10^{15} bytes) of data that need to be analyzed by large, geographically dispersed user communities, requesting more than ever efficient data-dissemination solutions. Examples of data-intensive collaborations 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]. Enabling 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]).

However, most data distribution strategies currently in place involve explicit data movement through batch jobs [8, 9] that are seldom sympathetic to changing network conditions, congestion and latency, and rarely exploit the collaborative nature [10] 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 [11]. Another approach is to exploit the orthogonal bandwidth that might be available outside a traditional, source-rooted data-distribution tree [12]. 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, these techniques have been deployed [11] and studied [13, 14] 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 in Grids is the dynamic distribution 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 NASA satellite hyperspectral data that need to be processed in near real time [15].

Two conflicting arguments compete for designing one-to-many delivery systems 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 using real-world network topologies. We consider solutions typically associated with peer-to-peer applications (such as BitTorrent [11] or Bullet [12]) and evaluate them in the large-scale data federation scenario. To this end, we use three real topologies of production Grid testbeds in our simulations: LCG [16], EGEE [17] and GridPP [18].

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. Our goal in this article is to analyze and contrast the many popular data dissemination solutions that already exist rather than propose new ones. As we point out, an entire suite of solutions already popular in the peer-to-peer domain are hardly used in Grid settings. In this situation, we feel that a systematic study of the merits of the various data dissemination strategies using multiple success metrics is timely and potentially more useful than introducing new solutions. 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 data dissemination, 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 is 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 appropriate solutions for specific deployments and can be extended to study new data dissemination solutions.

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 and describes in detail the dissemination schemes this study analyzes. Section 4 presents the design of our simulator and Section 5 presents our evaluation results. We summarize our findings in Section 6.

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 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 1 GB and the mean file size is larger than 300 MB [19]. This is more than 20 times larger than the 14 MB average file size transferred in the Kazaa network in 2002 as reported in [20], but within the same order of magnitude with the files currently transferred by BitTorrent (mainly, movies and software distributions): Bellissimo et al. [21] report an average file size of 600 MB.

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 [19]. 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 [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 terabytes. For example, the largest ten 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 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 indeed limit the overall system performance.

All these properties (huge size transfers, well provisioned networks, more stable resources, cooperative environments) invite the question of 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 efficient and overloads the data source, it is often adopted in current deployments [8].

A second well-understood solution is to use IP multicast. IP multicasting lowers the load on the data source and provides a scalable solution for unreliable connectionless multimedia multicasting [22]. However, despite significant efforts, IP multicast is not widely deployed as it requires new routing hardware, and faces challenging problems in providing reliability, congestion and flow control [23]. In addition, IP multicast is not widely deployed due to its limited support for group management, including authorization for group creation, receiver and sender authorization, distributed address allocation, support for network management, and limited support for congestion control [24].

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 paper (Section 3.2), and presents the criteria over which data dissemination solutions are typically evaluated (Section 3.3).

However, before delving into these techniques, it is important to note that no solution is 'optimal' for all deployment scenarios and success metrics. Generally, solutions that do well on a particular success metric (e.g., dissemination time) often compromise on others (e.g., fairness, or generated overheads). This situation further motivates our longitudinal study that quantitatively compares dissemination schemes in realistic deployment environments using a multitude of success metrics. Section 3.4 discusses more the optimality of the proposed solutions.

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 for data distribution. 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 atop the traditional Internet [25].

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 [26] or GridFTP probes [27]. 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 [28]. 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 [29]) to stage data. In this paper we study an idealized version of logistical multicast 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 [11], Bullet [12], SPIDER [30] 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 [31, 32]).

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 [12], a data dissemination solution developed in academia, and the widely popular BitTorrent file-sharing protocol [11].

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 and the participants are cooperative, as is the case with modern data collaborations deployed over heavily provisioned networks.

3.2 Candidate Solutions

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

Logistical multicast (LMT) [29] (as described earlier in Section 3.1.1), employs strategically placed nodes in an overlay to support data distribution. We evaluate an idealized version of this approach: we assume that logistical storage is associated with each router, 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.

Application-level multicast (ALM) solutions organize participating nodes into a source-rooted overlay tree used for data dissemination [33, 34]. 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 endnodes 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 as follows:

- *Resource ownership and infrastructure*. Some systems rely on strategically placed infrastructure proxies to support the construction of their data distribution trees (Overcast [34], OMNI [35]), while others aim to integrate end-nodes without infrastructure support (Narada [23], ALMI [33]).
- *Overlay structure*. Some dissemination tree construction algorithms assume the existence of a structured [36, 37] or unstructured [38] overlay substrate while others build the dissemination tree from scratch.
- *Centralized vs. distributed tree construction.* For small and medium scale systems centralized tree construction and management algorithms based on a full system view have been designed (e.g., ALMI [33]). At the other end of the spectrum, systems based on structured overlays are able to handle millions of nodes operating with partial views of the system.
- *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 message delay or generated network traffic (e.g. Narada, NICE [39], OMNI [35]).

Recently, a number of studies have proposed data dissemination algorithms targeting the Grid infrastructure. Grido [40], for example, builds a shortest-path-first tree based on a virtual coordinates system that advises each node of its nearby neighbors. Another system, MOB [41], adopts a hierarchical approach, wherein 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 below for details). MOB assumes that clustering information is available and globally known to all the nodes.

For our evaluation we chose a centralized solution based on global topology view (similar to ALMI [33]) appropriate for the scale we target and offering near optimal trees in terms of dissemination bandwidth. 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-completion of a data transfer is often considered the main data dissemination success metric. Our technical report [42] presents in detail our tree construction heuristic and analyzes its complexity.

In addition to solutions using a single trees we explore the performance of solutions using multiple source-rooted trees like *SPIDER* [30], which offers a set of heuristics that enables fast content distribution by building multiple source-rooted trees assuming global views. This way, SPIDER exploits existing orthogonal bandwidth. This technique can be used at the application as well as at lower network layers. For our simulations, we consider a scenario where SPIDER algorithms are used at the network layer, which offers an upper bound for solutions based on multiple source-rooted trees. Note that when SPIDER is able to build only a single tree it is equivalent to traditional IP-multicast.

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. For this, the construction algorithm selects from a set

of candidates the link that leaves the maximum outgoing bandwidth for its source node.

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 [43] constructs multiple, source-rooted multicast 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. Data to be distributed is then multicast in fixed-size blocks using all trees found.

Bullet [12] offers a way to exploit orthogonal bandwidth by initially distributing disjoint subsets of data on different paths of a distribution tree (while we use the ALM-built tree in this study the Bullet project demonstrates that the choice of the original tree is not essential). 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 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 pairwise 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.

BitTorrent [11] 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 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, 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 highly dependent on the application context. Thus, no data distribution solution is optimal for all cases and a careful evaluation of various techniques is required when 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. The focus can be 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 different dissemination mechanisms balance load among participating nodes.
- *Fairness* to other concurrent transfers can be an important concern depending on the level of isolation offered by lower-layer network

levels 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 concurrent applications operating in a single flow mode.

3.4 A Note on Optimality

It is important to note that in generic, realistic settings optimality is not achievable when considering the dissemination time related metrics, and even less when defining aggregate metrics that combine the above metrics using different weights.

A number of reasons support the above statement. First, on arbitrary topologies, determining a dissemination overlay that is optimal in terms of bandwidth is not scalable, because determining optimal overlays is at least an NP-complete problem [30, 44]. Second, the problem is further complicated by typical distributed system complexity related to the dynamicity of the system and the impossibility to guarantee accurate global system views when failures may be present. Finally, it is infeasible to optimize for the four different metrics due to their conflicting requirements. For instance, a data dissemination mechanism that is optimal in terms of load balancing will enforce that each node sends as much data as it receives, which results in a solution that can not be optimal in terms of transfer time, mainly because of the heterogeneous nodes capability and links bandwidth.

In restricted settings, however, optimal solutions can be deducted. For example, the particular characteristics of the topologies we use make logistical multicasting achieve optimal transfer times as this scheme is able to saturate all bottleneck links and does not incur overheads.

On the other side, for other individual metrics optimality can be immediately determined. For example, for the network overheads metric, IP- and logistical multicast provide an optimal solution as they use the router's duplication capability and the physical network topology information to avoid sending duplicate packets.

4 Simulating Data Dissemination

In order to evaluate the techniques above, we built a simulator that works at the file-block level. This section presents key details about simulating the techniques we chose to investigate (Section 4.1) and the simulation approach (Section 4.2) that guided the simulator design (Section 4.3). Further, this section discusses the scope of our simulations (Section 4.4), the validation of our simulator (Section 4.5) and evaluates and compares our simulator with similar simulators in the literature (Section 4.6).

4.1 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 analyze 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 minimizing traffic overhead and node load-balance. SPIDER builds multiple IP multicast trees in order to best exploit the existing bandwidth through different paths from source to destinations. Consequently, SPIDER performs identically to IP multicasting on sparse topologies where it can build only one tree but may show improvements on dense topologies.

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.

4.2 Simulation Approach

We built a high-level simulator to investigate the performance of different data distribution protocols. As with 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 design spectrum are packet-level

simulators (such as *ns* [45]) and emulators (such as ModelNet [46] or Emulab [47]): they require significant hardware resources but model application performance faithfully by running unmodified application code and simulating or 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 [48]. 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 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 have been successfully used to simulate multicast trees with hundreds of destinations without considerably reducing result accuracy [48].

Our simulator sits in between the two extremes above. The granularity used is file-blocks, a natural choice since many of the data dissemination schemes we investigate use file blocks as their data management unit. Individual block transfers are simulated at flow level. While we do not simulate at the packet level, however, we do consider all properties of the physical network and simulate link-level contention between application flows.

This approach, similar to the abstracted simulation approach presented in [48], offers a good balance between simulation scalability (as flow-level simulation require fewer resources), network simulation accuracy (as we do consider contention at physical link level), and accuracy in simulating dissemination techniques like BitTorrent or Bullet that continuously reconfigure data paths during the dissemination process.

In addition, our simulator design is guided by the following 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 control messages overlap or are often piggybacked on actual traffic. 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 the centralized versions we simulate (using global views) is an upper bound of the performance of original distributed versions.
- *Isolated evaluation.* The 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 5.2, 5.3, and 5.4), then we compare their impact on competing traffic (Section 5.5).

4.3 Simulator Design

For the naïve dissemination solution (where the source sends a copy of the file separately to each node) as well as for all tree-based solutions where all flows are stable during the entire simulation, the simulator analyses the physical topology at hand, determines the routing paths and the flow contention at the physical link level, and estimates transfer performance for each flow which is then used to compute data transfer times.

For the more 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.

For these protocols, 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 to decide flow paths using a shortest path algorithm based on network topology information; the peering module is responsible for constructing peering relationships between nodes according to the specific dissemination protocol specification; and, finally, the block transfer module uses the information provided by the two other modules to simulate block transfers between peers using the paths provided by the routing module while accounting for link level contention. We note that 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 the routing information for the topology is computed, the simulation works in rounds for the two dissemination solutions that use temporary peering, i.e., Bullet and BitTorrent. In each round, first, the peering then the transfer module are executed. The peering algorithm analyzes the content of each node and identifies the pairs of nodes that have to exchange data in the next round. Then, using these pairs and the computed routing paths between nodes, flow-level simulations can decide on network contention on each physical link, which determines the amount of data each flow can carry in the round. Finally, the set of blocks to be exchanged are selected, the actual block transfer is simulated, and timers can be advanced. 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 determining the overall simulation speed. In our simulation we set the round length to a quarter of a second, a reasonable small value compared to the time taken to finish the transfer by the fastest mechanism.

4.4 Simulation Study Scope

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).

4.5 Simulator Validation

We validated the correctness of simulator in three ways: we have inspected detailed execution log, we have compared simulation results and performance predictions obtained analytically on small topologies, and, finally, we have been able to reproduce a qualitative result presented in the Bullet original paper [12]. This section details these three avenues.

Firstly, we ran a set of Bullet and BitTorrent simulations on small topologies. For each, we logged all operations performed by the simulator. The detailed inspection of the simulator logs verified that each of the simulator's three main modules implements exactly the target mechanisms.

Secondly, we executed simulations on small, regular topologies for which computing data transfer times and load balance are tractable analytically. Simulation results on both these metrics matched the analytical results.

Finally,we considered repeating the experiments presented in previous Bullet or BitTorrent publications. Unfortunately we have not been able to find publications that make the experiments entirely reproducible. There are a number of reasons: the physical topologies used were not described in enough detail, or the results were based on experiments performed on non-isolated testbeds (e.g., PlanetLab [49]) where uncontrolled competing traffic influenced the results presented, or the level of abstraction used in simulations was different, or, finally, the simulation focused on evaluating performance in presence of nodes failures.

To avoid these issues we attempted to reproduce the main *qualitative* result of Kostic al. [12] using our simulator. To this end we compared the performance of Bullet using a bandwidth optimized tree ($\text{Bullet}_{\text{BOT}}$) and that of Bullet using a randomly generated tree ($\text{Bullet}_{\text{RandTree}}$). We simulated these two Bullet versions on three real-world testbeds (described in Section 5.1). Our results confirm the results of Kostic et al., namely that, Bullet performance is largely unaffected by the type of the dissemination tree selected for the data main stream dissemination.

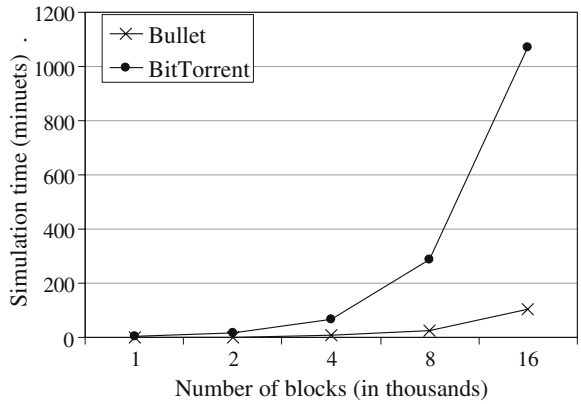

Fig. 1 Simulation time for a 25 node topology and 1 GB file

4.6 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 simulators used for similar studies.

IP-multicast, ALM, logistical multicast, SPIDER and independent transfers from the source to every node use deterministic protocols and the data distribution paths are stable during data dissemination. Consequently, their simulation is less complex than that of BitTorrent and Bullet, which use block level simulation.

Bullet and BitTorrent simulation starts with running the same routing module. However, each of them has a complex peering and block transfer module reflecting the respective 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.

Table 1 shows that the BitTorrent simulation has a higher complexity. This is a result of the complex peering (tit-for-tat) and block selection (rarest-first) policies used.

To evaluate the simulator performance in real settings we generated a set of Waxman topologies using BRITE [50] 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.

Figures 1 and 2 present the time required to simulate data dissemination with Bullet and BitTorrent. Practically, 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. We note that simulating the other protocols is much faster: topologies with few thousands end-nodes can be simulated in a few minutes.

Although we have not focused on performance (e.g., we implemented the simulator in Python), our simulator performance compares well to others described in literature. For instance, Bharambe et al. [14] present results for simulating

Table 1 The complexity of Bullet and BitTorrent protocol's modules

Module	Bullet	BitTorrent
Routing	$O(E^3 \times L)$	$O(E^3 \times L)$
Peering	$O(E^2 \times B \times \text{Log}(B))$	$O(E^2 \times B.\text{Log}(B) + E^3)$
Block transf.	$O(E \times P \times B)$	$O(E \times P^2 + E \times P \times \text{Log}(N))$

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

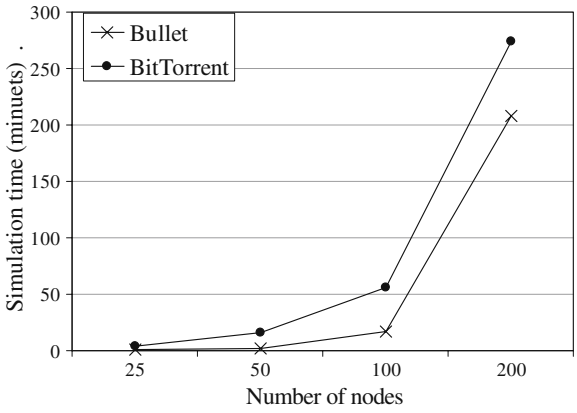

Fig. 2 Simulation time for disseminating a 1 GB file (divided into 2,000 blocks)

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 and Rodriguez [51] 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.

5 Simulation Results

This section presents the results of our comprehensive simulation study. We detail our experimental setup in Section 5.1 and present simulation results that compare, along multiple success metrics, the techniques we study. We present the data dissemination time in Section 5.2, protocol overhead in Section 5.3, load balancing characteristics in Section 5.4, and fairness to competing traffic in Section 5.5. Section 5.6 presents an experimental validation of our choices of protocols parameters for Bullet and BitTorrent.

5.1 Experimental Setup

We use the physical network topologies of three real-world Grid testbeds LCG [16], EGEE and GridPP [18] (Figs. 3, 4, 5). The authors of the above references have obtained the above topologies either from the network monitoring data [17] or from publicly available information on connectivity and link bandwidth [16]. The LCG topology incorporates 121 sites connected through 10 Gbps 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 three other sets of Waxman topologies using BRITE [50]. The first 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 1 GB 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 5.5).

The simulations use a default block size of 512KB as in deployed BitTorrent systems [11, 52]. 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.

5.2 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, Nth 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.

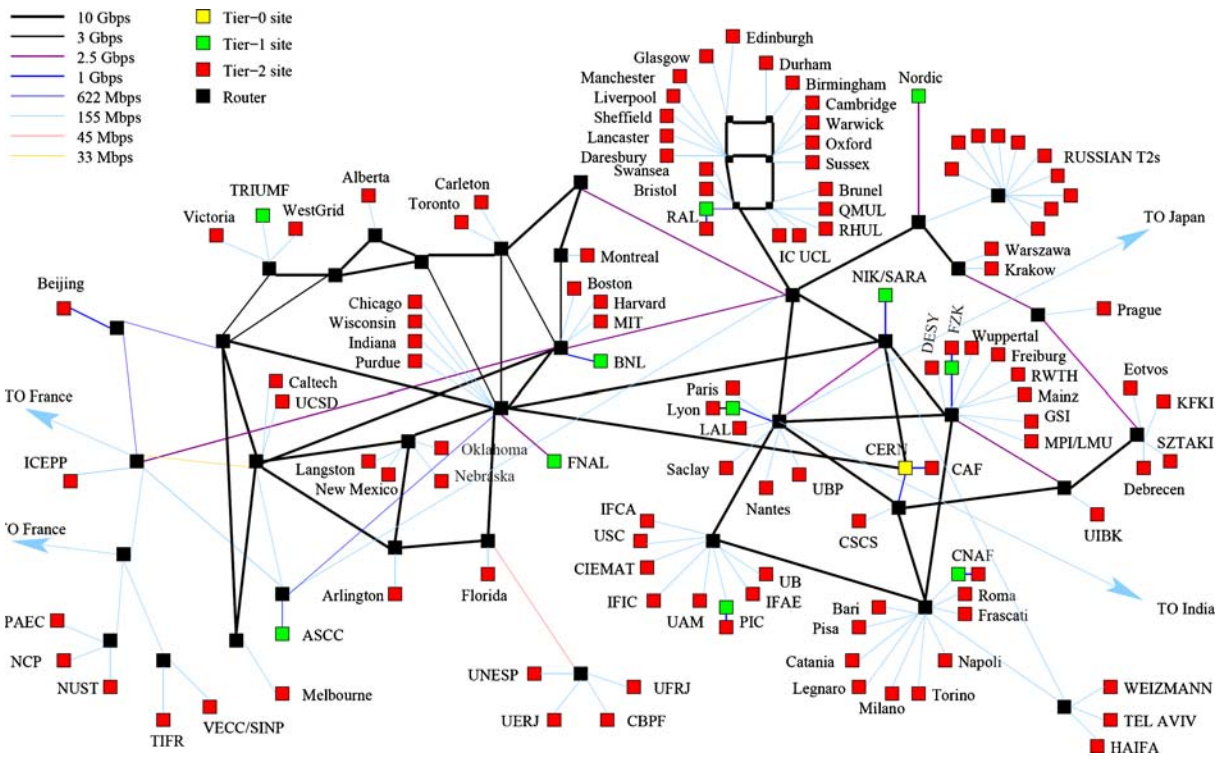

Fig. 3 LCG topology. In all simulations the data dissemination source is the CERN node (source [16])

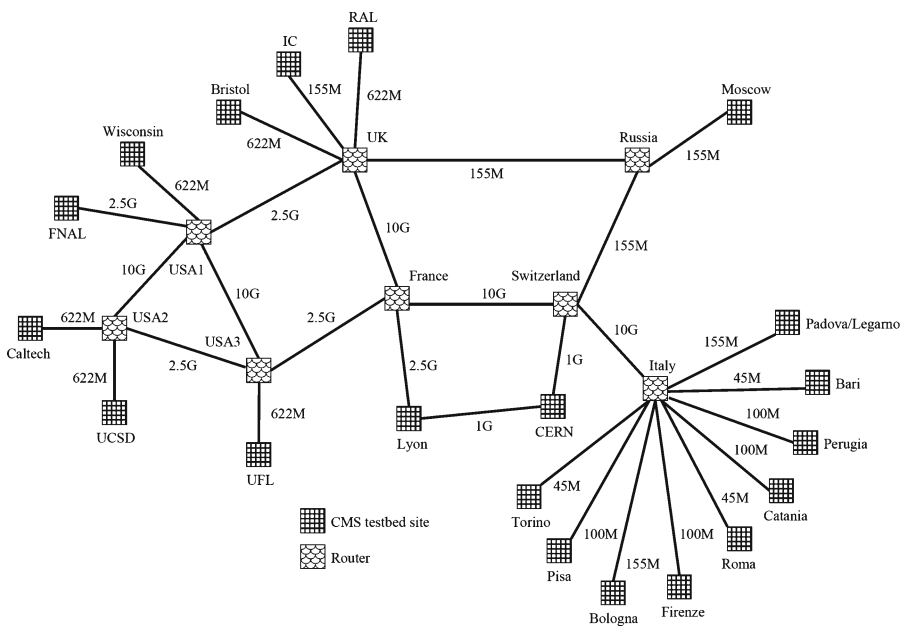

Fig. 4 EGEE topology. In all simulations the data dissemination source is the CERN node (source: [17])

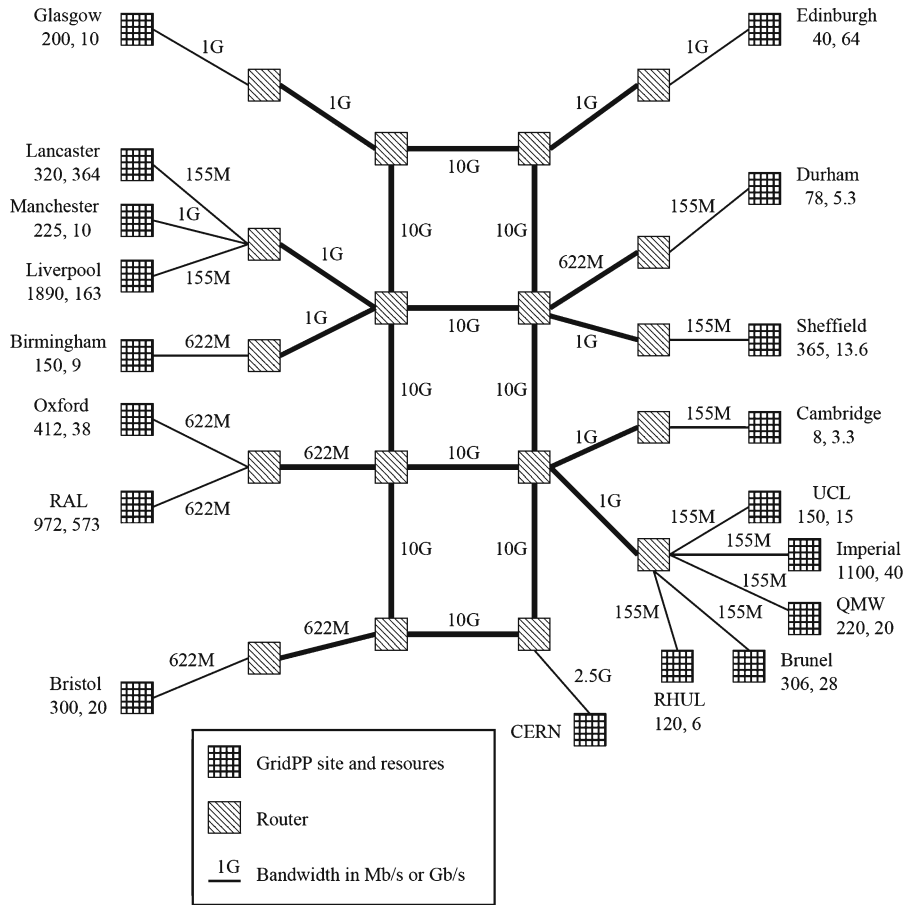

Fig. 5 GridPP topology. In all simulations the data source is the CERN node (source: [18])

Figures 6, 7 and 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

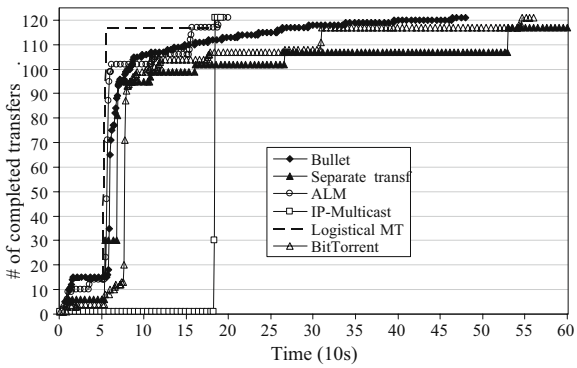

Fig. 6 Number of destinations that have completed the file transfer for the original LCG topology (separate transfer technique finishes the transfer at 730 s, not presented in the plot for better readability)

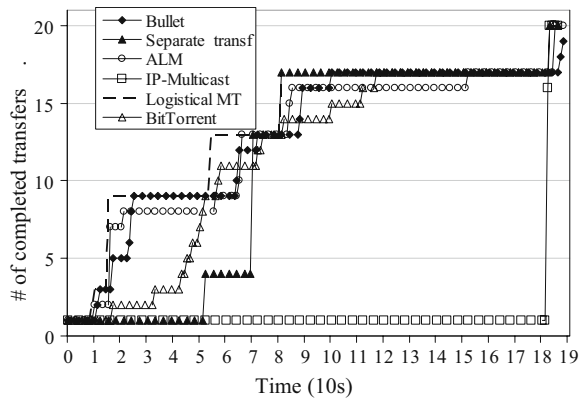

Fig. 7 Number of destinations that have completed the file transfer for the original EGEE topology

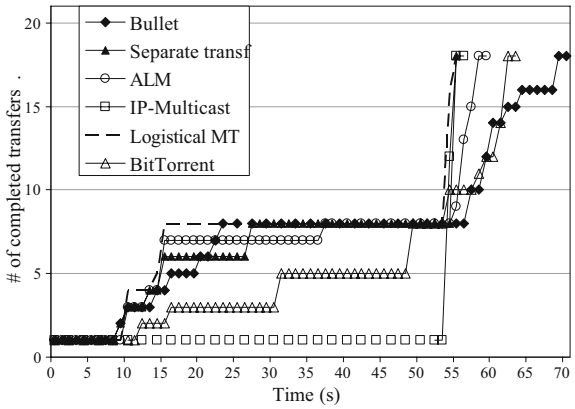

Fig. 8 Number of destinations that have completed the file transfer for the original GridPP topology

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 equivalent to IP-multicast for these three topologies.

IP-multicast provides the worst intermediate progress. 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 its 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.

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.

Figures 9, 10 and 11, present the time to complete the transfer to 50%, 90% and all destinations for the LCG, EGEE, and GridPP topologies, respectively, with different core link bandwidth. We summarize our observations below.

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.

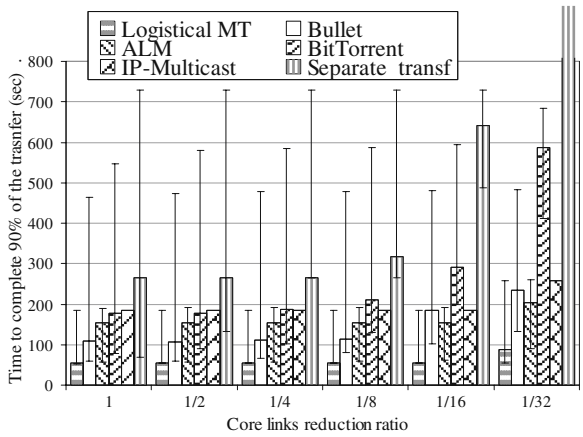

Fig. 9 Time to finish the transfer to 90% of the nodes for the original LCG topology and the topology with reduced core bandwidth. The lower error bar indicates the time to complete the transfer for 50% of the nodes while the top error bar indicates the time to complete the transfer for the last node. Separate transfer finishes in 2,280 s on the topology with core bandwidth reduced to 1/32 (not presented here for clarity)

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, although Bullet and BitTorrent offer good intermediate progress by exploiting orthogonal bandwidth, their dissemination completion time is worse than that of tree-based solutions. The reason is that, as we

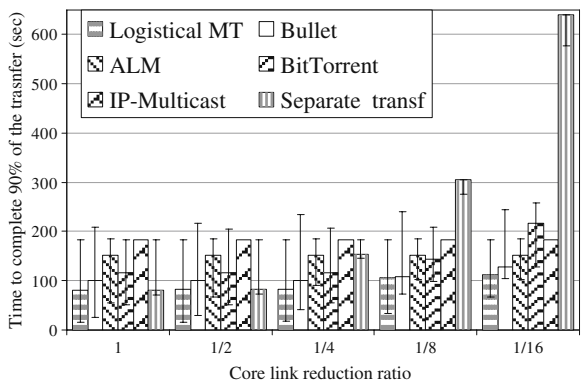

Fig. 10 Time to finish the transfer to 50%, 90%, and all nodes for the EGEE topology-original and reduced core bandwidth

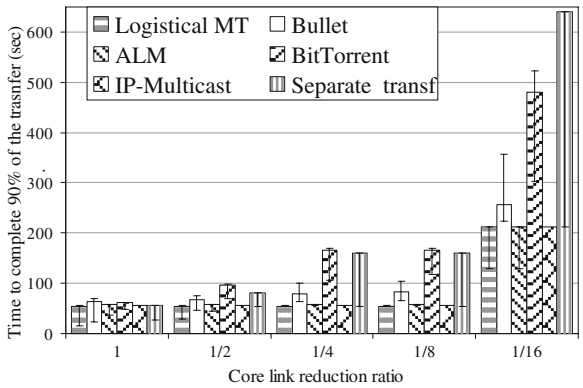

Fig. 11 Time to finish the transfer to 50%, 90%, and all nodes for the GridPP topology-original and reduced core bandwidth

demonstrate in Section 5.3, these algorithms generate higher network traffic overheads, and on constrained networks, these overheads lead lower dissemination performance.

To further investigate the ability to exploit alternate network paths, we have generated two sets of topologies in which the aggregate core bandwidth is maintained constant but the number of core links is changed. 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 Fig. 12, BitTorrent perfor-

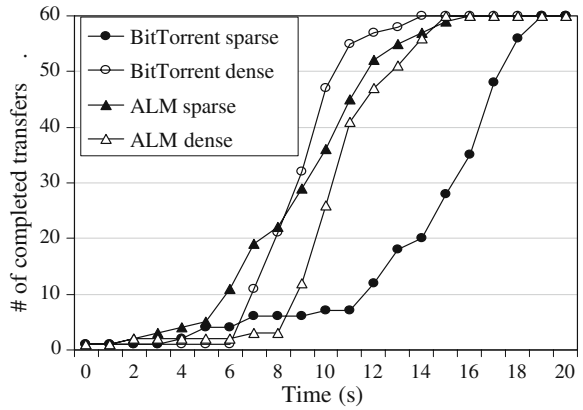

Fig. 12 Number of destinations that have completed the file transfer with two generated topologies. The dense topology has four times more links in the core with four times less average bandwidth per link

mance is better with more links in the core while ALM performance slightly degrades. These results underline BitTorrent's 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. Since Bullet and BitTorrent generate higher 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, such as high churn conditions specific to peer-to-peer systems.
- Bullet and BitTorent are more efficient in exploiting the orthogonal bandwidth available between the participating nodes, thus better able to cope with different topologies and adapt to dynamically changing workloads.

5.3 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 the data dissemination process.

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 shows the generated traffic (labeled as useful or duplicate) 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.

The following observations can be made based on Fig. 13 (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 generated 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.

When 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.

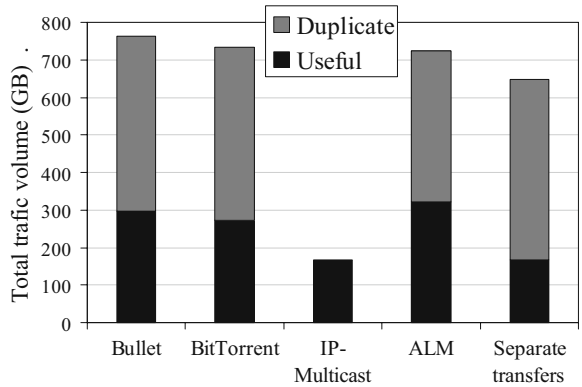

Fig. 13 Overhead for each protocol on the LCG topology

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 share in the generated traffic grows with the size of the topology. For example, while for the EGEE 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: their 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 on the same physical link or is sent multiple times through the core through different network paths.

5.4 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 by taking over all work performed by end-nodes while using the other application-level techniques.

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

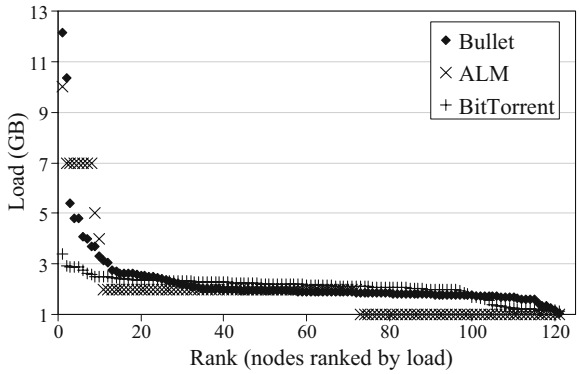

Fig. 14 Load balancing for ALM, BitTorrent and Bullet. Nodes are ranked in decreasing order of their load (LCG topology)

Figure 14 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 balance 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.

5.5 Fairness to Competing Traffic

While all the application layer dissemination schemes 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. Some of these dissemination schemes generate a large number of network flows that are sometimes mapped randomly over the

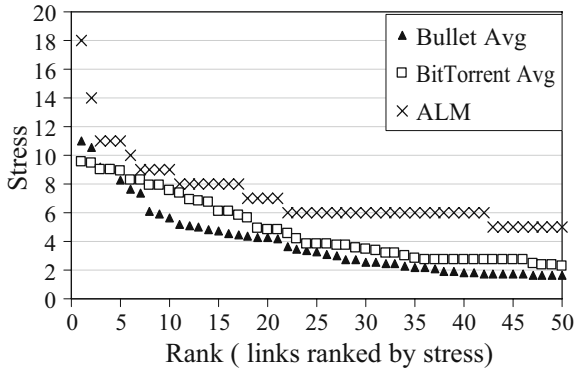

Fig. 15 Average link stress distribution for BitTorrent, Bullet and ALM over the LCG topology. The plot presents average link stress for the most stressed 50 links

network topology, and hence they have the potential to 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 to 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 definition of fairness. Even for IP-multicast, although fairness has been exhaustively studied [53], 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 point to point flow (e.g., 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 point to point flows, in order not to penalize competing unicast flows that share a portion of the path with the multicast session.

Different data dissemination solutions that work at the application-layer 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 [54], 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 of $(n-1) \times (n-2)$ flows 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 Fig. 15 presenting the *average* link stress distribution for the LCG 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, Fig. 16 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 dissemination trees.

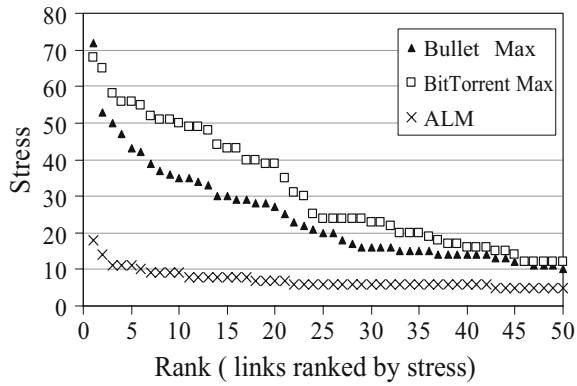

Fig. 16 Maximum link stress distribution of BitTorrent and Bullet over the LCG topology. The plot presents maximum link stress for the most stressed 50 links

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.

Of course, this is not a phenomenon particular to Grids. 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 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.

5.6 The Effect of Bullet and BitTorrent Configurations

The number of 'peering relationships', that is, 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 found that, for our topologies, configuring Bullet with two peers and BitTorrent with four peers in EGEE and GridPP topologies 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 using more than the default four peers (as in our experiments) slightly improves load balance 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 EGEE 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.

6 Summary

This study focuses on the problem of disseminating large data volumes 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?

Table 2 summarizes, for each data dissemination solution studied, its characteristics: that is,

Table 2 Summary of results

	Solution criteria	IP-multicast	SPIDER	LM	Naïve solution	ALM	Bullet	Bit-torrent
Characteristics	Data staging	No	No	Yes	No	Yes	Yes	Yes
	Data partitioning	No	Yes	No	No	No	Yes	Yes
	Orthogonal bandwidth exploitation	No	Limited by the number of created trees	No	No	No	Yes	Yes
	Uses physical topology information	Yes	Yes	Yes	No	Appl.-level bandwidth info. only	No	No
	Deployment effort	Medium	High	High	None	None	None	None
Performance	Dissemination finish time	Fast	Fast	Fast	Slow	Fast	Depends on the topology	
	Dissemination intermediate progress	Bad	Bad	Good	Bad	Good	Good	Good
	Overhead	Low (optimal)	Low (optimal)	Low (optimal)	High	High	High	High
	End-node load balance	Good (optimal)	Good (optimal)	Good (optimal)	Bad	Medium	Medium	Medium
	Fairness (impact on competing traffic)	Low	Low	Low	Highest around the source	High	High	High

Deployment effort estimates the amount of changes in the network infrastructure needed to support the mechanism under consideration

which approach to accelerate data dissemination is used, whether the solution uses physical network topology information to make decisions, and gauges the complexity of the changes required to the networking infrastructure to deploy the solution in practice. The table also summarizes the performance characteristics of the solutions studied in terms of the four success metrics described in Section 3.3: transfer time, generated overhead, end-node load balancing, and fairness to competing traffic.

In summary, our simulation-based investigation of 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 balance, 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 property which might be desirable in some scenarios.
- The naive solution of individual 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 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 [55], 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 reduce costs while building/provisioning collaborations, but also derive improved performance from deployed networks.

References

1. The Large Hardon Collider. http://lhc.web.cern.ch/lhc/. Accessed 2008
2. The Spallation Neutron Source. http://www.sns.gov/. Accessed 2008
3. The D0 Experiment, Fermi National Laboratory. http://www-d0.fnal.gov. Accessed 2008
4. The TeraGrid: a primer. http://www.teragrid.org (2004). Accessed 2008
5. Brown, M.: Blueprint for the future of high-performance networking. Commun. ACM **46**(11), 30–77 (2003)
6. Allcock, W., Chervenak, A., Foster, I., Kesselman, C., et al.: Protocols and services for distributed data-intensive science. In: Advanced Computing and Analysis Techniques in Physics Research (ACAT), AIP Conference Proceedings (2000)
7. Bassi, A., Beck, M., Moore, T., Plank, J.S., et al.: The internet backplane protocol: a study in resource sharing. Future Gener. Comput. Syst. **19**(4), 551–561 (2003)
8. Terekhov, I., Pordes, R., White, V., Lueking, L., et al.: Distributed data access and resource management in the D0 SAM system. In: IEEE International Symposium on High Performance Distributed Computing (2001)
9. Wang, F., Xin, Q., Hong, B., Brandt, S.A., et al.: File system workload analysis for large scientific computing applications. In: NASA/IEEE Conference on Mass Storage Systems and Technologies (MSST 2004) (2004)
10. Iamnitchi, A., Ripeanu, M., Foster I.: Small-world file-sharing communities. In: Infocom 2004, Hong Kong (2004)
11. Cohen, B.: BitTorrent web site. http://www.bittorrent.com. Accessed 2008
12. Kostic, D., Rodriguez, A., Albrecht, J., Vahdat A.: Bullet: high bandwidth data dissemination using an overlay mesh. In: SOSP'03, Lake George, NY (2003)
13. Guo, L., Chen, S., Xiao, Z., Tan, E., et al.: Measurements, analysis and modeling of BitTorrent-like

systems. In: ACM SIGCOMM Internet Measurement Conference, New Orleans, LA (2005)
14. Bharambe, A.R., Herley, C., Padmanabhan, V.N.: Analysing and improving a BitTorrent network's performance mechanisms. In: The 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2006), Barcelona, Spain (2006)
15. Plaza, A., Valencia, D., Plaza, J., Martinez, P.: Commodity cluster-based parallel processing of hyperspectral imagery. J. Parallel Distrib. Comput. **66**(3), 345–358 (2006)
16. Doyle, A.T., Nicholson, C.: Grid data management: simulations of LCG 2008. In: Computing in High Energy and Nuclear Physics, CHEP'06, Mumbai, India (2006)
17. Cameron, D.G., Millar, A.P., Nicholson, C., Carvajal-Schiaffino, R., et al.: Analysis of scheduling and replica optimisation strategies for data Grids using OptorSim. J. Grid Comput. **2**(1), 57–69 (2004)
18. Britton, D., Cass, A.J., Clarke, P.E.L., Coles, J.C., et al.: GridPP: meeting the particle physics computing challenge. In: UK e-Science All Hands Conference (2005)
19. Iamnitchi, A., Doraimani, S., Garzoglio, G.: Filecules in high-energy physics: characteristics and impact on resource management. In: HPDC 2006, France (2006)
20. Gummadi, K.P., Dunn, R.J., Saroiu, S., Gribble, S.D., et al.: Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In: SOSP'03, Lake George, NY (2003)
21. Bellissimo, A., Shenoy, P., Levine, B.N.: Exploring the Use of BitTorrent as the Basis for a Large Trace Repository, University of Massachuttes–Amherst
22. Williamson, B.: Developing IP Multicast Networks, vol. I. Cisco Press 592 (2008)
23. Chu, Y.-h., Rao, S.G., Seshan, S., Zhang, H.: A case for end system multicast. IEEE J. Sel. Areas Commun. **20**(8), 1489–1499 (2002)
24. Diot, C., Levine, B.N., Lyles, B., Kassem, H., et al.: Deployment issues for the IP multicast service and architecture. IEEE Netw. **14**(1), 77–88 (2000)
25. Touch, J.D.: Overlay networks. Comput. Netw. **36**(2001), 115–116 (2001)
26. Wolski, R.: Forecasting network performance to support dynamic scheduling using the network weather service. In: Proc. 6th IEEE Symp. on High Performance Distributed Computing, Portland, Oregon (1997)
27. Vazhkudai, S., Schopf, J., Foster, I.: Predicting the performance of wide-area data transfers. In: 16th International Parallel and Distributed Processing Symposium (IPDPS 2002). Fort Lauderdale, FL (2002)
28. Vazhkudai, S., Tuecke, S., Foster, I.: Replica selection in the globus data Grid. In: IEEE International Conference on Cluster Computing and the Grid (CCGRID2001), Brisbane, Australia (2001)
29. Beck, M., Moore, T., Plank, J.S., Swany M.: Logistical networking: sharing more than the wires. In: Active Middleware Services Workshop, Norwell, MA (2000)
30. Ganguly, S., Saxena, A., Bhatnagar, S., Banerjee, S., et al.: Fast replication in content distribution overlays. In: IEEE INFOCOM, Miami, FL (2005)
31. Byers, J.W., Luby, M., Mitzenmacher, M., Rege, A.: A digital fountain approach to reliable distribution of bulk data. In: SIGCOM (1998)
32. Byers, J., Considine, J., Mitzenmacher, M., Rost, S.: Informed content delivery across adaptive overlay networks. In: SIGCOMM2002, Pittsburg, PA (2002)
33. Pendarakis, D., Shi, S., Verma, D., Waldvogel, M.: ALMI: an application level multicast infrastructure. In: USITS'01 (2001)
34. Jannotti, J., Gifford, D.K., Johnson, K.L., Kaashoek, M.F., et al.: Overcast: reliable multicasting with an overlay network. In: 4th Symposium on Operating Systems Design and Implementation (OSDI 2000), San Diego, California (2000)
35. Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., et al.: OMNI: an efficient overlay multicast infrastructure for real-time applications. Comput. Netw. **50**(6) (2006)
36. Ratnasamy, S., Handley, M., Karp, R.M., Shenker, S.: Application-level multicast using content-addressable networks. In: Third International COST264 Workshop on Networked Group Communication (2001)
37. Castro, M., Druschel, P., Kermarrec, A.-M., Rowstron, A.: Scribe: a large-scale and decentralized application-level multicast infrastructure. IEEE J. Sel. Areas Commun. **20**(8) (2002)
38. Ripeanu, M., Iamnitchi, A., Foster, I., Rogers, A.: In Search of Simplicity: a Self-organizing Group Communication Overlay. University of British Columbia, Vancouver (2007)
39. Banerjee, S., Bhattacharjee, B., Kommareddy, C.: Scalable application layer multicast. In: SIGCOMM2002, Pittsburgh, PA (2002)
40. Das, S., Nandan, A., Parker, M.G., Pau, G., 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), FL, USA (2005)
41. Burger, M.d., Kielmann, T.: MOB: zero-configuration high-throughput multicasting for Grid applications. In: 16th International Symposium on High Performance Distributed Computing (HPDC), California, USA (2007)
42. Al-Kiswany, S., Ripeanu, M., Iamnitchi, A., Vazhkudai, S.: Are P2P data-dissemination techniques viable in today's data intensive scientific collaborations?, Technical Report, NetSysLab-TR-2007-01, University of British Columbia (2007)
43. Izmailov, R., Ganguly, S.: Fast parallel file replication in data Grid. In: Future of Grid Data Environments Workshop, GGF-10. Berlin, Germany (2004)
44. Garg, N., Khandekar, R., Kunal, K., Pandit, V.: Bandwidth maximization in multicasting. In: European Symposium on Algorithms. Budapest (2003)
45. The Network Simulator—ns-2. http://www.isi.edu/nsnam/ns/. Accessed 2008

46. Vahdat, A., Yocum, K., Walsh, K., Mahadevan, P., et al: Scalability and accuracy in a large-scale network emulator. In: OSDI (2002)
47. White, B., Lepreau, J., Stoller, L., Ricci, R., et al.: An integrated experimental environment for distributed systems and networks. In: OSDI, Boston, MA (2002)
48. Huang, P., Estrin, D., Heidemann, J.: 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, Montreal, Canada (1998)
49. Chun, B., Culler, D., Roscoe, T., Bavier, A., et al.: PlanetLab: an overlay testbed for broad-coverage services. ACM Comput. Commun. Rev. **33**(3) (2003)
50. Medina, A., Lakhina, A., Matta, I., Byers, J.: BRITE: an approach to universal topology generation. In: International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS '01, Cincinnati, Ohio (2001)
51. Gkantsidis, C., Rodriguez, P.R.: Network coding for large scale content distribution. In: 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005), Miami, FL (2005)
52. Azureus. http://azureus.sourceforge.net/. Accessed 2008
53. Yang, Y.R., Lam, S.S.: Internet multicast congestion control: a survey. In: ICT 2000, Acapulco, Mexico (2000)
54. Cherkasova, L., Lee, J.: FastReplica: efficient large file distribution within content delivery networks. In: Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems, Seattle, Washington (2003)
55. Open Science Grid. http://www.opensciencegrid.org/. Accessed 2008