CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2009)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1410

Reliability in grid computing systems[‡]

Christopher Dabrowski[*,†]

National Institute of Standards and Technology, 100 Bureau Drive, Stop 8970, Gaithersburg, MD 20899-8970, U.S.A.

SUMMARY

In recent years, grid technology has emerged as an important tool for solving compute-intensive problems within the scientific community and in industry. To further the development and adoption of this technology, researchers and practitioners from different disciplines have collaborated to produce standard specifications for implementing large-scale, interoperable grid systems. The focus of this activity has been the Open Grid Forum, but other standards development organizations have also produced specifications that are used in grid systems. To date, these specifications have provided the basis for a growing number of operational grid systems used in scientific and industrial applications. However, if the growth of grid technology is to continue, it will be important that grid systems also provide high reliability. In particular, it will be critical to ensure that grid systems are reliable as they continue to grow in scale, exhibit greater dynamism, and become more heterogeneous in composition. Ensuring grid system reliability in turn requires that the specifications used to build these systems fully support reliable grid services. This study surveys work on grid reliability that has been done in recent years and reviews progress made toward achieving these goals. The survey identifies important issues and problems that researchers are working to overcome in order to develop reliability methods for large-scale, heterogeneous, dynamic environments. The survey also illuminates reliability issues relating to standard specifications used in grid systems, identifying existing specifications that may need to be evolved and areas where new specifications are needed to better support the reliability. Published in 2009 by John Wiley & Sons, Ltd.

Received 14 March 2008; Revised 21 August 2008; Accepted 1 November 2008

KEY WORDS: grid computing; grid computing system; reliability; fault tolerance; dependability

1. INTRODUCTION

In recent years, grid technology has emerged as an important tool for solving compute-intensive problems within the scientific community and in industry. To further the development and adoption

*Correspondence to: Christopher Dabrowski, National Institute of Standards and Technology, 100 Bureau Drive, Stop 8970, Gaithersburg, MD 20899-8970, U.S.A.
†E-mail: cdabrowski@nist.gov
‡This article is a U.S. Government work and is in the public domain in the U.S.A.

of this technology, researchers and practitioners from different disciplines have collaborated to produce standard specifications for creating large-scale, interoperable grid systems. The focus of this activity has been the Open Grid Forum (OGF) [1], but other standards development organizations have also produced specifications, such as [2,3], that are used in grid systems. To fully transition grid technology to operational use and to expand the range and scale of grid applications, grid systems must exhibit high reliability; i.e. they must be able to continuously provide correct service [4]. Moreover, it is important that the specifications used to build these systems fully support reliable grid services. With the increase in use of grid technology, achieving these goals will be made more difficult as grid systems become larger, more heterogeneous in composition, and more dynamic.

Given the newness of grid technology, it is somewhat understandable that, initially, work on grid systems reliability might be less extensive than efforts to develop basic capabilities. In recent years, the body of work on grid reliability produced by researchers and practitioners in academe and industry[§] has increased steadily. This study surveys this work and reviews progress made. The survey characterizes and distinguishes large-scale grid systems consisting of heterogeneous, dynamic computing resources from other kinds of distributed systems. The survey identifies important issues that researchers are working to resolve in order to develop reliability methods for such grid environments. In addition, the survey illuminates reliability issues relating to the standard specifications used in grid systems, identifying existing specifications that may need to be evolved to better support reliability and areas where new specifications are needed. The important topic of metrics for measuring reliability in grid systems is also covered.

The survey shows that currently deployed commercial and research grid systems can and do behave reliably at present levels of scale using available technology. However, efforts are in progress to develop reliability methods for grid environments that are expected to have increased scale, heterogeneity, and dynamism. These efforts have focused on the following distinct functional areas of grid systems:

- Reliability of computational hardware, software, and data resources that comprise the grid and provide the means to execute user applications;
- Reliability capabilities initiated by end users from within applications they submit to the grid for execution;
- Reliability of infrastructure and management services that perform essential functions necessary for grid systems to operate, such as resource allocation and scheduling; and
- Reliability of grid networks for messaging and data transport.

An important contribution of this paper is to show that while there has been substantial progress in these functional areas, scalable reliability solutions for grid systems remain largely experimental. In each area, different research problems exist that must yet be solved and key specifications need to be developed or extended to better support reliability. The paper also shows that thus far ensuring reliability has centered on providing *fault tolerance*—defined as the ability to ensure continuity

[§]Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is neither intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

of service in the presence of faults, or events that cause a system to operate erroneously [4]. The emphasis on fault tolerance is partly due to the conditions in grid system environments, in which failures are likely. It is also partly due to the existence of redundant resources in grid systems, which allow substitution of properly functioning resources when failures occur. In contrast, there has been less focus on developing testing methods to find and remove faults in grid systems. Similarly, there has been less effort on creating metrics for measuring grid system reliability. With the gradual growth in scale of grid systems, researchers are also beginning to realize the importance of system-level approaches for improving reliability that considers techniques such as complex systems analysis.

To date, there has not yet been a comprehensive survey of work on grid reliability. Given the importance of this topic for the progress of grid technology and the extent of recent efforts, it is an appropriate time for such a survey to be conducted. The remainder of this paper describes reliability research in the four functional areas identified above and discusses specific issues and problems faced in each area. Section 2 characterizes large-scale, heterogeneous, dynamic grid computing environments, and the challenges involved in ensuring reliability of grid systems. This section also identifies other areas of distributed systems that provide a foundation for, and continue to influence, work on grid reliability. Section 3 surveys research on improving reliability of grid computing resources, including fault detection, recovery, and removal through testing. Section 4 discusses work on providing fault tolerance from within user applications running in grid systems, focusing on applications logically organized into workflows. Section 5 addresses reliability work in core infrastructure and management services. Section 6 discusses reliability of grid networks. Section 7 discusses approaches for analysis of grid reliability from an overall system-level perspective including reliability metrics. Section 8 summarizes findings and concludes. In the appendix, the references used in this paper are categorized by subject area. Finally, it is necessary to briefly discuss the scope of this study, which in addition to grid reliability, includes the closely related topic of *availability*, or readiness to provide correct service [4]. Both belong to the larger topic of *dependability*, which also comprises of safety, integrity, and maintainability [4]. Space limitations prevent covering these other areas of dependability, or to include the critical and extensive subject of grid security. For the same reason, computer hardware reliability and physical site integrity are also omitted.

2. THE CHALLENGE OF ENSURING RELIABILITY IN GRID SYSTEMS

In [5–7], a vision of grid systems was articulated, in which computing and data resources belonging to many enterprisers are organized into a single, virtual computing entity that can be transparently utilized to solve compute- and data-intensive problems. Subsequently, this vision has continued to evolve as use of grid technology has grown within industry and science. To realize the long-term goals of grid computing will require development of methods that ensure that grid services are reliably provided under conditions of scale, heterogeneity, and dynamism (also sometimes called dynamicity). Large scale in computing and data resources is already in evidence in some systems, such as [8,9]. These systems contain in the order of 10^5 servers, making them large scale by today's reckoning (though systems with fewer servers and sufficiently heavy workload may also be seen as large scale). Over time, as grid systems grow, the threshold for what is considered large scale will shift upward. In addition to scale, coordination of grid system resource use will be made difficult by the wide distribution and heterogeneous nature of computing, data, and network resources. Grid resources will belong to many different enterprisers that may have little knowledge of each other, rather than to a single organization or small group of cooperating organizations. As such, they will be managed in different administrative domains, rather than belonging to one, centrally managed domain. Different domains are likely to have dissimilar access and security policies, while the resources they manage will employ different processor architectures and operating systems. Further, the environment in which these resources exist will be highly dynamic, due to the combined effects of enterprises continuously joining and leaving the grid, administrative policy changes, and component upgrades.

Even with standard interfaces and communications protocols in place, resource heterogeneity and dynamism will likely lead to component interactions that result in faults and failures, which imperil executing user applications. Some faults encountered in current grid systems may prove hard to detect [10,11] or wreak havoc by propagating through the grid [12]. Long-running applications that require many resources and must produce precise results are likely to be especially vulnerable [13]. Another potential source of faults will come from grid network services that transport large data sets. To transfer large amounts of data, network services will need to coordinate many heterogeneous network components and maintain stable high-bandwidth connections for long periods [14], which also increases the chance of faults. Another complicating factor will be the asynchronous nature of this environment, in which distributed components utilize independent clocks and messages may be subject to unbounded delay. As a result, it will be more difficult for distributed components to coordinate their computations or to know if a component has failed or is just responding slowly [15]. The need to manage large numbers of computational, data, and network resources under such conditions of scale, heterogeneity, and dynamism distinguishes grid systems from other types of distributed systems. As others have argued [16], these differences motivate development of reliability methods that are designed specifically for the conditions that prevail in grid environments.

Despite these distinguishing characteristics, methods for ensuring reliability of grid systems are closely related to, and partly based on, reliability methods developed in other branches of distributed systems research. Reliability work in other areas of distributed systems has a long and rich history, as evidenced by past work on wide-area networks [17–19], high-performance cluster computing [20–22], and distributed database systems [23]. Also important for grid systems is previous work on quantitative estimation algorithms that measure reliability in distributed systems [24,25]. Peer-to-peer networks also influence grid systems, but because this is also a new technology, reliability has been less extensively researched [26]. Another emerging area likely to influence grid computing is *cloud computing*, which provides simplified interfaces for accessing virtual mass computing services that are tailored to narrowly defined application domains [27]. Though there has yet been a little opportunity to explore if cloud system designs can foster reliability, on-demand cloud services are now being offered with transparent fault tolerance [28]. Reliability methods from these areas of distributed computing provide a basis for, and influence, grid reliability research. Where appropriate, this study discusses these relationships.

3. RELIABILITY OF GRID RESOURCES

Because of its obvious importance, more effort has been directed toward ensuring the reliability of computing resources that comprise grid systems than the other functional areas identified above.

Grid resources include processor clusters, supercomputers, storage devices, and related hardware, together with operating system and other software for managing these resources. Grid resources also include software components that perform analysis functions such as data mining. Another category of grid resources is data stores used in data and multimedia grids.

To date, developing methods to ensure reliability of grid resources in both research and commercial systems have largely meant developing methods for fault tolerance. Fault tolerance consists of: (1) detecting faults and failures in grid resources and (2) recovery to allow computations to continue. As with reliability in general, fault-tolerance methods used in today's operational commercial and science grid systems are based on distributed systems technology. When applied by capable engineers, these methods can and do provide fault tolerance for systems at current scale in a single enterprise [8]. However, in anticipation of the growth in grid systems, researchers are developing fault-tolerance methods that extend current technology to operate in conditions of greater scale, heterogeneity, and dynamism. Work on test methods for fault removal, though less extensive, is also covered.

3.1. Detecting faults and failures in grid resources

Building on previous work on fault and failure detection for distributed systems [29–31], researchers have investigated scalable fault detection methods for grid systems environments. Work has also focused on fault isolation and diagnosis techniques for recognizing different types of faults. Another area of concern involves hard-to-detect faults that are expected to occur in large-scale, heterogeneous, dynamic grids. However, most failure detection methods developed so far remain experimental.

Limitations of current methods for fault detection. Failure detection methods developed for current distributed systems have generally not been regarded as suitable for large-scale, heterogeneous, dynamic grid systems. One reason is that available network monitoring protocols and tools, such as those based on SNMP [32], rely on detailed knowledge of network structure. Such information is less likely to be available in large-scale, dynamic environments having multiple administrative and security domains. Failure detection in grid systems can also be compromised by another well-known problem that occurs in asynchronous distributed systems in which management functions (including failure detection) are decentralized and subject to failure. In such systems, the work of Fischer *et al.* [33] and Barborak and Malek [15] showed that it is impossible for a group of distributed failure detectors to reach consensus deterministically on what resources have failed if any component involved in the detection process also fails. Building on [15, 33], Chandra and Toueg [34] first characterized failure detection in distributed systems in terms of properties of *completeness*, the ability to detect all components that have failed, and *accuracy*, the ability to avoid mistakes. Assuming an asynchronous distributed environment, they showed that a group of failure detectors, some of which may fail or make errors, could reach consensus using deterministic procedures, but at the cost of delaying fulfillment of the completeness property and achieving only partial accuracy.

The work of [15,33,34] led to development of experimental failure detectors that sought to guarantee completeness, but were only probabilistically accurate [31]. These systems detected failure deterministically, using heartbeat techniques in which resources regularly sent messages (heartbeats) to other members of a heartbeat group. If heartbeats were absent, group members then used a consensus procedure to determine which members had failed. However in [31], failure detectors that used this approach were shown not to scale in worst-case network load scenarios, while alternative approaches that incorporated centralized monitoring were subject to message bottlenecks and other anomalies that degraded performance. In [35], Hayashibara *et al.*, looked at other factors related to scalability. They concluded that failure detection methods used in current distributed systems would not be effective in the face of the message explosion, dynamic resource composition, and variability of user applications that are expected in grid environments. Early research grid systems also reflect the lack of scalable fault detection. Zanikolas and Sakellariou [36] conducted a survey of 19 grid monitoring systems based on the OGF *Grid Monitoring Architecture* [37]. This study concluded that most systems did not have the potential to scale their monitoring functions, thus impeding scalability of fault detectors that relied on them. The lack of failure detection methods suitable for grid environments has thus motivated the work described below.

Research on scalable fault detection methods. An early distributed fault detector for the Globus system [38] addressed issues of completeness and accuracy [34] by making estimates of the likelihood of resource failure. User applications could obtain these estimates and interpret them at their own discretion. The approach was designed to improve efficiency and scalability of fault detection by decoupling this function from the monitoring function. This work was followed by other experimental failure detectors that built on [31,34], examples of which are discussed here. In [39], a failure detector was proposed that sought to preserve completeness and achieve scalability by organizing grid resources into heartbeat groups on the basis of the logical network topology reflected in their Internet addresses. Leader nodes, or monitors to which resources sent heartbeats, were made redundant for fault tolerance. The total number of heartbeats required to monitor all resources was shown to scale with a computational complexity of $O(n)$, where n is the number of heartbeat groups. The viability of this approach was demonstrated in experiments in a testbed which simulated 144 processor nodes. Horita *et al.* [40] proposed a scalable, self-organizing fault detection system based on earlier work on using group membership protocols for fault detection [31,41]. In this approach, each process was monitored by a small group (4 or 5) of randomly chosen processes on remote nodes. The monitoring processes established a Transmission Control Protocol (TCP) connection to the monitored process and periodically transmitted short messages to check if the connection is alive. This resulted in creation of a virtual monitoring subnetwork within a grid, consisting of heterogeneous resource types, through which notifications of connection failure could be propagated. Experiments showed scalability for a system of three-node clusters (or grid sites) that contained 300 resources. In [42], resources were organized into separate domains in which they emitted heartbeats to a domain monitor; here, monitoring domains were structured hierarchically for scalability.

Detecting different types of faults. An important aspect of providing completeness and accuracy is recognizing different types of faults that lead to failures. Examples of fault-type taxonomies related to grid system environments were described in [43–45]. Early work on a fault-handling framework that could distinguish between different kinds of failures during simulated grid operations was reported in [16]. This system also initiated recovery actions designed to remedy different fault types. Jitsumoto *et al.* [46] developed a detector that differentiated between hardware, process, and transmission faults. Here, users were allowed to preselect a recovery procedure to be invoked in response to occurrence of a particular fault type. This approach was found to exhibit good performance in a 32-node cluster testbed. In [47], a method was proposed to detect and predict different faults types using a fault classification scheme and data mining. In [48], a fault detection and recovery method for transient process faults were reported. Here, an adaptive scheme was used to periodically checkpoint (i.e. store the state of) replicated processes that executed in parallel. Checkpointed process states were then compared with discover erroneous computations affected by transient faults. Because this procedure was computationally intensive, the checkpoint interval was dynamically varied in response to the observed frequency of faults to improve its efficiency. Jin *et al.* [49] developed a hierarchical grid failure detector and failure handler that adapted to changing user requirements and system conditions. Testbed experiments showed that this approach scaled to up to 1000 components at two sites. Work on adaptive mechanisms for detecting different fault types has also been done in the European Union Datagrid Project [50]. In [51], long-term fault patterns were studied in a grid testbed provided by the Pacific Rim Application and Grid Middleware Assembly.

Hard-to-detect faults. In scaled, heterogeneous grid environments, some types of faults may be difficult to find. Kola *et al.* [13] reported work on 'silent' fault types that typically do not indicate their presence immediately after they occur. Work has also been reported on using consensus-based algorithms to detect Byzantine faults. Byzantine faults cannot easily be traced to failed links, processes, and messages. They originate, for example, when equipment periodically or randomly malfunctions due to aging, sabotage or external damage, or is subjected to transient events such as electromagnetic interference. In [10,11], the results of identical, simultaneously executing computations were compared to detect components affected by Byzantine faults. Finally, in large, heterogeneous distributed systems, faults may originate at one component and propagate to others, potentially creating dangerous cascading effects. Thus far, with the exception of [12,52], methods for isolating faults that propagate through grid systems have been little studied. Because hard-to-detect faults and cascading failures can lower user confidence, continuing efforts to develop scalable methods for isolation and detection of these types of faults are essential.

3.2. Research in recovery methods for grid resources

As in distributed systems generally, recovery methods in grid systems rely on exploitation of redundancy. There are two forms of redundancy to consider. Temporal redundancy involves repeated attempts to restart failed resources or services. Spatial redundancy attempts to take advantage of multiple copies of computing resources. Both temporal and spatial redundancy is used in grids. However, because grid systems inherently provide redundant computing resources, spatial redundancy has been a focus of fault-tolerance research and is therefore the main topic of this section. Three techniques that emphasize spatial redundancy exist: (1) *checkpointing*, or periodically saving the state of a process running on a computing resource so that, in the event of resource failure, it can resume on a different resource; (2) *replication*, or maintaining a sufficient number of replicas, or copies, of a process executing in parallel on different resources so that at least one replica succeeds; and (3) *rescheduling*, or finding different resources to rerun failed tasks. Note that (1) and (3) involve operations that repeat over time and are therefore temporally redundant as well. This section also reviews work on data replication, which uses spatial redundancy to provide fault tolerance in data grids.

3.2.1. Checkpoint and recovery

Taking checkpoints is the process of periodically saving the state of a running process to durable storage. Checkpointing allows a process that fails to be restarted from the point its state was last saved, or its *checkpoint*. If the host processor has not failed, temporal redundancy can be used to *roll back* and restart the process on the same platform. As in other systems, this method is widely used in grids [16,46,49]. Otherwise, if the host has failed, the process may be *migrated*, or transferred, to a different execution environment where it can be restarted from a checkpoint (a technique also referred to as *failover*). This section begins by discussing checkpoint and process migration methods used in commercial and science grid systems that are based on methods used in high-performance cluster computing. This is followed by discussion of new methods being developed or adapted for scaled grid environments, together with related issues that need to be resolved. Most notable is the issue of finding efficient methods for checkpointing many concurrent, intercommunicating processes, so that in the event of failure, they can resume from a common saved state [22]. Checkpointing can be initiated either from within grid systems or within applications. This section considers the former. Section 4 discusses application checkpointing.

Checkpointing and recovery in current grid systems. Checkpoint and process migration methods have been long used in high-performance computing environments and a substantial body of work on this subject precedes grid computing [21,22]. Many currently deployed grid systems that manage computing clusters and run parallel processes employ methods based on high-performance cluster computing. Examples are commercial grid products assembled from cluster computing components such as [53–58]. These systems also provide recovery for server managers, or cluster head nodes, that manage concurrently executing processes. For instance, Reference [57] provides a fault-tolerant grid infrastructure for server managers and node clusters. If the manager fails, another node takes over the management function, while failure of a compute node results in restart of the checkpointed processes on another node. Despite repeated failures, individual clusters preserve a logical structure in which a manager continues to supervise the remaining compute nodes.

Research grids and grids used for science applications that manage clusters also employ checkpoint and process migration techniques based on high-performance computing. Early efforts in using checkpoint or process migration in large cluster environments were reported in the Legion [59], Cactus [60,61], and in Condor [62]. In the HA-OSCAR research grid system [63], fault tolerance in cluster head nodes was improved by taking checkpoints of job-queue information and regularly updating a backup server. If the primary server failed, the backup could access up-to-date job-queue information. Testbed experiments demonstrated faster restart of in-progress jobs using this approach.

Research in checkpointing methods. An important research issue for grid systems is the ability to provide efficient and scalable checkpointing of concurrent, intercommunicating processes whose states must be synchronized to ensure consistent recovery. Synchronization is required, because as these processes exchange messages, they cause changes to each other's internal states. If a checkpoint is taken when a message between two processes is in transit, the resulting saved states may be inconsistent. Three checkpointing strategies were described for concurrent processes in [22]. In *coordinated checkpointing*, processes synchronize checkpoints to ensure their saved states are consistent with each other, so that the overall combined, saved state is also consistent. Taking coordinated checkpoints requires correctly accounting for messages that are in transit when the checkpoint operation is initiated. In contrast, in *uncoordinated checkpointing*, processes schedule checkpoints independently at different times and do not account for messages. Here, attempts by processes that exchange messages to roll back and recover a previous common, consistent state may be subject to a repeating *domino effect*. The domino effect occurs when rollback of one process causes processes to which it had earlier sent messages to also roll back. If these processes also sent messages earlier, additional processes that received the messages may also then also have to roll back. As the effect continues, the processes involved resemble falling dominos as they repeatedly cause other processes to roll back. A third strategy, *communication-induced checkpointing* attempts to coordinate only selected critical checkpoints, but is thought to be inadequate for large-scale environments [64].

Coordinated checkpointing has thus far received the most attention as a recovery mechanism for concurrent, intercommunicating processes. The study by Elnozahy *et al.* [22] found that in a distributed systems environment, coordinated checkpointing procedures that blocked (halted) processes during the synchronized checkpointing operation degraded performance, while procedures that did not employ blocking exhibited better performance but complicated recovery. Much of the work on coordinated checkpointing has been done in connection with use of the Message Passing Interface (MPI) [65], a key specification for enabling communication between concurrent processes. Coordinated checkpointing has been implemented in MPI in [46,64,66–69]. For example in [67,68], Yeom *et al.*, proposed a fault-tolerant version of MPI, MPICH-GF, for grid systems that employed coordinated checkpoints with blocking. As in [22], Buntinas *et al.* [70] compared MPI-based, blocking, and non-blocking coordinated checkpoint protocols for efficiency in a grid testbed of over 500 processors at 9 sites in a wide-area network. The results indicated that blocking incurred higher overhead than the non-blocking protocol, while the scalability of both methods appeared to be uncertain for larger systems. Given the interest in integrating coordinated checkpointing with MPI, this specification should be reviewed to determine if extensions of this kind are needed to provide greater fault tolerance for grid systems. In addition, it will be important to determine the extent to which checkpointing methods can scale when the number of concurrent, intercommunicating processes exceeds levels reported in [70].

Message logging and process migration. As an alternative to coordinated checkpointing, uncoordinated checkpointing may be combined with *message logging* to achieve process state synchronization and avoid the domino effect. In logging, information about messages is logged, or stored, on stable media and later replayed to recover a lost state [71]. During recovery of a failed process, messages that occurred after a checkpoint are replayed to recreate a pre-failure state that is consistent with the states of other concurrent processes, rather than initiating rollbacks to find a consistent state. The effectiveness of different logging methods was surveyed in [22]. Use of uncoordinated checkpointing in combination with logging in MPI systems was reported in [64] and found to be less efficient than coordinated checkpointing in a 32-node cluster [72]. However, these two techniques have not yet been compared in grid environments under scaled or heterogeneous conditions.

Additional approaches to improving checkpointing in grid systems are possible. These include increasing fault tolerance by replicating checkpointed data on distributed repositories [73] and improving efficiency by determining optimal checkpoint intervals [74]. Finally, an important issue for recovery in grid systems is the migration of checkpointed processes to platforms with dissimilar execution environments and different administrative or security domains. Here, preliminary work [75,76] also precedes the emergence of grids. For grid systems, the problem was investigated in [77,78] and work in this area is likely to continue. As with checkpointing, the scalability of process migration methods in large, heterogeneous environments requires further investigation.

3.2.2. Grid resource replication

In grid resource replication, multiple grid resources simultaneously perform an identical computation and maintain identical state. The goal of replication is to ensure that at least one replica is always able to complete the computation in the event others fail. In some cases, one replica may be designated as a primary copy for purposes of external interaction, whereas others assume the role of backups. This section reviews resource replication methods developed for improving fault tolerance in large-scale grid systems. Two aspects of research in resource replication are considered: (1) algorithms for determining optimal (or near-optimal) placement of replicas in order to increase fault tolerance and (2) methods for synchronizing replica states to ensure their consistency. Both remain research problems, for which proposed solutions have been evaluated in limited-scale testbeds or in simulations. Under scaled conditions, replica synchronization can incur high overhead costs, and a comprehensive understanding is lacking of tradeoffs between increasing fault tolerance through replication versus synchronization overhead. Another issue is when to use checkpoint and process migration instead of resource replication and vice versa. This issue is considered in [46], but is generally not well understood. Overall, replication methods have been studied less than checkpointing and recovery. Yet, given the significant opportunities presented by resource redundancy in large-scale grid systems, replication warrants more investigation.

Replica selection and placement methods. An early attempt to evaluate fault tolerance and scalability properties of adaptive algorithms for replica placement services in dynamic, distributed systems was reported in [79]. Weissman and Lee [80] proposed a replica management system that dynamically allocates replicated resources on the basis of user demand. If individual replicas failed, this system then allowed user resource requests to be transparently rerouted. Testbed experiments demonstrated that response time scaled well for over 1000 requests at three sites. The Scalable Replication Infrastructure using Resilient Autonomic Meshes research system [81] was designed to improve resource availability and fault tolerance in grids and other distributed environments. Here, computing resources were members of networks, or meshes, which could be searched to find nodes on which grid processes could be replicated. Search of large meshes was made more efficient through organization of computing resources in a spanning tree structure and through intermediate caching of query results for reuse. The spanning tree automatically re-configured as nodes were added or removed, allowing the system to scale in response to changing conditions. Participating resources operated securely and anonymously, allowing the mesh to incorporate multiple administrative domains.

More recently, other replication schemes have been proposed. Through experiments, most have demonstrated limited scalability (involving 10^2 processing resources, or less). Valcarenghi and Piero [82] described a service replication approach in which replicas were located in proximity to each other to form *service islands* in a grid network. Different replica configurations were evaluated using a Mixed Integer Linear Programming model to determine which configuration of islands exhibits higher fault tolerance. Simulation showed that the approach could enable recovery of a high percentage of long-distance connections to remote services, while minimizing the number of replicas needed and thus simplifying replica management. In [83], dynamic process replication was used to provide fault tolerance and enable fulfillment of service level agreements (SLAs) [84] in a grid system developed by a telecommunications company. Similarly in other systems that use SLAs, such as in commercial cloud computing, resource replication has been used to provide transparent fault tolerance [28].Within the e-Demand project, Townend *et al.* [85] proposed a replication method that detected faulty computations among multiple, inter-communicating tasks in a grid. Genaud and Rattanapoka [86] developed a mechanism for MPI-based grid environments that used replication to increase fault tolerance of parallel computations and demonstrated scalability in a testbed having up to 128 processors. Other methods for replicating computations on resources have been proposed [87,88] and demonstrated experimentally.

Replica synchronization. The research literature indicates that less work has been done on efficient and scalable methods for synchronization of replica states. One method [82], based on selective replica placement, has been described above (see Section 3.2.2). In [89], this issue was investigated for service replicas that exhibit non-deterministic behavior and use asynchronous messaging. Here, the researchers proposed an optimized version of the Paxos algorithm [90] for synchronizing replicas in distributed environments and demonstrated efficiency in a wide-area testbed with four sites and up to 128 clients submitting transactions. In earlier work [91], a more traditional primary-backup approach was used to investigate replication of grid services that were implemented using Open Grid Services Infrastructure (OGSI) [92] and the Globus toolkit [93]. In [91], use of this strategy resulted in higher service availability in local area environments. However, the study found that significant overhead costs were imposed by OGSI notification when used to synchronize states of service replicas that behaved non-deterministically. The study showed that synchronization overhead could be eliminated by restarting failed tasks on replicated resources reserved for this purpose. To date, this work has not been repeated with successor specifications to OGSI. Dasgupta *et al.* [94] proposed a resource allocation framework in which redundant backup resources could be reserved for use in case the primary resource failed. Simulation showed circumstances where this approach improved efficiency of system resource allocation. If the potential of replication to provide fault tolerance is to be realized on a large scale, additional work on efficient and scalable synchronization methods will be necessary.

3.2.3. Rescheduling failed tasks

In addition to process migration and replication, a failed task can be dynamically rescheduled on different resources. This method uses existing grid resource allocation services for rescheduling, thus eliminating the overhead of checkpointing or replica synchronization. In [95,96], a prototype rescheduling mechanism was developed and tested in a production environment containing over 500 heterogeneous computing resources at 13 sites. Similarly in [97], a prototype was created that efficiently rescheduled failed jobs in a testbed with over 2000 heterogeneous platforms at 25 sites. Rescheduling appears to be a viable fault-tolerance tool to consider under some circumstances. However, rescheduling may not be appropriate for long-running processes and may adversely impact other processes that are executing concurrently with, or are dependent on, a rescheduled process. Nevertheless, the limited success seen so far indicates that rescheduling merits consideration as a fault-tolerance method.

3.2.4. Data replication

Replication of data sets has been a research topic of long standing in large-scale grid systems and has also been implemented commercially, as for example [98,99]. In [8], use of large-scale replicated data stores to promote fault tolerance across heterogeneous storage platforms at multiple sites was reported, while [28] described commercially available mass data storage services that employ replication to achieve fault tolerance. Redundant arrays of inexpensive disks, used to improve fault tolerance in distributed storage systems [100], have been adapted for data grids [101], including commercial examples [102,103]. However, scalable solutions that work in heterogeneous, dynamic grid environments are not yet commonplace. Within the research community, most experimental data replication methods have been developed to improve performance, with fewer efforts focusing on fault tolerance.

Early research on data grids demonstrated the potential benefits of data replication for performance, data availability, and fault tolerance [104–106]. Subsequently, other studies emphasized performance improvements obtained through data replication, including [107–109]. Studies that have focused more on fault tolerance include [110], where a quorum-based protocol was described for managing replicated data. Here, experimental results demonstrated fault tolerance by showing that data retrieval can succeed when as many as 75% of replicas failed. Lei *et al.* [111] used reliability metrics to evaluate three data replica placement optimization algorithms for improved data reliability. This study used simulations, in which up to 200 files were accessed by 10^5 jobs that required 3–20 files each.

A number of researchers investigated using decentralizing data replication management services to improve fault tolerance in data grids. In [112], Chervenak *et al.*, described a decentralized replica location service for the Globus toolkit [93] that was designed to be scalable and fault tolerant. Here, distributed, redundant indexes maintained information on data replicas in a consistent manner. Experimental testbed results documented improved performance for over 10^3 data operations per second. In [113], this work was extended to test performance using scientific data sets in wide-area environments and was later implemented in production science grids. In related work, Ripeanu and Foster [114] described a decentralized replica location service that was intended to achieve robustness through a redundant, distributed replica management service. However, experiments focused on system performance rather than fault tolerance. Zhang *et al.* [115] proposed an algorithm for dynamically locating data replica servers within a grid to optimize performance and improve fault tolerance, though scalability issues were not examined. Following the example of [113], more research on fault-tolerant data grids needs to be extended to large-scale environments. In addition, few of the studies discussed above addressed heterogeneous or dynamic environments.

3.3. Fault removal in software through testing and code certification

Software component testing to find and remove potential faults is a traditional method for improving component reliability. Components that have passed tests can then be certified as having achieved a level of reliability. Methods for testing and certifying grid components have received less attention from researchers than fault-tolerance methods. Nevertheless, the argument for software testing to remove faults as a precondition to fault tolerance is strong. The economic importance of adequate testing methods and tools is discussed in [116,117].

To date, experimental methods and tools have been developed that can be used to discover defects in grid system software. One such method is fault injection, also used earlier in distributed systems [118–120], as well as in software systems in general [121–123]. Looker *et al.* [124] reported preliminary work on use of fault injection to identify malfunctioning distributed software components. In [125], the work in [124] was extended to generate fault-injection tests using an ontology-based approach. Similarly, Reinecke *et al.* [126] used fault injection to analyze restart oracles in distributed systems. Hoarau and Tixeuil [127] studied the use of fault injection to discover grid system components that were susceptible to process failure faults. Monnet and Bertier [128] also developed a fault-injection tool to test the ability of fault detectors to find independent and correlated failures in scaled grid environments.

Beyond fault injection, other testing techniques have been explored. In [129], preliminary work was reported on designing methods for assessing the impact of periodic upgrades on the dependability of distributed commercial off-the-shelf software (COTS) product components. Song and colleagues [130,131] analyzed a well-known COTS, using component dependency graphs to identify crucial *hub* components through which a large portion of system messages flowed. Hub components with faults are more likely to adversely impact operation of a system and therefore should be prioritized for testing. Bitonti *et al.* [132] described a tool for sending test probes to legacy code components that were integrated into the Globus Monitoring and Discovery Service (MDS4) [93]. In [133], a patterned series of query and file-transfer test probes were used to measure robustness and stability of Globus-based grid systems. A different approach was taken in the *ConCert* project [134], where *certifying* compilers were used to produce machine code for execution on grid resources. The code contained checkable certificates that could be used to automatically verify code properties when the code was deployed. Initially used to verify code safety, the approach also showed promise for detecting faulty or malicious code [135].

These efforts represent a start toward developing testing technology for grids. An important step toward developing methods and tools for systematic testing is to first obtain a better understanding of cost–benefits of testing grid components. Such a study could be used to determine how grid system functions should be prioritized for testing and certification, what kind of tests would be most cost effective (component tests, integration tests, interaction tests, etc.), and how tests should be administered. Here, previous research in fault prediction in software components [136–139] provides a basis for developing procedures to identify likely areas to test in grid systems. In addition, an approach such as [134,135] could allow testing of already deployed components in a grid system that must be continuously online.

4. RELIABILITY PROVIDED BY GRID APPLICATIONS AND WORKFLOWS

Section 3 described fault-tolerance capabilities that may be provided to grid resources from within the grid. However in many cases, the grid system may provide these capabilities inconsistently or not at all. Therefore, from the application's point of view, there may be no guarantee that resources available in a grid will be reliable. For this reason, users are often motivated to design their applications with built-in fault tolerance. This is especially true if the application consists of multiple tasks organized into a workflow. In this case, workflow design languages and tools are used to specify the task execution order and the data flow between tasks. Once the workflow is defined, discovery and resource allocation services are used to assign grid resources to the workflow tasks. A workflow management service may then be used to supervise workflow execution. The workflow may be designed to have built-in fault tolerance. For example, a workflow manager may schedule tasks redundantly on replicated resources or, during execution, may dynamically reschedule tasks so that the workflow computation can continue if a task fails.

Fault tolerance in workflow management systems has been a research topic prior to the advent of grid systems [140–144]. As is the case with workflow management tools generally, grid workflow tools do not yet have well-developed fault-tolerance capabilities. Today, there is no standard specification for grid workflow design and management that allows definition of built-in fault-tolerance capabilities. As a step in this direction, the OGF is defining a checkpointing and recovery service for individual processes that can be initiated from a user application [145]. In what follows, current research on fault-tolerance capabilities for grid workflows is described. Then, the important issue is addressed of coordinating fault tolerance provided by workflows with fault-tolerance capabilities originating within the grid.

4.1. Fault-tolerance capabilities originating from within grid workflows

In recent years, there has been a significant amount of research on developing languages and tools for design and management of workflows in web service-based grid environments. From the standpoint of grid systems, these workflow languages and tools can be divided into: (1) those developed specifically for grid systems and (2) those developed for more generic distributed environments based on web service standards. Though tools in the second category do not provide grid-specific features, they have been used for grid applications.

Fault-tolerant capabilities provided by research workflow languages and tools for grid systems were described by Hwang and Kesselman in [146]. These capabilities facilitated recovery by manipulating the workflow structure to minimize the impact of real-time failures. Recovery actions could be specified in the workflow definition or initiated independently by the workflow manager. These included rescheduling failed tasks on slower but more reliable resources, replicating tasks on multiple resources, and executing user-defined exception-handling procedures. Yu and Buyya [147] surveyed 13 research workflow management tools for grid systems and identified tools that supported the workflow recovery actions described in [146]. These included References [148–155], and also Xiang *et al.* [48] proposed an adaptive checkpoint and recovery scheme for grid workflows (see Section 3.1). However, Yu and Buyya found overall that 'most fault handling techniques have not been developed or implemented in many grid workflow systems'.

General-purpose languages for defining and managing workflows do not provide fault-tolerant capabilities targeted for grid environments. Nonetheless, the Business Process Execution Language for Web Services (BPEL4WS) [156] standard does provide extensive workflow-level mechanisms for fault handling, using traditional throw and catch semantics. Many researchers have composed grid workflows using early versions of BPEL4WS, such as [157–159]. In [160] it was observed that use of BPEL4WS fault handling to recover one failed workflow process required restart of all other concurrently executing processes, which would be highly disadvantageous in a grid workflow. In response, Tartanoglu *et al.* [160] proposed a specification language and related mechanisms to overcome this problem. Wasserman and Emmerich [161] conducted a study of failure in scientific grid workflows that were composed with an early version of BPEL4WS. They also concluded that available methods for ensuring reliability in workflows were inadequate. Other standard specifications for coordinating distributed computations also address fault tolerance in a general web service context, such as [162,163]. Finally, the recent emergence of Web 2.0 network services [164,165] has made available a set of software components for constructing grid workflows. Because it is new, Web 2.0 has not yet been investigated from the standpoint of grid reliability.

4.2. Coordinating workflow and grid resource fault-tolerance strategies

The preceding discussions identified two sources for providing fault tolerance in grid systems: fault-tolerance capabilities provided from within the grid system and capabilities provided from a grid application or workflow. Grid system designers, users, administrators, and operators need to consider when to use each approach in order to prevent unnecessary redundancy. Grid applications and workflows may not need to provide fault tolerance if the grid resources allocated to their tasks already provide these capabilities. For instance, a grid workflow need not prescribe recovery actions for a set of concurrent processes if the grid resources hosting these processes perform coordinated checkpointing. However, if grid resources are known not to provide fault tolerance, it may be prudent for applications to ensure these capabilities themselves. In commercial grid environments, one can envision users and service providers negotiating how fault tolerance is to be provided as part of creating a SLA [84]. To enable this coordination will require standardized conventions for describing and negotiating fault-tolerance capabilities, a topic for future research.

5. SUPPORTING GRID INFRASTRUCTURE AND RESOURCE MANAGEMENT

Infrastructure and management services are essential services for managing operation of the grid. They include services for monitoring status of resources and discovery of grid resources through directories or other facilities. They also include services for scheduling use of grid resources, managing execution of user applications on resources, grid usage and accounting services, security (authentication, authorization, encryption, etc.), and others. As with grid resources, the reliability of infrastructure and management services can be improved by applying fault-tolerance methods described in Section 3. However, in contrast to grid resources, infrastructure and management services have a wider scope, and their function is central to the operation of the grid. Therefore, ensuring reliability of these services is critical, and thus in this study, fault tolerance of infrastructure and management services is treated as a separate area of research. However, this area has received less attention than it merits. For example, few of the grid resource management systems surveyed in [166] were reported to have built-in fault-tolerant capabilities. This section reviews research directed toward making infrastructure and management services more fault tolerant.

Work on fault-tolerant data replica management services [111–114] has been described above (see Section 3.2.4). The OGF *Grid Monitoring Architecture* document [37] describes generic requirements for the grid monitoring function, including fault tolerance, security, scalability, and interoperability across heterogeneous grid resources. While many grid monitoring systems surveyed in [36] were shown to lack fault tolerance, a few experimental systems have been developed that attempt to realize this goal. In [167], a hierarchically based grid monitoring design was used to improve fault tolerance; while in [168] monitored data were replicated at multiple nodes. In [169], registries containing information about monitored resources were replicated for fault-tolerance purposes. In [170], a fault-tolerant service discovery system was described that is built on top of the Jini Service Discovery protocol [171]. This approach exploited the inherent redundancy of Jini lookup services to build a distributed, hierarchical index of grid resources, in which index nodes were replicated and geographically distributed. The number of steps needed to access a node in the hierarchy was shown to scale in relation to the number of index nodes with a complexity of $O(\log n)$, where n is the number of index nodes. Experimental testbed results documented system performance, but did not address fault tolerance. Another important function in grid systems is co-allocation, or co-scheduling, of tasks that must run concurrently on different resources. In [172], a co-allocation service was proposed that achieved fault tolerance by using distributed, redundant transaction coordinators, and the Paxos consensus algorithm [90,173]. In [174], redundancy techniques were described that improved survivability and resistance to attack of secure communications services in the Cactus system [60].

Finally, reliability metrics have an especially important role in connection with infrastructure and management services. Because these services manage grid resources, they provide a convenient means for applying metrics to measure the reliability of grid resources and, thus indirectly, the grid itself. In [175], grid resource allocation was supported by a scheme for measuring reliability, or trust, of compute nodes in a desktop grid system using Dempster–Shafer uncertainty theory [176]. In [88], quantitative reliability ratings for computing resources were used to guide resource allocation and scheduling. In [111], data availability metrics were used for evaluating algorithms for optimizing data replication. In [177], a framework for evaluating quality of service provided by a grid system was described that defined metrics for service accessibility and availability.

6. GRID CONNECTION AND TRANSPORT RELIABILITY

The OGF informational document [14] sets forth requirements for grid network systems. Among the most important are high network availability and reliable, rapid transport of bulk data (over 1 Gigabit per second). Because grid applications often require data transport capabilities of this size for extended durations, connections between user application and grid resource sites must be reliable and stable for long periods. Another key requirement is reliable multicast transmission of large data sets to multiple remote computing resources. Here again, connections must be maintained for extended periods and data delivery must be ensured in the face of faults among lower-level network components.

The need to maintain connections for long periods requires use of a large number of network components. This in turn increases the probability of failures that necessitate rerouting connections through functioning components. For these reasons, ensuring reliable transport in large-scale grid networks is critical. The section first discusses standards for reliable connectivity and data transport that are coming into use in grid environments and related work on strengthening their fault-tolerance features. Then, the section examines methods being investigated for ensuring reliable connectivity and data transport in grid networks. Two are of special interest: overlay (or virtual) networks and dedicated networks. Finally, reliable multicast transmission in grid systems is addressed.

6.1. Specifications for reliable connection and transport

To date, there are three main specifications for point-to-point unicast communications used in grid environments: two specifications for reliable connection and message exchange and the GridFTP specification for bulk data transfer. TCP provides general purpose, fault tolerant, point-to-point connectivity in network systems and is employed for connection establishment in grid environments. In this role, TCP is used in combination with other transport protocols. In [178], an OGF survey of available TCP alternatives found that no alternative, by itself, fully met the requirements of grid networks.

Web Services Reliable Messaging (*WS-ReliableMessaging*) [179] was developed by a group of vendors to define a protocol for guaranteed message delivery. The specification provides procedures for transferring a sequence of messages between remote components. WS Reliable Messaging also specifies requirements for tracking the status of messages sent between components, guaranteed message ordering, and elimination of duplicate messages—in order to guarantee *at-most-once* message delivery. A second specification, *WS-Reliability*, [180] was also created by a vendor group and provides similar capabilities. In [181], a comparison of the two reliable messaging specifications concluded that *WS-ReliableMessaging* provides more flexible features for re-initiating erroneous transmissions and more extensive capabilities for reporting faults that occur during transmission. Initial efforts to implement *WS-ReliableMessaging* [181,182] and *WS-Reliability* have not yet yielded information on the comparative effectiveness of these specifications in production environments. Subsequent standardization of *WS-ReliableMessaging* [179] by OASIS [2] indicates growing use of this specification.

Grid File Transfer Protocol (GridFTP) [183] is a specification developed by the Globus Alliance and OGF. GridFTP extends the FTP protocol [184] to permit point-to-point transfer of large, ‘bulk’ data over a wide-area network. Widely used in grid systems for scientific applications, GridFTP transfers large files by taking advantage of ‘long fat’ communication channels to create multiple TCP data streams that significantly improve aggregate throughput. GridFTP utilizes fault-tolerance mechanisms provided by TCP and employs a checksum technique to detect data loss during transfer. In [185], GridFTP was found to be highly reliable and scalable in comparison with other bulk data transfer protocols. However, a known problem in GridFTP is that failure of a client necessitates restart of data transmission, a disadvantage when transferring large data sets. This is overcome by fault-tolerance mechanisms described below.

6.2. Research in fault-tolerant grid networks

The goal of the specifications discussed above is to describe protocols for attaining reliable connectivity and data transport. To achieve this goal requires that the underlying network implementation itself be fault tolerant and highly available. This in turn first requires the ability to assess the reliability of the network state. Initial reliability metrics for grid networks were specified by the OGF [186] and by Lowekamp *et al.* [187] for this purpose. To achieve fault tolerance and ensure high network availability, researchers have investigated use of overlay, or virtual, networks as well as networks dedicated to grid systems use.

In [188,189], a messaging infrastructure was described for supporting communication and large-scale data transfer in grid systems. The infrastructure employed redundant distributed intermediate brokers to form a virtual software overlay network, the *NaradaBrokering* system, for managing large data streams. The infrastructure supported multiple protocols including UDP, TCP and parallel TCP, Simple Object Access Protocol (SOAP) messaging [190], as well as web service specifications for addressing [191] and event notification [192]. The infrastructure implemented both *WS-Reliable Messaging* and *WS-Reliability* to facilitate ordered, guaranteed, at-most once delivery of messages and events. The overlay network of redundant brokers and links provided fault tolerance in the face of broker failure, failure or disconnect of communicating services, and link failure. Flexible reconfiguration of the broker topology [193] through operations specified in WS-Management [194] enhanced scalability of the network. The viability of this approach was shown in the implementation of prototype grid applications involving streaming audio and video data and in the use of the broker network to implement a recovery mechanism for GridFTP [195]. The Globus toolkit [93] also includes a service for fault-tolerant transfer of data by GridFTP through an intermediate distributed DBMS that performs a function analogous to the broker network in [188, 189]. A solution to this problem was also proposed in [196], who demonstrated a service for bulk data transfer between heterogeneous storage systems that use dissimilar data access protocols. This service relied on redundant data transfer across intermediate hops to improve fault tolerance. In [197], improved reliability of parallel connections was demonstrated using prototype tools for detecting and recovering from connection failures.

The OGF informational document [14] identified efficient routing as a key to achieving high availability in networks that serve grid systems. Routing involves using a traffic engineering function [198] to select paths through a network that maximizes data flow within available bandwidth. Effective routing mechanisms are important for dynamically rerouting grid data flows around failed network components, and therefore, identifying routing mechanisms that perform well in grid environments is an important research topic. This is especially so because both current interior gateway protocols, such as the combined Open Shortest Path First, Intermediate System to Intermediate System (OSPF/IS-IS) [199,200], and exterior gateway protocols, such as the Border Gateway Protocol (BGP) [201], are regarded as insufficient by some [14,82]. A possible approach to improved routing in grid networks, discussed in [14], involves creating virtual overlay networks on top of existing physical networks using the Multi-Protocol Label Switching [202,203] standard. Clapp *et al.* [204] also proposed a dynamic grid networking layer that provided automatic bandwidth on demand. An important issue for grid overlay networks is the interaction between the overlay and underlying network resources, since the reliability of the overlay depends on the reliability of these resources. For example, the overlay network may wish to be notified of underlying resource failure in order to take appropriate recovery actions or request allocation of additional resources for rerouting purposes. Thus, from a reliability standpoint, understanding interactions between overlays and the supporting network layers is an area where more work is needed.

Given the research on promoting fault tolerance at different logical network layers represented by such systems as the NaradaBrokering system [188] (high layer) and overlay networks (lower layer), an interesting question to consider is whether a combined solution is possible that leverages multiple overlays at different logical network levels. For instance, would mapping a software overlay represented by a broker network over lower-level virtual paths belonging to an overlay network lead to improved availability in a grid network? Similarly, would fault tolerance be enhanced by mapping long-distance connections between the service islands described in [82] onto an overlay network? Questions such as these may be future topics of research. More generally, additional work on overlay networks is needed to determine how best to deploy these solutions in grid environments. It is important to know how management of overlay networks might differ for grid applications, particularly with respect to fault tolerance. Finally, it is necessary to investigate the use of dedicated grid networks in which network resources are used exclusively by a grid, rather than shared in a public network. Given the heavy demands of grid data transport, some have chosen to form dedicated networks. An example of this is the interlinking of the TeraGrid research grid [205] using optical network backplane [206]. However, to date, dedicated networks have been used largely for grids with a limited number of participants. An important question to answer is whether this solution will scale in very large grids with dynamic membership.

6.3. Reliable multicasting of large data sets

Multicast, or point-to-multipoint, transmission of large data sets in grid environments is a highly critical capability. For instance, scientific grid systems may require transmission of instrument or simulation data originating at one site to multiple, remote storage sites. One well-known example of the use of multicasting in grid applications is the Access Grid [207], where large audio and video data sets are regularly broadcast to many participants. However, in [207] and in other grid networks [188], only best-effort multicast is used, which provides high throughput and low end-to-end delay, but does not provide guaranteed delivery.

Reliable multicast protocols have been the subject of research for years, both within grid settings and for more general purpose use [208–213]. The Nack-oriented reliable multicast (NORM) protocol is currently being developed as an IETF standard [213]. In [214], a series of trials were conducted in a wide-area grid network testbed to compare performance of NORM, the Multicast Dissemination Protocol (MDP) [215], and a variant of TCP extended for multicasting. The results revealed that NORM and MDP had significant design problems, bottlenecks, and limitations that hampered throughput. These protocols were also determined to be less robust than TCP in real-world production networks.

Other multicast solutions also have been proposed. For instance, use of hierarchical multicast trees in [216] provided good performance for multicast groups consisting of up to 100 members in a simulated wide-area network, but did not provide guaranteed delivery. In [217], experiments were reported on using the Tree-based Reliable Multicast (TRAM) protocol [218] to multicast across a 10-node, heterogeneous compute grid. Despite these and other efforts, no multicast protocol has been yet identified that provides delivery of large data sets that is reliable, scalable, and meet the performance requirements of large-scale grid systems. Nevertheless, a survey of available multicast techniques in [210] indicated the potential for such a protocol, and efforts to develop a standard reliable multicast protocol that is scalable and efficient continue [219].

7. RELIABILITY CONCERNS FROM AN OVERALL SYSTEM PERSPECTIVE

This section discusses approaches to grid system reliability which are based on an overall system-wide view, rather than focusing on individual functional areas or component types. The overall perspective facilitates insight into system-level processes that might not be obtained by analyzing individual components, sites, or subsystems. There are three approaches to overall system reliability to consider. The first views the grid from an architectural standpoint. An architectural approach seeks to analyze the design or structure of the grid to improve the overall reliability by, for instance, identifying architectural alternatives than foster fault tolerance. A second approach involves using quantitative methods and algorithms to measure overall grid system reliability. The third approach involves viewing the grid as a complex system, in which the individual behaviors of large numbers of components may collectively lead to an emergent global behavior that cannot be predicted from behavior of individual components. If the resulting global behavior degrades the overall performance of the grid system, this effectively constitutes a fault situation.

7.1. Grid reliability from an architectural perspective

A *grid system architecture* describes the structure of a grid system, which consists of nodes[¶] and their interconnections. Nodes, or sites, contain grid resources and related software components. A grid architecture can be viewed as a high-level design of a grid system. To date, few researchers have investigated how differences in architecture might affect system reliability. Grid architectures may be distinguished in several ways that might impact reliability. Architectures may be differentiated by different topologies of sites. For instance, a hierarchical architecture contrasts with a decentralized architecture. In the former, sites are organized into a logical tree, while the latter has no central point of control that could also become a point of failure. Architectures can also be distinguished by number and location of sites or by the distribution of resources across sites; i.e. a few sites with many resources versus many sites having few resources. Distribution of resources across sites could impact resource availability if a site becomes unreachable. Architectures can be differentiated by the number and location of infrastructure and management services on nodes within the grid system, which may also influence reliability if single points of failure or bottlenecks develop. Another distinguishing factor is the logical structure of software components that implement infrastructure and management services. Both the distribution of these services and the design of the software system that implements them may affect their ability to reliably carry out management functions. To date, few researchers have used architectural concepts to analyze grid reliability. An exception is [130,131] which, as described in Section 3.3, proposed a method for identifying central ‘hub’ software components that are more likely to impact overall system reliability. In [220], a grid system architecture based on the Open Grid Service Architecture (OGSA) [7] was proposed to enhance reliability. The influence of architecture on grid reliability was also investigated in [189,221]. Given the importance of architecture considerations, these initial efforts should be expanded, so that ultimately it is possible to develop guidelines for design of fault-tolerant grid system architectures.

7.2. Methods for quantitative assessment of grid system reliability

Methods for quantitatively measuring and optimizing the overall reliability of distributed systems have long been a research topic (see [24,25,222] for overviews). It was noted early that quantitative

[¶]This concept is distinguishable from a reference model architecture, such as the OGSA [7].

assessments of system reliability depend on architectural considerations, since computing reliability of arbitrarily structured distributed systems was shown to be intractable [223]. More recently, Xie *et al.* [222] described a method that uses known component failure rates to measure the overall reliability of a grid system. The method considered a system model consisting of a set of nodes, links, grid resources, and a resource management system. A set of workflow tasks was allocated to grid resources on particular nodes. Given that each of these model components has a known failure rate, the reliability of a grid system could be estimated by the probability that a set of user applications, executing on the grid as workflows, will complete. In subsequent work, Dai *et al.* [224] revised this approach into a layered hierarchal grid model, where reliability analysis considered the probability of different kinds of failures at each layer. The analysis incorporated Markov chain modeling of resource request queues in which blocking and time-out failures occur. In related work, Dai and Levitin [225] presented a model for measuring reliability of a set of grid components that share common communications links and are controlled by the same resource manager. Using this model as a basis, an algorithm for optimizing resource allocation was presented in [226]. In [227], an algorithm for computing reliability of grid systems having a star architecture was described. Developing quantitative metrics for evaluating operational, mission-critical grid systems will be critical for adoption and use of grid technology. However, work on grid systems metrics is only in the initial stages.

7.3. Grid reliability from the complex systems perspective

Complex systems are large collections of interconnected components whose interactions can lead to emergent global behaviors that may not be predicable from individual component behaviors. From the standpoint of grid reliability, the study of grid systems as complex systems seeks to develop analytical methods that reveal emergent global states in which performance is impaired to a degree that constitutes a system-wide fault state. In some cases, component interactions may lead to undesirable global states even though individual components have functioned as intended. Understanding the causes of emergent behavior provides a basis for developing decentralized methods of control that leads to desirable global states when implemented by components across a grid. However, developing tractable methods to understand causes of emergent behavior presents a challenge because of grid system scale, heterogeneity, and dynamism.

Work toward developing simulation tools to study dynamics of grid systems was reported in [228]. Other simulation studies demonstrated the advantages of viewing a grid as a complex system. In [229], it was shown through simulation that when resource allocation operations are randomly subjected to malicious spoofing on a global scale, a plausible response intended to isolate spoofed service providers can actually lead to further degradation of global system performance. In [230], it was found a decentralized grid compute economy produced good global resource allocations during periods of excess demand and responded well to sudden overloads caused by temporary provider failure. Related work in [231] demonstrated in more detail the feasibility of using grid standard specifications to produce viable resource allocations in a grid compute economy. These examples show that complex systems methods can be used to understand global behavior in grid systems, and thus provide a basis for improving system reliability. As with architectural analysis and quantitative measurement, the importance of this work is likely to increase as the scale of grid systems increases. Societal investment in research to develop analysis methods that are based on overall systems perspectives is therefore a necessity.

8. FINDINGS AND CONCLUSIONS

This study has surveyed the progress in making grid systems more reliable. The study has described how grid systems are characterized by conditions of large scale, heterogeneity, and dynamism. These factors distinguish grid systems from other kinds of distributed systems and pose significant challenges to ensuring reliability.

The study has found that, to date, efforts to make grid systems more reliable have centered on developing methods for fault tolerance. Efforts toward improving fault tolerance have focused on distinct functional areas of grid computing, including computational hardware and software resources, user applications and workflows, infrastructure and management services, and grid networks. Generally, work has progressed differently in these areas, and in each area, different problems remain to be solved. For grid resources, new fault detection methods have been proposed, but the problem of ensuring completeness and accuracy in scaled environments still requires more work. Progress has been made in checkpointing and recovery methods, but here, scalability also needs further investigation. Resource replication shows promise as a fault-tolerance method for large-scale environments. However, this area has thus far been the subject of less effort than is warranted, given the fault-tolerance potential provided by the innate redundancy of grid resources. Similarly, dynamic rescheduling has not been sufficiently investigated as a fault-tolerance technique. Overall, fault-tolerance methods for grid resources remain in the research stage and have been largely unproven in large-scale, production environments. Still less work has occurred on test methods for fault prevention. Here, a comprehensive study is needed on the cost effectiveness of testing in order to better understand requirements for testing grid components in heterogeneous and dynamic environments. As in the case of test methods, fault-tolerance methods for grid applications and workflows, as well as for infrastructure and management services, need additional focus. With regard to grid networks, there has been notable progress in ensuring reliable connectivity and bulk data transport. Significant efforts have also been devoted to improving fault tolerance through the use of overlay networks and dedicated networks. However, questions remain about the use of these methods under conditions of scale, heterogeneity, and dynamism.

The study found a number of key specifications that need to be further examined to determine if they should be extended to support reliability. In some cases, needed specifications are incomplete, such as for reliable multicasting and application checkpoint and recovery. Comprehensive guidelines are needed for implementing fault tolerance in grid environments. Guidelines are needed for such areas as specifying when fault tolerance should be provided by workflow management systems, ensuring reliability of core infrastructure and management services, and designing fault-tolerant grid system architectures. Similarly, while progress has been made in developing metrics for grid networks, there has been much less work on metrics for other functional areas of grid computing. Without measurement methods for all phases of grid operations, it will not be possible to measure the reliability of large-scale grid systems. In the area of new grid-related technologies, the recent emergence of Web 2.0 network services [164,165] and its potential use in grid systems has not yet been investigated from the standpoint of grid reliability. Similarly, there has not been sufficient time to examine whether cloud computing [27] may yield reliability benefits that could be applied in grid systems.

In conclusion, developing reliability methods for large-scale, heterogeneous, dynamic grid systems remain a challenge that must be met if the vision of future grid systems is to be fully realized. At present levels of scale, fault-tolerance methods adapted from other areas of distributed computing can provide reliability to grid systems used in individual enterprises or in small groups of cooperating organizations. However, as grid systems grow in scale, heterogeneity, and dynamism, existing methods will have to be adapted and new methods will need to be developed. Similarly, grid system growth will require new reliability measurement and test methods. This work will have to proceed with full appreciation of the distinguishing circumstances of grid environments. These circumstances also indicate that increased emphasis will need to be placed on analysis of grid systems from an overall systems perspective. In particular, understanding grids as complex systems will become increasingly necessary, because scale and dynamism are more likely to facilitate undesirable emergent behaviors that cannot be predicted, or prevented, by analyzing individual component behavior. Given that ensuring reliability is critical to the continued advancement of grid technology, it is important that work in all areas of grid reliability continues and is expanded, and that promising experimental methods continue to be evolved for deployment in future production environments.

APPENDIX A

Table AI lists references provided in this paper by the major topical areas of grid system reliability each reference addresses. Some references pertain to more than one area.

Table AI. References listed by major topical area of grid system reliability.

Fault tolerance of grid resources	
Fault detection	[10–13,15,16,29–52]
Checkpointing and process migration	[16,21,22,46,49,53–78]
Resource replication	[28,46,79–94]
Rescheduling	[95–97].
Data replication	[8,28,93,98–115]
Testing and certification of grid resources	
	[93,116–139]
Reliability of grid applications and workflows	
	[48,84,140–165]
Infrastructure and management services	
	[36,37,60,88,90,111–114,166–177]
Grid connection and transport reliability	
Reliability specifications	[2,178–185]
Fault tolerance methods	[14,82,93,186–206]
Reliable multicasting	[188,207–219]
Overall system perspective	
Architecture	[7,130,131,189,220,221]
Measurement methods	[24,25,222–227]
Complex systems	[228–231]

ACKNOWLEDGEMENTS

I would like to thank the participants in the workshops on Grid reliability sponsored by the Open Grid Forum for their valuable contributions. Special thanks also go to Matti Hiltunen and Kevin Mills for their many insightful comments that helped improve this manuscript.

REFERENCES

1. Open Grid Forum. http://www.ogf.org [4 August 2008].
2. Organization for the Advancement of Structured Information Standards (OASIS). http://www.oasis-open.org [4 August 2008].
3. Internet Engineering Task Force. http://www.ietf.org [4 August 2008].
4. Avizienis A, Laprie J, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* 2004; **1**(1):11–33.
5. Foster I, Kesselman C, Tuecke S. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications* 2001; **15**(2):200–222.
6. Foster I, Kesselman C, Nick J, Tuecke S. The physiology of the Grid: An open Grid services architecture for distributed systems integration. http://www.globus.org/alliance/publications/papers.php [4 August 2008].
7. Foster I, Kishimoto J, Savva A, Berry D, Djaoui A, Grimshaw A, Horn B, Maciel F, Siebenlist F, Subramaniam R, Treadwell J, Von Reich J. The open Grid services architecture, version 1.5. *GFD.80*, Open Grid Forum, July 2006.
8. Raffo D. Grid, redundancy, and home-cooked management help site survive. *Byte and Switch*, 22 November 2006.
9. Carr D. How Google works. *Baseline Magazine*, 6 July 2006. http://www.baselinemag.com [15 July 2008].
10. Mogilevsky D, Koenig G, Yurcik W. Byzantine anomaly testing for Charm++: Providing fault tolerance and survivability for Charm++ empowered clusters. *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, May 2006. IEEE Computer Society Press: Washington, DC, 2006; 30.
11. Wang X, Zhuang Y, Hou H. Byzantine fault tolerance in MDS of Grid system. *Proceedings of the 2006 International Conference on Machine Learning and Cybernetics*, August 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 2782–2787.
12. Li Q, Xu M, Zhang H. A root-fault detection system of Grid based on immunology. *Proceedings of the Fifth International Conference on Grid and Cooperative Computing*, October 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 369–373.
13. Kola G, Kosar T, Livny M. Faults in large distributed systems and what we can do about them. *Proceedings of the 11th European Conference on Parallel Processing* (*Lecture Notes in Computer Science*, vol. 3648), Cunha J, Medeiros P (eds.). Springer: Berlin, 2005; 442–453.
14. Sander V (ed.). Networking issues for Grid infrastructure. *Informational Document GFD-I.037*, Open Grid Forum, November 2004.
15. Barborak M, Malek M. The consensus problem in fault-tolerant computing. *ACM Computing Surveys* 1993; **25**(2): 171–220.
16. Hwang S, Kesselman C. A flexible framework for fault tolerance in the Grid. *Journal of Grid Computing* 2003; **1**(3):251–272.
17. Katker S, Geihs K. A generic model for fault isolation in integrated management systems. *Journal of Network and Systems Management* 1997; **5**(2):109–130.
18. Medhi D. Network reliability and fault tolerance. *Wiley Encyclopedia of Computer Science and Engineering*, Webster J (ed.), vol. 14. Wiley: New York, 1999; 213–218.
19. Medard M, Lumetta S. Network reliability and fault tolerance. *Wiley Encyclopedia of Engineering*, Proakis J (ed.). Wiley: New York, 2003.
20. Baker M (ed.). Cluster Computing White Paper, version 2.0. IEEE Computer Society Task Force on Cluster Computing (TFCC), December 2000.
21. Milojicic D, Douglis F, Paindaveine Y, Wheeler R, Zhou S. Process migration survey. *ACM Computing Surveys* 2000; **32**(3):241–299.
22. Elnozahy E, Johnson D, Wang Y. A survey of rollback recovery protocols in message-passing systems. *ACM Computing Surveys* 2002; **34**(3):375–408.
23. Lin J, Dunham M. A survey of distributed database checkpointing. *Distributed and Parallel Databases* 1997; **5**(3): 289–319.
24. Goseva-Popstojanova K, Trivedi K. Architecture-based approaches to software reliability prediction. *Computers and Mathematics with Applications* 2003; **46**(7):1023–1036.

25. Kuo W, Prasad V. An annotated overview of system-reliability optimization. *IEEE Transactions on Reliability* 2000; **49**(2):176–187.
26. Lua E, Crowcroft J, Pias M, Sharma R, Lim S. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys* 2005; **7**(2):72–93.
27. Jha S, Merzky A, Fox G. Using clouds to provide Grids higher-levels of abstraction and explicit support for usage modes. *Concurrency and Computation*: *Practice and Experience*, *Special OGF Issue* 2008; in press.
28. Amazon Simple Storage Service (Amazon S3). http://aws.amazon.com/s3 [4 August 2008].
29. Bianchini R, Buskens R. An adaptive distributed system-level diagnosis algorithm and its implementation. *Proceedings of the Twenty-first International Symposium on Fault-Tolerant Computing*, *Digest of Papers*, June 1991. IEEE Computer Society Press: Los Alamitos, CA, 1991; 222–229.
30. Stok P, Claessen M, Alstein D. A hierarchical membership protocol for synchronous distributed systems. *Proceedings of the First European Dependable Computing Conference on Dependable Computing* (*Lecture Notes in Computer Science*, vol. 852), Echtle K, Hammer D, Powell D (eds.). Springer: London, 1994; 599–616.
31. Gupta I, Chandra T, Goldszmidt G. On scalable and efficient distributed failure detectors. *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing*, August 2001. ACM Press: New York, 2001; 170–179.
32. Presuhn R, Case J, Rose M, Waldbusser S. Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). *RFC 3416*, Internet Engineering Task Force, December 2002.
33. Fischer M, Lynch N, Paterson M. Impossibility of distributed consensus with one faulty process. *Journal of the Association for Computing Machinery* 1985; **32**(2):374–382.
34. Chandra T, Toueg S. Unreliable failure detectors for reliable distributed systems. *Journal of the Association for Computing Machinery* 1996; **43**(2):225–267.
35. Hayashibara N, Cherif A, Katayama T. Failure detectors for large-scale distributed systems. *Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems*, October 2002. IEEE Computer Society Press: Los Alamitos, CA, 2002; 404–409.
36. Zanikolas S, Sakellariou R. A taxonomy of Grid monitoring systems. *Future Generation Computer Systems* 2005; **21**(1):163–188.
37. Tierney B, Aydt R, Gunter D, Smith W, Swany M, Taylor V, Wolski R. A Grid monitoring architecture. *Informational Document GFD-I.7*, Open Grid Forum, January 2002.
38. Stelling P, Foster I, Kesselman C, Lee C, von Laszewski G. A fault detection service for wide area distributed computations. *Cluster Computing* 1999; **2**(2):117–128.
39. Jain A, Shyamasundar R. Failure detection and membership management in Grid environments. *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*, November 2004. IEEE Computer Society Press: Los Alamitos, CA, 2005; 44–52.
40. Horita Y, Taura K, Chikayama T. A scalable and efficient self-organizing failure detector for grid applications. *Proceedings of the Sixth IEEE/ACM International Workshop on Grid Computing*, November 2005. IEEE Computer Society Press: Los Alamitos, CA, 2006; 202–210.
41. Das A, Gupta I, Motivala A. SWIM: Scalable weakly-consistent infection-style process group membership protocol. *Proceedings of the International Conference on Dependable Systems and Networks*, June 2002. IEEE Computer Society Press: Los Alamitos, CA, 2002; 303–312.
42. Abawajy J. Fault detection service architecture for Grid computing systems. *Proceedings of the International Conference on Computational Science and Its Applications*, *Part II* (*Lecture Notes in Computer Science*, vol. 3044), Lagana A, Gavrilova M, Kumar V, Mun Y, Tan C, Gervasi O (eds.). Springer: Berlin, 2004; 107–115.
43. Chan K, Bishop J, Steyn J, Baresi L, Guinea S. A fault taxonomy for web service compositions, University of Pretoria, December 2006.
44. Bruning S, Weissleder S, Malek M. A fault taxonomy for service-oriented architecture. *Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium*, November 2007. IEEE Computer Society Press: Los Alamitos, CA, 2007; 367–368.
45. Hofer J, Fahringer T. A multi-perspective taxonomy for systematic classification of grid faults. *Proceedings of the 16th Euromicro International Conference on Parallel*, *Distributed and Network-based Processing*, February 2008. IEEE Computer Society Press: Los Alamitos, CA, 2008; 126–130.
46. Jitsumoto H, Endo T, Matsuoka S. ABARIS: An adaptable fault detection/recovery component framework for MPI. *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, March 2007. IEEE Computer Society Press: Los Alamitos, CA, 2007; 1–8.
47. Duan R, Prodan R, Fahringer T. Data mining-based fault prediction and detection on the Grid. *Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing*, June 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 305–308.
48. Xiang Y, Li Z, Chen H. Optimizing adaptive checkpointing schemes for Grid workflow systems. *Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops*, October 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 181–188.

49. Jin H, Shi X, Qiang W, Xou D. DRIC: Dependable Grid computing framework. *IEICE Transactions on Information and Systems* 2006; **E89-D**(2):612–623.
50. Röblitz T, Schintke F, Reinefeld A, Bärring O, Barroso Lopez M, Cancio G, Chapeland S, Chouikh K, Cons L, Poznanski P, Defert P, Iven J, Kleinwor T, Panzer-Steindel B, Polok J, Rafflin C, Silverman A, Smith T, Van Eldik J, Front D, Biasotto M, Aiftimiei C, Ferro E, Maron G, Chierici A, Dell'agnello L, Serra M, Michelotto M, Hess L, Lindenstruth V, Pister F, Steinbeck T, Groep D, Steenbakkers M, Koeroo O, de Cerff W, Venekamp G, Anderson P, Colles T, Holt A, Scobie A, George M, Washbrook A, Garcıa Leiva R. Autonomic management of large clusters and their integration into the Grid. *Journal of Grid Computing* 2005; **2**(3):247–260.
51. Tanimura Y, Ikegami T, Nakada H, Tanaka Y, Sekiguchi S. Implementation of fault-tolerant GridRPC applications. *Journal of Grid Computing* 2006; **4**(2):145–157.
52. Duarte A, Brasileiro F, Cirne W, Filho J. Collaborative fault diagnosis in Grids through automated tests. *Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, April 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 69–74.
53. DataSynapse GridServer. http://www.datasynapse.com/en/products/gridserver.php [4 August 2008].
54. Hewlett-Packard Grid Computing. http://h20331.www2.hp.com/enterprise/cache/125369-0-0-225-121.html [4 August 2008].
55. Tivoli NetCool/OMNIbus. http://www-306.ibm.com/software/tivoli/products/netcool-omnibus/ [4 August 2008].
56. IBM alphaWorks Grid Computing. http://www.alphaworks.ibm.com/grid [4 August 2008].
57. Administering Platform Process Manager, version 3.0. Platform Computing Corporation, March 2005.
58. N1 Grid Engine User's Guide. Sun MicroSystems, Inc., May 2005.
59. Natrajan A, Humphrey M, Grimshaw A. Capacity and capability computing in legion. *Proceedings of the International Conference on Computational Sciences—Part I*, May 2001. Springer: London, 2001; 273–283.
60. Goodale T, Allen G, Lanfermann G, Masso J, Radke T, Seidel E, Shalf J. The cactus framework and toolkit: Design and applications. *Proceedings of the Fifth International Conference on Vector and Parallel Processing* (*Lecture Notes in Computer Science*, vol. 2565), Goos G, Hartmanis J, van Leeuwen J (eds.). Springer: Berlin, 2003; 15–36.
61. Lanfermann G, Allen G, Radke T, Seidel E. Nomadic migration: Fault tolerance in a disruptive Grid environment. *Proceedings of the 2nd IEEE/ACM International Symposium Cluster Computing and the Grid*, May 2002. IEEE Computer Society Press: Los Alamitos, CA, 2002; 280–282.
62. Adding High Availability and Replication to the Condor Central Manager. http://dsl.cs.technion.ac.il/projects/gozal/project_pages/ha/ha.html [4 August 2008].
63. Limaye K, Leangsuksum B, Greenwood Z, Scott S, Engelmann C, Libby R, Chanchio K. Job-site level fault tolerance for cluster and Grid environments. *Proceedings of the IEEE International Conference on Cluster Computing*, September 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 1–9.
64. Bouteiller A, Herault T, Krawezik G, Lemarinier P, Cappello F. MPICH-V: A multiprotocol automatic fault tolerant MPI. *International Journal of High Performance Computing and Applications* 1999; **20**(3):319–330.
65. MPI-2, Extensions to the message-passing interface. Message Passing Interface Forum, November 2003. http://www.mpi-forum.org/ [14 July 2008].
66. Sankaran S, Squyres J, Barrett B, Sahay V, Lumsdaine A, Duell J, Hargrove P, Roman E. The Lam/Mp1 checkpoint/restart framework: System-initiated checkpointing. *International Journal of High Performance Computing Applications* 2005; **19**(4):479–493.
67. Yeom H. Providing Fault-tolerance for parallel programs on Grid (FT-MPICH). *First Workshop on Reliability and Robustness in Grid Computing Systems*, February 2006. Open Grid Forum. http://www.ogf.org [15 July 2008].
68. Woo N, Yeom H, Park T. MPICH-GF: Transparent checkpointing, rollback-recovery for Grid-enabled MPI processes. *IEICE Transactions on Information and Systems* 2004; **E87-D**(7):1820–1828.
69. Kim H, Yeom H. A user-transparent recoverable file system for distributed computing environment. *Proceedings of the Workshop on Challenges of Large Applications in Distributed Environments*, July 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 45–53.
70. Buntinas D, Coti C, Herault T, Lemarinier P, Pilard L, Rezmerita A, Rodriquez E, Cappello F. Blocking vs non-blocking coordinated checkpointing for large-scale fault tolerant MPI. *Future Generation Computer Systems* 2008; **24**(1):73–84.
71. Strom E, Yemini S. Optimistic recovery in distributed systems. *ACM Transactions on Computer Systems* 1985; **3**(3):204–226.
72. Woo N, Jung H, Shin D, Han J, Yeom H, Park T. Performance evaluation of consistent recovery protocols using MPICH-GF. *Proceedings of the 5th European Dependable Computing Conference* (*Lecture Notes in Computer Science*, vol. 3463), Dal Cin M, Kaaniche M, Pataricza A (eds.). Springer: Berlin, 2005; 167–178.
73. de Camargo R, Cerqueira R, Kon F. Strategies for storage of checkpointing data using non-dedicated repositories on Grid systems. *Proceedings of the 3rd International Workshop on Middleware for Grid Computing*, November 2005. ACM Press: New York, 2006; 1–8.
74. Ren X, Eigenmann R, Bagchi S. Failure-aware checkpointing in fine-grained cycle sharing systems. *Proceedings of the 16th International Symposium on High performance Distributed Computing*, June 2007. ACM Press: New York, 2007; 33–42.

75. Ramkumar B, Strumpen V. Portable checkpointing for heterogeneous architectures. *Proceedings of the Twenty-seventh Annual International Symposium on Fault-tolerant Computing*, *Digest of Papers*, June 1997. IEEE Computer Society Press: Los Alamitos, CA, 1997; 58–67.
76. Zandy V, Miller B, Livny M. Process hijacking. *Proceedings of the Eighth International Symposium on High Performance Distributed Computing*, August 1999. IEEE Computer Society Press: Washington, DC, 1999; 177–184.
77. Vadhiyar S, Dongarra J. SRS—A framework for developing malleable and migratable parallel software. *Parallel Processing Letters* 2003; **13**(2):291–312.
78. Fernandes R, Pingali K, Stodghill S. Mobile MPI programs in computational Grids. *Proceedings of the Eleventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, March 2006. ACM Press: New York, 2006; 22–31.
79. Andrzejak A, Graupner S, Kotov V, Trinks H. Algorithms for self-organization and adaptive service placement in dynamic distributed systems. *HPL-2002-259*, Hewlet Packard Corporation, 2002.
80. Weissman J, Lee B. The virtual service Grid: An architecture for delivering high-end network services. *Concurrency and Computation*: *Practice and Experience* 2002; **14**(4):287–319.
81. Verma D, Sahu S, Calo S, Shaikh A, Chang I, Acharya A. SRIRAM: A scalable resilient autonomic mesh. *IBM Systems Journal* 2003; **42**(1):19–28.
82. Valcarenghi L, Piero C. QoS-aware connection resilience for network-aware Grid computing fault tolerance. *Proceedings of the 2005 7th International Conference on Transparent Optical Networks*, July 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 417–422.
83. Lac C, Ramanathan S. A resilient telco Grid middleware. *Proceedings of the 11th IEEE Symposium on Computers and Communications*, June 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 306–311.
84. Andrieux A, Czajkowski K, Dan A, Keakey K, Ludwig H, Nakata T, Pruyne J, Rofrano J, Tuecke S, Xu M. Web services agreement specification (WS-Agreement). *GFD.107*, Open Grid Forum, May 2007.
85. Townend P, Groth P, Looker N, Xu J. FT-grid: A fault-tolerance system for e-science. *Proceedings of the Fourth UK e-Science All Hands Meeting*, September 2005. Engineering and Physical Sciences Research Council: Swindon, U.K., 2005.
86. Genaud S, Rattanapoka C. P2P-MPI: A peer-to-peer framework for robust execution of message passing parallel programs on Grids. *Journal of Grid Computing* 2007; **5**(1):27–42.
87. Abawajy J. Fault-tolerant scheduling policy for Grid computing systems. *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, April 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004; 238–244.
88. Budati K, Sonnek J, Chandra A, Weissman J. RIDGE: Combining reliability and performance in open Grid platforms. *Proceedings of the 16th International Symposium on High Performance Distributed Computing*, June 2007. ACM Press: New York, 2007; 55–64.
89. Zhang X, Junqueira F, Hiltunen M, Marzullo K, Schlichting R. Replicating nondeterministic services on Grid environments. *Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing*, June 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 105–116.
90. Lamport L. Paxos made simple. *ACM SIGACT News* (*Distributed Computing Column*) 2001; **32**(4):18–25.
91. Zhang X, Zagorodnov D, Hiltunen M, Marzullo K, Schlichting R. Fault–tolerant Grid services using primary-backup: Feasibility and performance. *Proceedings of the IEEE International Conference on Cluster Computing*, September 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004; 105–114.
92. Tuecke S, Czajkowski K, Foster I, Frey J, Graham S, Kesselman C, Maquire T, Sandholm T, Snelling D, Vanderbilt P (eds.). Open Grid Services Infrastructure (OGSI), version 1.0 (Proposed Recommendation). Open Grid Forum, July 2003.
93. *The Globus Toolkit*. http://www.globus.org/toolkit/ [4 August 2008].
94. Dasgupta G, Dasgupta K, Purohit A, Viswanathan B. QoS-GRAF: A framework for QoS based Grid resource allocation with failure provisioning. *Proceedings of the 14th IEEE International Workshop on Quality of Service*, June 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 281–283.
95. Huedo E, Montero R, Llorente I. A framework for adaptive execution in grids. *Software—Practice and Experience* 2004; **34**(7):631–651.
96. Huedo E, Montero R, Llorente I. Evaluating the reliability of computational Grids from the end user's point of view. *Journal of Systems Architecture* 2006; **52**(12):727–736.
97. In J, Avery P, Cavanaugh R, Chitnis L, Kulkarni M, Ranka S. SPHINX: A fault-tolerant system for scheduling in dynamic Grid environments. *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, April 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 12b.
98. IBM alphaWorks, Grid File Replication Manager. http://www.alphaworks.ibm.com/tech/gfrm [4 August 2008].
99. Oracle Corporation. *Data Grids and Service-oriented Architecture*, *an Oracle White Paper*. http://www.oracle.com/technologies/grid/index.html [4 August 2008].
100. Patterson D, Gibson G, Katz R. A case for redundant arrays of inexpensive disks (RAID). *Proceedings of the 1998 ACM SIGMOD International Conference on the Management of Data*, June 1988. ACM Press: New York, 1988; 109–116.

101. Joukov N, Rai A, Zadok E. Increasing distributed storage survivability with a stackable RAID-like file system. *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*, May 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 82–89.
102. Shrikumar H. De-layered Grid storage server. *SIGBED Review*, *Special Issue on the Second Workshop on High Performance*, *Fault Adaptive*, *Large Scale Embedded Real-time Systems* 2005; **2**(3):13–19.
103. Chaffin B. Oracle Implements Xserve RAID Internally; Endorses Apple. *The Mac Observer*, 6 December 2004. http://www.macobserver.com/article/2004/12/06.7.shtml [15 July 2008].
104. Chervenak A, Foster I, Kesselman C, Salisbury C, Tuecke S. The data Grid: Towards an architecture for the distributed management and analysis of large scientific data sets. *Journal of Network and Computer Applications* 2001; **23**:187–200.
105. Hoschek W, Jaen-Martinez J, Samar A, Stockinger H, Stockinger K. Data management in an international data grid project. *Proceedings of the 1st IEEE/ACM International Workshop on Grid Computing* (*Lecture Notes in Computer Science*, vol. 1971), Buyya R, Baker M (eds.). Springer: Berlin, 2000; 77–90.
106. Stockinger H, Samar A, Allcock B, Foster I, Holtman K, Tierney B. File and object replication in data Grids. *Proceedings of the 10th IEEE International Symposium on High Performance and Distributed Computing*, July 2001. IEEE Computer Society Press: Los Alamitos, CA, 2001; 76–86.
107. Bell W, Cameron D, Carvajal-Schiaffino R, Millar A, Stockinger K, Zini F. Evaluation of an economy-based file replication strategy for a data Grid. *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, May 2003. IEEE Computer Society Press: Los Alamitos, CA, 2003; 661–668.
108. Takizawa S, Takamiya Y, Nakada H, Matsuoka S. A scalable multi-replication framework for data Grid. *Proceedings of the 2005 Symposium on Applications and the Internet Workshops*, January 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 310–315.
109. Lui P, Wu J. Optimal replica placement strategy for hierarchical data Grid systems. *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, May 2006. IEEE Computer Society Press: Washington, DC, 2006; 417–420.
110. Deris M, Abawajy J, Suzuri H. An efficient replicated data access approach for large-scale distributed systems. *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*, April 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004; 588–594.
111. Lei M, Vrbsky S, Zijie Q. Online Grid replication optimizers to improve system reliability. *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, March 2007. IEEE Computer Society Press: Los Alamitos, CA, 2007; 1–8.
112. Chervenak A, Palavalli N, Bharathi S, Kesselman C, Schwartzkopf R. Performance and scalability of a replica location service. *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, June 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004; 182–191.
113. Chervenak A, Schuler R, Kesselman C, Koranda S, Moe B. Wide area data replication for scientific collaborations. *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, November 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 8.
114. Ripeanu M, Foster I. A decentralized, adaptive replica location mechanism. *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, June 2002. IEEE Computer Society Press: Washington, DC, 2002; 24–32.
115. Zhang Q, Yang J, Gu N, Zong Y, Ding Z, Zhang S. Dynamic replica location service supporting data Grid systems. *Proceedings of the Sixth IEEE International Conference on Computer and Information Technology*, September 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 61.
116. Demmy W, Petrini A. Statistical process control in software quality assurance. *Proceedings of the IEEE 1989 National Aerospace and Electronics Conference*, May 1989. IEEE Computer Society Press: Los Alamitos, CA, 2006; 1585–1590.
117. Research Triangle Institute. The Economic Impacts of Inadequate Infrastructure for Software Testing, May 2002.
118. Barton J, Czeck E, Segall Z, Siewiorek D. Fault injection experiments using FIAT. *IEEE Transactions on Computers* 1990; **39**(4):575–582.
119. Carreira J, Costa D, Silva J. Fault injection spot-checks computer system dependability. *IEEE Spectrum* 1999; **36**(8): 50–55.
120. Dawson S, Jahanian F, Mitton T. ORCHESTRA: A probing and fault injection environment for testing protocol implementations. *Proceedings of IEEE International Computer Performance and Dependability Symposium*, September 1996. IEEE Computer Society Press: Los Alamitos, CA, 2006; 404–414.
121. Kanawati G, Kanawati N, Abraham J. FERRARI: A flexible software-based fault and error injection system. *IEEE Transactions on Computers* 1995; **44**(2):248–260.
122. Voas J, Miller K. Using fault injection to assess software engineering standards. *Proceedings of the Second IEEE International Software Engineering Standards Symposium 'Experience and Practice'*, August 1995. IEEE Computer Society Press: Los Alamitos, CA, 1995; 139–145.

123. Madeira H, Costa D, Vieira M. On the emulation of software faults by software fault injection. *Proceedings of the International Conference on Dependable Systems and Networks*, June 2000. IEEE Computer Society Press: Los Alamitos, CA, 2000; 417–426.
124. Looker N, Burd L, Drummond S, Xu J, Munro M. Pedagogic data as a basis for web service fault models. *Proceedings of the IEEE International Workshop on Service-oriented System Engineering*, October 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 117–125.
125. Looker N, Munro M, Xu J. Determining the dependability of service-oriented architectures. *International Journal of Simulation and Process Modeling* 2007; **3**(1–2):88–97.
126. Reinecke P, van Moorsel A, Wolter K. The fast and the fair: A fault-injection-driven comparison of restart Oracles for reliable Web services. *Proceedings of the Third International Conference on the Quantitative Evaluation of Systems*, September 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 375–384.
127. Hoara W, Tixeuil S. A language-driven tool for fault injection in distributed systems. *Proceedings of the Sixth IEEE/ACM International Workshop on Grid Computing*, November 2005. IEEE Computer Society: Washington, DC, 2005; 8.
128. Monnet S, Bertier M. Using failure injection mechanisms to experiment and evaluate a Grid failure detector. *Proceedings of the Seventh International Conference on High Performance Computing for Computational Science* (*Lecture Notes in Computer Science*, vol. 4395), Dayde M, Palma J, Coutinho A, Pacitti E, Lopes J (eds.). Springer: Berlin, 2007; 610–621.
129. Kharchenko V, Popov P, Romanovsky A. On dependability of composite Web services with components upgraded online. *Proceedings of the International Conference on Dependable Systems and Networks* (*Lecture Notes in Computer Science*, vol. 3549), de Lemos R, Gack C, Romanovsky A (eds.). Springer: Berlin, 2005; 92–121.
130. Song C, Topkara U, Woo J, Park S. Reliability assessment of Grid software systems using emergent features. *Second Workshop on Reliability and Robustness in Grid Computing Systems*, January 2007, Open Grid Forum. http://www.ogf.org [14 July 2008].
131. Topkara U, Song C, Woo J, Park S. Connected in a small world: Rapid integration of biological resources. *Proceedings of the Grid Computing Environments Workshop*, November 2006. http://umut.topkara.org/papers/research_work.html [4 August 2008].
132. Bitonti L, Kiss T, Terstyanszky G, Delaitre T, Winter S, Kacsuk P. Dynamic testing of legacy code resources on the Grid. *Proceedings of the 3rd Conference on Computing Frontiers*, May 2006. ACM Press: New York, 2006; 261–268.
133. Chun G, Dail H, Casanova H, Snavely A. Benchmark probes for Grid assessment. *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, April 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004; 276–283.
134. Chang B, Crary K, Delap M, Harper R, Liszka J, Murphy T, Pfenning F. Trustless Grid computing in ConCert. *Proceedings of the Third International Workshop on Grid Computing* (*Lecture Notes in Computer Science*, vol. 2536), Goos G, Hartmanis J, van Leeuwen J (eds.). Springer: Berlin, 2002; 112–125.
135. Vanderwaart J, Crary K. Automated and certified conformance to responsiveness policies. *Proceedings of the ACM SIGPLAN Workshop on Types in Language Design and Implementation*, January 2005. ACM Press: New York, 2005; 79–90.
136. Munson J, Khoshgoftaar T. The detection of fault-prone programs. *IEEE Transactions on Software Engineering* 1992; **18**(5):423–433.
137. Graves T, Karr A, Marron J, Siy H. Predicting fault incidence using software change history. *IEEE Transactions on Software Engineering* 2000; **26**(7):653–661.
138. Ostrand T, Weyuker E, Bell R. Predicting the location and number of faults in large software systems. *IEEE Transactions on Software Engineering* 2005; **31**(4):340–355.
139. Weyuker E, Ostrand T, Bell R. Adapting a fault prediction model to allow widespread usage. *Proceedings of the Second International Promise Workshop*, September 2006. http://promisedata.org/repository/papers.html [14 July 2008].
140. Ceri S, Grefen P, Sanchez G. WIDE—A distributed architecture for workflow management. *Proceedings of the Seventh International Workshop on Research Issues in Data Engineering*, April 1997. IEEE Computer Society Press: Los Alamitos, CA, 1997; 76–79.
141. Borgida A, Murata T. Tolerating exceptions in workflows: A unified framework for data and processes. *Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration*, February 1999. ACM Press: New York, 1999; 59–68.
142. Hagan C, Alonso G. Exception handling in workflow management systems. *IEEE Transactions on Software Engineering* 2000; **26**(10):943–958.
143. Cardoso J, Luo Z, Miller J, Sheth A, Kochut K. Survivability architecture for workflow management systems. *Proceedings of the 39th Annual ACM Southeast Conference*, March 2001. ACM Press: New York, 1999; 207–216.
144. Alonso G, Hagen C, Agrawal D, El Abbadi A, Mohan C. Enhancing the fault tolerance of workflow management systems. *IEEE Concurrency* 2000; **8**(3):74–81.
145. Stone N, Simmel D, Kielmann T. An architecture for grid checkpoint and recovery (GridCPR) services and a GridCPR application programming interface (draft document). Open Grid Forum, September 2005.

146. Hwang S, Kesselman C. GridWorkflow: A flexible failure handling framework for the Grid. *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, June 2003. IEEE Computer Society Press: Los Alamitos, CA, 2003; 126–137.
147. Yu J, Buyya R. A taxonomy of workflow management systems for Grid computing. *Technical Report GRIDS-TR-2005-1*, University of Melbourne, Australia, March 2005.
148. Condor DAGMan. http://www.cs.wisc.edu/condor/dagman/ [4 August 2008].
149. Deelman E, Blythe J, Gil Y, Kesselman C, Mehta G, Vahi K, Blackburn K, Lazzarini A, Arbree A, Cavanaugh R, Koranda S. Mapping abstract complex workflows onto Grid environments. *Journal of Grid Computing* 2003; **1**(1):25–39.
150. Altintas I, Birnbaum A, Baldridge K, Sudholt W, Miller M, Amoreira C, Potier Y, Ludaescher B. A framework for the design and reuse of Grid workflows. *Proceedings of the First International Workshop on Scientific Applications of Grid Computing* (*Lecture Notes on Computer Science*, vol. 3458), Herraro P, Perez M, Robles V (eds.). Springer: Berlin, 2005; 119–132.
151. von Laszewski G, Hategan M. Java CoG Kit Karajan/GridAnt workflow guide. *Technical Report*, Argonne National Laboratory, Argonne, IL, 2005.
152. Fahringer T, Jugravu A, Pllana S, Prodan R, Seragiotto C, Truong H. ASKALON: A tool set for cluster and Grid computing. *Concurrency and Computation*: *Practice and Experience* 2005; **17**(2–4):143–169.
153. Oinn T, Addis M, Ferris J, Marvin D, Senger M, Greenwood M, Carver T, Glover K, Pocock M, Wipat A, Li P. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 2004; **20**(17):3045–3054.
154. Ra D, Shakeb A, Gupta I, Harkik D, Upadhyay A, Alves L, Damodaran A, Chakrabarti A, Ghosh A. Scalable enterprise level workflow manager for the Grid. *Proceedings of the Fifth International Conference on Quality Software*, September 2005. IEEE Computer Society Press: Los Alamitos, CA, 2006; 341–348.
155. Yu J, Buyya R. A novel architecture for realizing Grid workflow using tuple spaces. *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, November 2004. IEEE Computer Society Press: Los Alamitos, CA, 2005; 119–128.
156. Alves A, Arkin A, Askary S, Bloch B, Curbera F, Goland Y, Kartha N, Liu C, Konig D, Mehta V, Thatte S, van der Rijn D, Yendluri P, Yiu A (eds.). Web Services Business Process Execution Language, version 2, Organization for the Advancement of Structured Information Standards (OASIS), April 2007.
157. Emmerich W, Butchart B, Chen L, Wassermann B, Price S. Grid service orchestration using the business process execution language (BPEL). *Journal of Grid Computing* 2006; **3**(3):283–304.
158. Cybok D. A Grid workflow infrastructure. *Concurrency and Computation*: *Practice and Experience* 2006; **18**(10): 1243–1254.
159. Turner K, Tan K. Graphical composition of Grid services. *Proceedings of the Third International Workshop on Rapid Integration of Software Engineering Techniques* (*Lecture Notes in Computer Science*, vol. 4401), Guelfi N, Buchs D (eds.). Springer: Berlin, 2007; 1–17.
160. Tartanoglu F, Issarny V, Romanovsky A, Levy N. Coordinated forward error recovery for composite Web services. *Proceedings of the 22nd International Symposium on Reliable Distributed Systems*, October 2003. IEEE Computer Society Press: Los Alamitos, CA, 2003; 167–176.
161. Wassermann B, Emmerich W. Reliable scientific service compositions. *Proceedings of the 4th International Conference on Service-oriented Computing* (*Lecture Notes in Computer Science*, vol. 4652), Georgakopoulos D, Ritter N, Benatallah B, Zirpins C, Feuerlicht G, Schonherr M, Motahari N (eds.). Springer: Berlin, 2007; 14–25.
162. Feingold M, Jeyaraman R. Web Services Coordination (WS-Coordination), version 1.1, Organization for the Advancement of Structured Information Standards (OASIS), April 2007.
163. Furniss P (ed.). Business Transaction Protocol, version 1.1.0 (Committee Draft), Organization for the Advancement of Structured Information Standards (OASIS), November 2004.
164. Fox G, Guha R, McMullen D, Mustacoglu A, Pierce M, Topcu A, Wild D. Web 2.0 for Grids and e-Science. *Proceedings of the Second International Workshop on Distributed Cooperative Laboratories* (*Grid Enabled Remote Instrumentation*), Davoli F, Meyer N, Pugliese R, Zappatore S (eds.). Springer: New York, 2008; 409–431.
165. Topcu A, Mustacoglu A, Fox G, Cami A. Integration of collaborative information systems in Web 2.0. *Proceedings of the Third International Conference on Semantics*, *Knowledge and Grid*, October 2007. IEEE Computer Society Press: Los Alamitos, CA, 2007; 523–526.
166. Krauter K, Buyya R, Maheswaran M. A taxonomy and survey of Grid resource management systems for distributed computing. *Software—Practice and Experience* 2002; **32**(2):135–164.
167. Massie M, Chun B, Culler D. Ganglia distributed monitoring system: Design, implementation, experience. *Parallel Computing* 2004; **30**(7):817–840.
168. Iosup A, Tapu N, Vialle S. A monitoring architecture for control grids. *Proceedings of the European Grid Conference* (*Lecture Notes in Computer Science*, vol. 3470), Sloot P, Hoekstra A, Priol T, Reinefeld A, Bubak M (eds.). Springer: Berlin, 2005; 922–931.
169. Byrom R, Coghlan B, Cooke A, Cordenonsi R, Cornwall L, Craig M, Djaoui A, Duncan A, Fisher S, Gray A, Hicks S, Kenny S, Leake J, Lyttleton O, Magowan J, Middleton R, Nutt W, O'Callaghan D, Podhorszki N, Taylor P, Walk J,

Wilson A. Fault tolerance in the R-GMA information and monitoring system. *Proceedings of Advances in Grid Computing European Grid Conference* (*Lecture Notes in Computer Science*, vol. 3470), Sloot P, Hoekstra A, Priol T, Reinefeld A, Bubak M (eds.). Springer: Berlin, 2005; 751–760.

170. Juhasz Z, Andics A, Pota S. Towards a robust and fault-tolerant discovery architecture for global computing grids. *Scalable Computing*: *Practice and Experience* 2003; **6**(2):22–33.
171. Arnold K, Wollrath A, Scheifler R, Waldo J. *The Jini Specification*, *V1.0*. Addison-Wesley: Boston, 1999.
172. MacLaren J, Keown M, Pickles S. Co-allocation, fault tolerance, Grid computing. *Proceedings of the UK e-Science All Hands Meeting*, September 2006. Engineering and Physical Sciences Research Council: Swindon, U.K., 2006; 155–162.
173. Gray J, Lamport L. Consensus on transaction commit. *MSR-TR-2003-96*, Microsoft Research Corporation, 2004.
174. Hiltunen M, Schlichting R, Ugarte C. Enhancing survivability of security services using redundancy. *Proceedings of the International Conference on Dependable Systems and Networks*, July 2001. IEEE Computer Society Press: Los Alamitos, CA, 2001; 173–182.
175. Shang L, Wang Z, Zhou X, Huang X, Cheng Y. TM-DG: A trust model based on computer users' daily behavior for desktop grid platform. *Proceedings of the 2007 Symposium on Component and Framework Technology in High-performance and Scientific Computing*, October 2007. ACM Press: New York, 2007; 59–66.
176. Shafer G. *A Mathematical Theory of Evidence*. Princeton University Press: Princeton, NJ, 1976.
177. Colling D, Ferrari T, Hassoun Y, Huang C, Kotsokalis C, McGough A, Patel Y, Ronchieri E, Tsanakas P. On quality of service support for grid computing. *Proceedings of the Second International Workshop on Distributed Cooperative Laboratories* (*Grid Enabled Remote Instrumentation*), Davoli F, Meyer N, Pugliese R, Zappatore S (eds.). Springer: New York, 2008; 313–327.
178. He E, Vicat-Blanc P, Weizl M. A survey of transport protocols other than 'standard' TCP. *Informational Document GFD-I.055*, Open Grid Forum, November 2005.
179. Davis D, Karmarkar A, Pilz G, Winkler S, Yalcinalp U (eds.). Web services reliable messaging (WS-ReliableMessaging), Organization for the Advancement of Structured Information Standards (OASIS), June 2007.
180. Iwasa K, Durand J, Rutt T, Peel M, Kunisetty S, Bunting D (eds.). WS-Reliability 1.1, Organization for the Advancement of Structured Information Standards (OASIS), November 2004.
181. Pallickara S, Fox G, Pallickara S. An analysis of reliable delivery specifications for web services. *Proceedings of the International Conference on Information Technology*: *Coding and Computing*, April 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 360–365.
182. Tai S, Mikalsen T, Rouvellou I. Using message-oriented middleware for reliable web services messaging. *Proceedings of the Second International Workshop on Web-services*, *E-Business and the Semantic Web* (*Lecture Notes on Computer Science*, vol. 3095), Bussler C, Fensel D, Orlowska M, Yang J (eds.). Springer: London, 2004; 89–104.
183. Mandrichenko I, Allcock W, Perelmutov T. GridFTP v2 protocol description. *GFD-R-P.047*, Open Grid Forum, May 2005.
184. Postel J, Reynolds J. File transfer protocol. *RFC 959*, Internet Engineering Task Force, October 1985.
185. Mattmann C, Kelly S, Crichton D, Hughes J, Hardman S, Ramirez P, Joyner R. A classification and evaluation of data movement technologies for the delivery of highly voluminous scientific data products. *Document 20060044153*, National Aeronautics and Space Administration, Jet Propulsion Laboratory, May 2006.
186. Lowekamp B, Tierney B, Cottrell L, Hughes-Jones R, Kielmann T, Swany M. A hierarchy of network performance characteristics for grid applications and services (proposed recommendation). *GFD-R-P.023*, Open Grid Forum, May 2004.
187. Lowekamp B, Hughes-Jones R, Tierney B, Kielmann T, Cottrell L, Swany M. Enabling network measurement portability through a hierarchy of characteristics. *Proceedings of the Fourth International Workshop on Grid Computing*, November 2003. IEEE Computer Society Press: Washington, DC, 2003; 68–75.
188. Fox G, Pallickara S, Pierce M, Gadgil H. Building messaging substrates for web and grid applications. *Philosophical Transactions of the Royal Society*: *Mathematical*, *Physical and Engineering Sciences* (*Scientific Applications of Grid Computing Special Issue*) 2005; **363**(1833):1757–1773.
189. Fox G, Aydin G, Bulut H, Gadgil H, Pallickara S, Pierce M, Wu W. Management of real-time streaming data grid services. *Concurrency and Computation*: *Practice and Experience* 2007; **19**(7):983–998.
190. Gudgin M, Hadley M, Mendelsohn N, Moreau J, Nielsen H, Karmarkar A, Lafon Y. SOAP version 1.2, Part 1: messaging framework (2nd edn), World Wide Web Consortium, April 2007.
191. Gudgin M, Hadley M, Rogers T. Web services addressing, version 1.0, World Wide Web Consortium, May 2006.
192. Box D, Cabrera L, Critchley C, Curbera F, Ferguson D, Graham S, Hull D, Kakivaya G, Lewis A, Lovering B, Niblett P, Orchard D, Samdarshi S, Schlimmer J, Sedukhin I, Shewchuk J, Weerawarana S, Wortendyke D. Web services eventing (WS-Eventing), (draft submission), World Wide Web Consortium, March 2006.
193. Gadgil H, Fox G, Pallickrara S, Pierce M. Fault-tolerant management of grid services. *Proceedings of the 2007 IEEE International Conference on Cluster Computing*, September 2007. IEEE Computer Society Press: Los Alamitos, CA, 2007.

194. McCollum R, Murray B, Reistad B (eds.). Web services for management (WS-Management), version 1.0.0a, Distributed Management Task Force, April 2006.
195. Lim S, Fox G, Pallickara S, Pierce M. Web service robust GridFTP. *Proceedings of the 2004 International Multiconference in Computer Science and Computer Engineering*, June 2004. Computer Science Research, Education, and Applications (CSREA) Press: Irvine, CA, U.S.A., 2004; 725–730.
196. Kosar T, Kola G, Livny M. Data pipelines: Enabling large scale multi-protocol data transfers. *Proceedings of the 2nd Workshop on Middleware for Grid Computing*, October 2004. ACM Press: New York, 2004; 63–68.
197. Maassen J, Bal H. Smartsockets: Solving the connectivity problems in grid computing. *Proceedings of the 16th International Symposium on High Performance Distributed Computing*, June 2007. ACM Press: New York, 2007; 1–10.
198. Awduche D, Chiu A, Elwalid A, Widjaja I, Xiao X. Overview and principles of internet traffic engineering. *RFC 3272*, Internet Engineering Task Force, May 2002.
199. Moy J. OSPF, version 2. *RFC 2328*, Internet Engineering Task Force, December 1999.
200. Information technology—Telecommunications and information exchange between systems—Intermediate system to intermediate system intra-domain routing information exchange protocol. *ISO/IEC 10589*, International Organization for Standardization, 2002.
201. Rekhter Y (ed.). A border gateway protocol 4 (BGP-4). *RFC 1771*, Internet Engineering Task Force, March 1995.
202. Rosen E, Viswanathan A, Callon R. Multiprotocol label switching architecture. *RFC 3031*, Internet Engineering Task Force, January 2001.
203. Boyle J, Gill V, Hannan A, Cooper D, Awduche D, Christian B, Lai W. Applicability statement for traffic engineering with MPLS. *RFC 3346*, Internet Engineering Task Force, August 2002.
204. Clapp G, Gannet J, Skoog R. Requirements and design of a dynamic grid networking layer. *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*, April 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004; 633–639.
205. TeraGrid. http://www.teragrid.org/ [4 August 2008].
206. World's Fastest Network Launched to Connect TeraGrid Sites. *PSC News Center*, 27 February 2003. http://www.psc.edu/publicinfo/news/2003/ [4 August 2008].
207. AccessGrid Home Page. http://www.accessgrid.org/ [4 August 2008].
208. Levine N, Garcia-Luna-Aceves J. A comparison of reliable multicast protocols. *Multimedia Systems* 1998; **6**(5):334–348.
209. Mankin A, Romanow A, Bradner S, Paxson V. IETF criteria for evaluating reliable multicast transport and application protocols. *RFC 2357*, Internet Engineering Task Force, January 1998.
210. Popescu A, Constantinescu D, Erman D, Ilie D. A survey of reliable multicast communication. *Proceedings of the Third EuroNGI Conference on Next Generation Internet Networks*, May 2007. IEEE Computer Society Press: Los Alamitos, CA, 2007; 111–118.
211. Floyd S, Jacobson V, Liu C, McCanne S, Zhang L. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking* 1997; **5**(6):784–803.
212. Paul S, Sabnani K, Lin J, Bhattacharyya S. Reliable multicast transport protocol (RMTP). *IEEE Journal on Selected Areas in Communications* 1997; **15**(3):407–421.
213. Adamson B, Bormann C, Handley M, Macker J. NACK-oriented reliable multicast (NORM) protocol (Draft), Internet Engineering Task Force, March 2007.
214. Nekovee M, Barcellos M, Daw M. Reliable multicast for the grid: A case study in experimental computer science. *Philosophical Transactions of the Royal Society*, *Series A* 2005; **10**(1098):1775–1791.
215. Macker J. The multicast dissemination protocol (MDP) toolkit. *Proceedings of the IEEE Military Communications Conference*, November 1999. IEEE Computer Society Press: Los Alamitos, CA, 1999; 626–630.
216. Waters G, Crawford J, Lim S. Optimising multicast structures for grid computing. *Computer Communications* 2004; **27**(14):1389–1400.
217. Ranaldo N, Tretola G, Zimeo E. Hierarchical and reliable multicast communication for Grid systems. *Proceedings of the International Conference on Current and Future Issues of High-End Computing*, June 2005. Central Institute for Applied Mathematics: Jülich, Germany, 2005; 137–144.
218. Chiu D, Hurst S, Kadansky M, Wesley J. TRAM: A tree-based reliable multicast protocol. *SMLI TR-98-66*, Sun Microsystems Laboratories, July 1998.
219. Reliable Multicast Transport (rmt) Working Group (Internet Engineering Task Force). http://www.ietf.org/html.charters/rmt-charter.html [4 August 2008].
220. Nguyen-Tuong A, Grimshaw A, Wasson G, Humphrey M, Knight J. Towards dependable grids. *Technical Report CS-2004-11*, Department of Computer Science, University of Virginia, 2004.
221. Bezzine S, Galtier V, Vialle S, Baude F, Bossy M, Doan V, Henrio L. A fault tolerant and multi-paradigm grid architecture for time constrained problems: Application to option pricing in finance. *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, December 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 49.
222. Xie M, Dai Y, Poh K. *Computing Systems Reliability*. Kluwer Academic Publishers: New York, NY, 2004.

223. Lin M, Chang M, Chen D. Distributed-program reliability analysis: Complexity and efficient algorithms. *IEEE Transactions on Reliability* 1999; **48**(1):87–95.
224. Dai Y, Pan Y, Zou X. A hierarchical modeling and analysis for grid service reliability. *IEEE Transactions on Computers* 2007; **56**(5):681–691.
225. Dai Y, Levitin G. Reliability and performance of tree-structured grid services. *IEEE Transactions on Reliability* 2006; **55**(2):337–349.
226. Dai Y, Levitin G. Optimal resource allocation for maximizing performance and reliability in tree-structured grid services. *IEEE Transactions on Reliability* 2007; **56**(3):444–453.
227. Levitin G, Dai Y, Ben-Haim H. Reliability and performance of star topology grid service with precedence constraints on subtask execution. *IEEE Transactions on Reliability* 2006; **55**(3):507–515.
228. Liu X, Xia H, Chien A. Validating and scaling the microgrid: A scientific instrument for grid dynamics. *Journal of Grid Computing* 2004; **2**(2):141–161.
229. Mills K, Dabrowski C. Investigating global behavior in computing grids. *Proceedings of the Second International Workshop on Self-Organizing Systems* (*Lecture Notes in Computer Science*, vol. 4124), De Meer H, Sterbenz J (eds.). Springer: Berlin, 2006; 120–136.
230. Mills K, Dabrowski C. Can economics-based resource allocation prove effective in a computation marketplace? *Journal of Grid Computing* 2008; **6**(3):291–311.
231. Dabrowski C. Investigating resource allocation in a standards-based grid compute economy. *Interagency Report 7463*, National Institute of Standards and Technology, Gaithersburg, MD, November 2007.