Reliability in Grid Computing Systems

Christopher Dabrowski
National Institute of Standards and Technology
100 Bureau Drive, Stop 8970
Gaithersburg, MD 20899-8970
Phone: +1 301 975-3249
FAX: +1 301 948-6213
cdabrowski@nist.gov

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 reliability.

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

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 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 [1], but other standards development organizations have also produced specifications, such as [2, 3], that are being used in grid systems. To continue to 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 grow in scale, and become more heterogeneous and dynamic in composition.

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 though, the body of work on grid reliability produced by researchers and practitioners in academe and industry[1] 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 and identifies important issues that researchers are working to resolve in order to develop reliability methods for such grid environments. Similarly, the survey illuminates reliability issues relating to 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 to develop reliability methods for large-scale, heterogeneous, dynamic, grid environments are still in progress. 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.

[1] 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 not 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.

Perhaps the main contribution of the paper is to show that while there has been substantial progress in these functional areas, scalable reliability solutions for grid systems thus far developed 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 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 characteristics of grid system environments which tend toward higher likelihood of failures and partly due to the existence of redundant resources in grid systems, which provide opportunities to switch to functioning resources when failures occur. Despite the emphasis on fault tolerance, work has also begun on testing techniques to find and remove faults in grid systems software. With the gradual growth in scale of grid systems, researchers are also beginning to realize the importance of system-level approaches to improving reliability that consider techniques such as complex systems analysis.

To date there has been no 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 appropriate that such a survey 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 discusses the relationship between reliability work in grid computing and other areas of distributed systems. Section 3 surveys research on improving reliability of grid computing resources. Section 4 discusses work on providing fault tolerance from within user applications running in grid systems, focusing on applications in which tasks are logically organized into workflows. Section 5 addresses reliability work in core infrastructure and management services. Section 6 considers efforts to make grid networks more reliable. Section 7 discusses approaches for analysis of grid reliability from an overall system-level perspective. Section 8 concludes, and is followed by acknowledgements and references. In the appendix, individual references are cross-referenced by subject area. Finally, it is important to note areas that are out of the scope in this study. These include security, an extensive subject best treated separately, reliability oh hardware components, and physical site integrity, which are also studied elsewhere.

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 enterprises 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 grid services are reliably provided under conditions of scale, heterogeneity, and dynamism. Large scale in computing and data resources is already in evidence in some systems, such as [8, 9]. However in addition to large 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 enterprises 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 software. 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, as discussed in [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, which transport large datasets between grid resources. To do this, network services will need to coordinate many heterogeneous network components and maintain stable high-bandwidth connections for long periods [14], thus increasing 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 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 the distinguishing characteristics of grid systems, 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 in comparison with grid computing, 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 perhaps been less extensively researched [26]. The efforts described in these surveys provide a basis for grid reliability research, and where appropriate these links are discussed in this study.

3. RELIABILITY OF GRID RESOURCES

Because of its obvious importance, ensuring the reliability of computing hardware and software resources that comprise grid systems has probably been the object of more effort than 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 can also include dedicated software components that may be engaged by users to perform various functions, such as

data mining and other analysis. A third category of grid resources are data stores used in data and multimedia grids.

To date, developing methods to ensure reliability of grid resources in both research and commercial systems has 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. Fault-tolerance methods deployed in today's operational commercial and science grid systems are based on methods available in current distributed systems technology. When applied by capable engineers, these methods can and do provide fault tolerance for systems at current scale levels [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. Though there has been strong emphasis on fault tolerance, work has also been done on test methods for fault removal, which is also described in this section.

3.1 Detecting Faults and Failures in Grid Resources

Building on previous work on fault and failure detection for distributed systems [27-29], researchers have investigated scalable fault detection methods for grid systems environments. Work has also focused on techniques to recognize different types of faults, particularly hard-to-detect faults, which are expected to occur among heterogeneous resources under dynamic conditions, including fault isolation and diagnosis methods. To date, most failure detectors developed for scaled conditions 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, dynamic, heterogeneous grid systems described in section 2. One reason is that available network monitoring protocols and tools, such as those based on SNMP [30], rely on detailed knowledge of network structure. Such information is unlikely to be always available in large-scale dynamic environments divided into different administrative and security domains. Another well-known failure detection problem occurs in asynchronous distributed systems where management functions (including failure detectors) are decentralized and themselves may be subject to failure. Such conditions are likely to be found in large-scale grid systems. Under these conditions, the work of Fischer, Lynch, and Paterson [31] 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, 31], Chandra and Toueg [32] 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 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, 31, 32] identified an important barrier that needed to be overcome to achieve practical, scalable failure detection.

The work of [32] led to development of experimental failure detectors for distributed systems that sought to guarantee completeness, but were only probabilistically accurate [29]. These systems detected failure deterministically, using heartbeat techniques in which

resources regularly sent messages (heartbeats) to other members of a heartbeat group to indicate they had not failed. Group members then used absence of heartbeat message to reach consensus on which members failed. In [29], failure detection based on this approach was shown to have limited scalability, while approaches that used centralized monitoring by a subset of components were subject to message bottlenecks and other anomalies that degraded performance. Looking at other factors that effected grid system scalability, Hayashibara and colleagues [33] also concluded failure detection methods used in current distributed systems would not be effective in the face of scale, message explosion, dynamic resource composition, and diversity and variability of user applications that are expected in grid environments. Current research grid systems also reflect the lack of scalable fault detection capabilities. Zanikolas and Sakellariou [34] conduced a survey of 19 grid monitoring systems that were based on the OGF *Grid Monitoring Architecture* [35], which prescribes requirements for a monitoring function to support fault detection. The survey found most of these systems were designed without dedicated fault detection mechanisms. Further, most did not have the potential to scale their monitoring functions, thus also impeding scalability of fault detector functions that may rely on them. These shortcomings in failure detection methods developed for previous distributed systems motivated research into detectors for large-scale grids.

Research on scalable fault detection methods. Early work on a distributed fault detector for the Globus system [36] addressed issues of completeness and accuracy [32] by allowing potentially unreliable failure detectors to report information about the likelihood of resource failure, which applications interpret at their own discretion. The approach was also designed to improve flexibility, efficiency, and scalability by decoupling monitoring, detection, and notification functions. This work was followed by other experimental failure detectors that built on [29, 32], of which some examples are presented. In [37], 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 Internet addresses. Leader nodes, or monitors to which heartbeats were sent, were made redundant for fault-tolerance purposes. To maintain an efficient overall structure, heartbeat groups could divide if their size increased, or divide if it decreased. 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. Viability of this approach was also demonstrated in small-scale testbed experiments. Horita and his associates [38] proposed a scalable, self-organizing fault-detection system based on previous work on group membership protocols and fault detectors for distributed systems [29, 39]. In this approach, each process was monitored by a small group (4 or 5) of other 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, which may consist of heterogeneous resource types. Detection of connection failure caused notifications to be propagated through the monitoring network. Experiment results showed scalability for groups of node clusters, where individual clusters had hundreds of resources to monitor (approximating grid sites containing LANs). In [40], resources assigned to an application were placed in separate domains where they emitted heartbeats to a domain monitor; here, monitoring domains were organized hierarchically for scalability. Despite

these and other promising endeavors, failure detectors that follow this line of work have been largely deployed only experimentally in testbeds of moderate scale.

Detecting Different Types of Faults. An important aspect of guaranteeing completeness and accuracy is the ability to recognize different types of faults that lead to failures. Taxonomies of fault types related to grid system environments have been produced by [41-43]. Early work on a fault handling framework that could recognize, different kinds of failures during simulated grid operations was reported in [16]. This system could initiate different recovery actions designed to remedy different fault types. Jitsumoto and colleagues [44] also developed a detector that differentiated between hardware, process, and transmission faults. Here, users were allowed to pre-select a recovery procedure to be invoked in response to occurrence of a particular fault type. The efficiency of the approach was demonstrated in a testbed. In [45], a method was proposed to detect and predict different faults types using data mining techniques and a fault classification scheme. In [46], a fault detection and recovery method for transient process faults was reported. Here an adaptive scheme was used to periodically checkpoint (i.e., store the state of) replicated processes that execute in parallel. Checkpointed process states were then compared to discover erroneous computations affected by transient faults. Because this procedure was compute intensive, the frequency of observed faults was used to dynamically vary the checkpoint interval in order to minimize delay in process execution time. Jin and colleagues [47] developed a hierarchical grid failure detector and failure handler that adapted to user requirements and system conditions. Testbed experiments showed this approach scaled up to 1000 components at 2 sites. In the EU Datagrid Project, [48] described an adaptive mechanism for detecting and responding to different fault and failure types in large-scale, heterogeneous environments, while [49] reported results of a long-term study of fault patterns 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 and others [13] reported work on developing a model of "silent" fault types in the Condor system, which characteristically do not immediately indicate their presence after they occur. Work has also been reported on using consensus-based algorithms to detect Byzantine faults. Byzantine faults 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. This class of faults cannot easily be traced to failed links, processes, and messages. In [10, 11], replicated components compared results of identical simultaneously executing computations to detect components affected by Byzantine faults. Because Byzantine faults are hard-to-detect and recurring, they can disrupt many computations over time.

Finally, in large distributed systems, faults may originate at one component and propagate across the system, potentially creating dangerous cascading effects. Thus far, it appears that few researchers have addressed this class of faults in grid systems. One example is [50], in which a diagnostic approach was proposed for executing real-time tests on interdependent grid components in order to isolate the origin of a fault and determine recovery actions. Related work was reported by [12] on methods to isolate faults that originate at a particular component and propagate across a grid. As with hard-to-detect faults, frequent cascading failures can lower user confidence in a grid system over time. Therefore, continuing efforts in

developing scalable fault detection and isolation methods that correctly and accurately diagnose hard-to-detect 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 an executing process or a resource. Both temporal and spatial redundancy is used in grids. However, because grid systems inherently provide redundant resources, spatial redundancy has been a focus of fault-tolerance research, and so is the main topic of this section. Three techniques for spatial redundancy have been of particular importance: (1) *checkpointing*, or periodically saving the state of a process running on a computational resource so that, in the event of failure, it can be migrated to an operational resource (2) *replication*, or maintaining a sufficient number of replicas, or copies, of a process executing in parallel with identical state but on different resources, so that at least one replica is guaranteed to finish the process correctly, (3) in the event of failure, *rescheduling*, or finding different resources to that can accept and run failed tasks. The section also discusses work on replication of data, an important aspect of providing 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, 44, 47]. However, if the host has failed, the process may be *migrated*, or transferred, to a different execution environment. The migration of a process that is unable to continue on its original processor to a new processor is known 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. Then the section discusses methods being developed for grid environments having greater scale and resource heterogeneity. One important issue is finding efficient methods for checkpointing many concurrent, intercommunicating processes, so that in the event of failure, they can resume processing from a common saved state [22]. Another important issue is process migration across dissimilar computing platforms or administrative domains. Checkpoint and process migration can be initiated either from within grid systems or within applications. This section considers the former. Section 6 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 literature on this subject precedes grid computing, as surveyed in [21]. Many currently deployed grid systems that manage interconnected computing clusters employ checkpoint and process migration techniques for a set of parallel processes which are

based on methods used in high-performance cluster computing. Examples are commercial grid products assembled from cluster computing components such as [51-56]. These systems also provide recovery for server managers, or cluster head nodes, that manage concurrently executing processes. For instance, [55] 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 can preserve a logical structure in which a manager continues to supervise the remaining compute nodes which host parallel processes.

Research grids and grids used for science applications also manage clusters and employ checkpoint and process migration techniques based on high-performance computing. Early efforts in using checkpoint and process migration in large-scale grids were reported in the Legion system [57] and Cactus [58, 59]. In the HA-OSCAR research grid system [60], fault tolerance in cluster head nodes is improved by taking checkpoints of job-queue information and regularly updating a backup server. If the primary server fails, the backup has access to up-to-date job-queue information. Testbed experiments demonstrated faster restart of in-progress jobs using this approach. The Condor distributed processing system [61] also provides site server fault tolerance by replicating servers and employing process migration when the primary fails.

Research in Checkpointing Methods. An important research issue for grid systems is the ability to provide efficient and scalable checkpointing of concurrent, intercommunicating processes in situations where process 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 processes is in-transit, the resulting saved states may be inconsistent. Three checkpointing strategies have been considered for concurrent processes in a survey by Elnozany et al. [22]. In *coordinated checkpointing,* processes attempt to synchronize checkpoints to ensure individual saved states are consistent with respect to each other, and that the overall combined, saved state is also consistent. Taking coordinated checkpoints requires correctly accounting for messages that are in-transit when the synchronized 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 rollback and recover a previous common, consistent state may be subject to a repeating *domino effect* during rollback. 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 had also sent messages earlier, additional processes may also require rollback. If the effect continues, the processes involved resemble falling dominos as they repeatedly cause rollbacks. A third strategy, *communications induced checkpointing*, attempts to selectively coordinate some checkpoints to avoid taking *useless* checkpoints. However, it is thought that this method may not be scalable for grid environments [62].

Coordinated checkpointing has thus far received the most attention as a recovery mechanism for concurrent, intercommunicating processes. The study by [22] found that in distributed systems environments, coordinate checkpointing procedures that blocked (halted) processes during the synchronized checkpointing operation degraded performance, while

approaches 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) specification [63], an important specification for enabling communication between concurrent processes. Buntinas et al., [64] also compared an MPI-based, blocking and non-blocking coordinated checkpoint protocols for efficiency and scalability in an experimental grid environment. Here, the results also indicated that blocking reduced efficiency, while a non-blocking approach appeared to scale well but suffered from implementation issues. Coordinated checkpointing has been implemented in MPI, in LAM/MPI [65] as well as in [44, 62, 66, 67]. Yeom et al [66, 67] proposed a fault-tolerant version of MPI, MPICH-GF, for grid systems that employed coordinated checkpoints with blocking. In [68], this approach was extended to incorporate file recovery using a versioned file system. Other MPI extensions that employed coordinated checkpointing and process blocking were proposed for grid environments by Bouteiller and colleagues [62] and in [44], where performance was demonstrated under conditions of limited scale.

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, messages are stored on a stable media. Logging operates under the *piecewise deterministic assumption* [69] which states that information about non-deterministic events (messages) can be logged and replayed to recreate and recover a lost state. During recovery of a failed process, messages that occurred after a checkpoint can be replayed to recreate a pre-failure state that is consistent with the states of other concurrent processes, rather than initiating additional rollbacks to find a consistent state. Different logging methods were surveyed in [22] and found to ensure recovery to different degrees at the cost of slowing execution when storing messages or carrying out recovery. Use of uncoordinated checkpointing in combination with different logging techniques in MPI systems was reported by [62] and compared with coordinated checkpointing by [70], where the coordinated checkpointing was found to be more efficient. Both [62] and [70] used small-scale testbeds. To date, the comparative efficiency of coordinated checkpointing and uncoordinated checkpointing with logging has not been fully investigated under scaled conditions in heterogeneous grid systems. Further, because the MPI specification is widely used, it may need to be reviewed to determine if extensions should be provided for fault tolerance of grid applications. These remain possibilities for future work.

Process Migration in Heterogeneous Environments. Another important issue is developing recovery techniques that allow migration of a process to platforms with dissimilar execution environments and different administrative or security domains. Here, examples of preliminary work that precede the emergence of grids are [71] and [72]. For grid systems, the problem was investigated in [73]. More recently in [74], a method was presented for transferring concurrent processes of a parallel application across grid platforms having different processor architectures, number of processors, operating systems, as well as different MPI Implementations. Testbed experiments demonstrated scalability of the approach up to 256 processes. Other research issues relating to checkpointing in grid environments include developing methods for storing checkpointed data on distributed repositories [75, 76], selecting optimal checkpoint intervals [76], and the size of the

checkpointed data sets [74]. With increase in scale and heterogeneity of grid systems, these areas will likely require additional effort as well.

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 at least one replica is always able to continue and 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, while others assume the role of backups. This section reviews resource replication methods developed by researchers for improving fault tolerance in large-scale grid systems. Two aspects of research in resource replication are considered here: (1) developing algorithms for determining optimal (or near-optimal) placement of replicas so as to increase fault tolerance and (2) methods for synchronizing replica states to ensure their consistency. Both remain research issues, for which proposed solutions have been implemented mostly in limited-scale testbeds or as simulations. Under scaled conditions, replica synchronization can incur high overhead costs, and a comprehensive understanding seems lacking of tradeoffs between increasing fault tolerance through replication versus overhead in operational settings. Similarly, with some exceptions [16, 44, 47], there has been little work on understanding when to use checkpoint and process migration instead of resource replication and vice versa.

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 [77]. Weissman and Lee [78] proposed a replica management system which dynamically allocates replicated resources on the basis of user demand, where resource requests by users can be transparently rerouted if individual replicas fail. Testbed experiments demonstrated scalability of this approach. The SRIRAM research system [79] for automatic replication of computing resources in grids and other distributed environments was designed to improve resource availability and fault tolerance. 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 participant 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 and respond to dynamic conditions. Participant 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 and ability to operate under conditions of resource heterogeneity and dynamism. Valcarenghi [80] presented a service replication approach in which replicas are 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 shows the approach can enable recovery of a high percentage of long distance inter-service connections, while minimizing the number of replicas needed and thus simplifying replica management. In [81], a resource allocation system for a computing grid

used in a telecommunications company was reported. In this system, dynamic process replication was used to provide fault tolerance and enable fulfillment of terms of service level agreements. Within the e-Demand project, [82] proposed a replication method that detected faulty computations in grid workflows consisting of multiple tasks. Here, a workflow was simultaneously executed by different sets of service replicas. A voting process was used to select which replica set should return its result to the user. This approach also facilitated identification of faulty services failed in more than one workflow, allowing the service to be eliminated from future consideration. Testbed experiments demonstrated this approach improved workflow fault tolerance. Genaud and Rattanapoka [83] developed a mechanism for MPI-based grid environments that used resource replication to increase fault-tolerance of parallel computations and demonstrated limited scalability in experiments. Other methods for replicating computations on resources have been proposed in [84, 85] that have been verified experimentally under conditions of moderate scale.

Replica Synchronization. Less work appears to have been done of efficient and scalable methods for synchronization of replica states. One method, based on selective replica placement, proposed by [80], is described above. In [86], 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 [87] for synchronizing replicas in distributed environments and demonstrated the efficiency of their approach under both local and wide-area conditions. In earlier work [88], a more traditional primary-backup approach was used to investigate replication of grid services that were implemented using Open Grid Services Infrastructure (OGSI) [89] and the Globus toolkit [90]. In [88], it was found that the strategy could be readily implemented and resulted in higher service availability in local area environments. However, the overhead costs imposed by OGSI notification in order to synchronize states of service replicas that behaved non-deterministically were significant. The study showed that the overhead associated with replica synchronization can be eliminated by allowing failed tasks to be restarted on replicated resources reserved for this purpose. To date, this work has not been repeated with successor specifications to OGSI. Dasgupta et al. [91] proposed a framework for incorporating reservation of redundant backup resources into service-level agreements where failure of the primary allows switching to a backup. Simulation showed circumstances where this approach improved efficiency of system resource allocation. Finding efficient and scalable methods for replica synchronization remains a challenge that must be met before resource replication can be fully utilized as a fault-tolerance tool in grid environments.

3.2.3 Rescheduling Failed Tasks

In addition to process migration and replication, a failed task can be dynamically rescheduled using different resources. This method uses existing grid resource allocation services for rescheduling, thus eliminating the overhead of checkpointing or replica synchronization. However, the rescheduling operation can impose delays. This approach has been investigated in [92, 93], where a prototype rescheduling mechanism was developed and tested in a moderately-scaled production environment containing heterogeneous computing resources. Similarly in [94], a prototype was created that efficiently rescheduled failed jobs in a scaled testbed containing heterogeneous platforms subject to failure. Rescheduling appears to be a

viable fault-tolerance tool to consider under some circumstances. However, because rescheduling can incur delay, it may adversely impact other processes in a workflow that are executing concurrently with, or are dependent on, the process being rescheduled. Perhaps because of this concern, this technique has not been the subject of extensive research effort.

3.2.4 Data Replication

Replication of data sets has been a research topic of long standing for large-scale grid systems and has also been implemented commercially [95, 96]. In [8], use of large-scale replicated data stores to promote fault-tolerance across heterogeneous storage platforms at multiple sites was reported. However, scalable solutions that also work in heterogeneous, dynamic environments do not yet appear to be in place. Within the research community, most experimental scalable data replication methods have been developed to improve performance, with fewer efforts focusing on fault tolerance.

Early research on data grids predicted the benefits from data replication for performance, data availability, and fault tolerance [97-99]. Subsequently, other studies emphasized performance improvements obtained through data replication, including [100-102]. Studies that have focused on fault tolerance include [103], where a quorum-based protocol was described for managing replicated data in large-scale distributed systems, including data grids. Here, experimental results demonstrated fault tolerance by showing that data retrieval can succeed when as many as 75% of replicas have failed. Lei and et al. [104] used reliability metrics to evaluate three data replica placement optimization algorithms that provide improved data reliability in simulated environments. Both [103] and [104] demonstrated a degree of scalability, but did not consider heterogeneous or dynamic environments.

A number of researchers investigated decentralization of services that manage replication of data in order to improve fault tolerance of the service as well as reliability of data operations. In [105], a scalable, fault-tolerant, decentralized replica location service for the Globus toolkit was described, which was designed to avoid a single point of failure. Here, decentralized and redundant replica indexes maintained consistent information about data replicas and their location. These studies reported testbed results that documented performance and scalability, expressed as number of queries processed. Subsequently, this work was extended in [106] to present performance test results compiled using scientific datasets in wide-area environments. This replication service was used operationally in production science grids. In related work, [107] described a decentralized replica location service that was also intended to achieve robustness by avoiding a single point of failure through a redundant, distributed replica management service. This work considered scalability in terms of numbers of replicas and number of queries. Experimental testbed results were provided on performance of this system, but fault-tolerance characteristics were not documented. Zhang et al. [108] proposed an algorithm for dynamically locating data replica servers within a grid in order to optimize performance. This approach was designed to improve fault tolerance of grid data replication services, but scalability issues were not examined and no experimental results were provided. Aside from the example of [106], there appears to be a need to extend more of these results to scalable operational environments, as in other areas of grid fault-tolerance work.

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 be certified as having achieved a level of reliability. Methods for testing and certifying grid components have received less attention among researchers than fault tolerance methods. Nevertheless, the argument for software testing to remove faults as a precondition to fault tolerance is strong. There is ample evidence of the economic cost that arises from having inadequate methods and tools for testing software components prior to operational deployment [109] and in distributed systems for commercial use [110]. In grid systems, testing can also be employed dynamically during execution of user applications.

To date, experimental methods and tools have been developed to discover defects in grid system software. One such method is fault-injection, also used earlier in distributed systems [111-113] as well as for software systems in general [114, 115]. Looker and colleagues [116] reported preliminary work on use of fault-injection to identify malfunctioning distributed software components that could be employed in grid systems. In [117], the work in [116] was extended to generate fault injection test cases using an ontology-based approach. Similarly, Reinecke, van Moorsel, and Wolter [118] used fault-injection to analyze restart oracles in distributed systems environments. Hoarau and Tixeuil [119] studied use of fault-injection to discover grid system components susceptible to process failure faults. Monnet and Bertier [120] also developed a fault injection tool to test the ability of fault detectors to find independent and correlated failures, intended for scaled grid environments.

Beyond fault injection, other testing techniques have been explored. In [121], preliminary work was reported on designing methods for assessing dependability of compositions of distributed software components, focusing on impact of upgrading individual components of commercial off-the-shelf (COTS) software service products. Song and colleagues [122, 123] analyzed GridSphere systems by creating component dependency graphs to identify crucial *hub* components, through which a large portion of system messages flow. Hub components that contain faults are more likely to adversely impact overall system operation, and therefore, can be prioritized for testing. The approach was intended to be generalized for analysis of web-based COTS products and systems. Bitonti and colleagues [124] described a tool for sending test probes to legacy code components that were integrated with the Globus Monitoring and Discovery Service (MDS4). Earlier in [125], a patterned series of query and file-transfer probes were used to test robustness of Globus-based grid systems. Test results were then used to measure robustness and stability of a grid system. A different approach is taken in the *ConCert* project [126], where *certifying* compilers were used to produce machine code that will execute on grid resources. The code contained checkable certificates that could be used to automatically verify code properties when the code is deployed for execution. Though initially used to verify code safety, the approach also appears to be usable to detect faulty or malicious code [127].

These efforts represent a start toward developing testing technology for grids. Perhaps 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 which grid system functions should be prioritized for testing and certification and what kind of tests would be most cost effective (component tests, integration tests,

interaction tests, etc.). Here also, previous research in fault prediction in software components [128-131] can provide a basis for developing testing procedures that identify likely areas to test in large grid systems.

4. RELIABILITY PROVIDED BY GRID APPLICATIONS AND WORKFLOWS

The preceding sections discussed fault-tolerance capabilities that are provided for grid resources from within the grid itself. However in many cases, these capabilities may not be consistently provided or not provided at all. Therefore, from the application's point of view, there may be no guarantees about the reliability of resources available in a grid. For this reason, users are often motivated to design their applications to provide fault tolerance themselves, without the help of fault-tolerance capabilities provided from within grid systems. This is especially true when the application consists of multiple tasks organized into a workflow. In this case, workflow design languages and tools are used to specify the execution order of tasks and data movement between tasks. Once the workflow is defined, grid resources are assigned to workflow tasks using discovery and resource allocation services. A workflow management service may then be used to coordinate execution of the workflow on behalf of the user. As with single-task applications, a workflow may be designed to have built-in fault-tolerance capabilities that do not rely on fault-tolerance capabilities associated with grid resources. An example of this might be a workflow manager that dynamically schedules tasks redundantly on replicated resources for fault tolerance purposes. Here, the goal is for the overall workflow computation to be resilient and continue to execution in face of faults in resources that have been assigned to execute the workflow. Providing fault tolerance from within workflows is especially advantageous if recovery actions require coordination among multiple resources managed by different entities.

Adding fault-tolerance to workflow management has been a research topic prior to appearance of grid systems [132-135], and lack of fault tolerance in available workflow tools was noted early in [136]. This section shows that providing fault tolerance capabilities to workflows in grid systems are also not yet well developed. Today, there is no standard specification for grid workflow design and management that allows specification of fault-tolerance capabilities from within workflows. As a step in this direction, the OGF is specifying requirements for a checkpointing and recovery service that can be initiated by a user application for individual processes [137]. In what follows, current research on fault-tolerance capabilities for grid workflow is described. Then, the important issue is addressed of coordinating fault-tolerance capabilities provided from within workflows with fault-tolerance capabilities originating within the grid. Finally it is important to note that the recent emergence of Web 2.0 network services [138, 139] has provided a set of software components that can be used to construct grid workflows. Because of its newness, the impact of the use of Web 2.0 has not yet been investigated from the standpoint of impact on grid reliability.

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 workflow design and management in web-service-based grid environments. From the standpoint of grid systems, these efforts can be divided into two categories: (1) workflow

languages and tools developed specifically for grid systems with features intended to facilitate fault tolerance in grid environment, and (2) languages and tools developed for more generic distributed service environments based on web service standards developed in the Organization for the Advancement of Structured Information Standards (OASIS) [2]. The second category does not include grid-specific features, but has also been used for grid applications. This section describes work on fault-tolerance capabilities for both classes.

Fault-tolerant capabilities provided by research workflow languages and tools intended for grid systems were described by [140] and [141]. These capabilities facilitate recovery from real-time failures by manipulating the workflow structure to minimize impact of the failure on the workflow computation. The actions to be taken in response to failures may be specified in the workflow definition or initiated by the workflow manager. In [140], these actions included rescheduling failed tasks on slower but more reliable resources, replicating tasks on multiple resources, and user-defined exception handling techniques. Yu and Buyya [141] surveyed 13 research workflow management tools for grid systems and identified tools that support some or all workflow recovery actions described in [140], including [142], [143-147]. A number of tools surveyed in [141] were also found to support transparent recovery of individual workflow tasks, using capabilities provided from within the grid systems described in section 3. Nevertheless, the survey concluded that "most fault handling techniques have not been developed or implemented in many Grid workflow systems", especially techniques initiated from within the workflow. In addition to [141, 148] and also [149] proposed grid workflow managers that were designed to initiate recovery from within workflows, while in [46], an adaptive checkpoint and recovery scheme for grid workflows was studied (see above).

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) [150] standard specification 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, including [151-154]. However, in [151] it was observed that when multiple workflow processes execute concurrently in BPEL4WS, failure of one process requires termination and restart of all. This, of course, would be highly disadvantageous in a grid workflow where extensive process concurrency is common. In response, [151] proposed an XML-based specification language and related mechanisms that permits a concurrent computation to continue. Wasserman and Emmerich [155] conducted a study of failure in scientific grid workflows defined using an early version of BPEL4WS and also concluded that current approaches for ensuring reliability in workflows were inadequate. Other standard web service specifications for coordination of distributed computations also addressed fault-tolerance in a general, web service context. These include WS-Coordination [156], the Business Transaction Protocol [157].

4.2 Coordinating Workflow and Grid Resource Fault-tolerance Strategies

The preceding discussions have presented two different sources for providing fault tolerance in grid systems: fault-tolerance capabilities provided from within the grid system and capabilities provided by a grid user or the user's agent. 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 be fault-tolerant if the grid resources allocated to their tasks already provide adequate fault-tolerant 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 service level agreement [158]. To enable this coordination will require standardized conventions for describing and negotiating fault-tolerance capabilities, a potential topic for future research.

5. SUPPORTING GRID INFRASTRUCTURE AND RESOURCE MANAGEMENT.

Infrastructure and management services are essential services that manage operation of the grid. This includes services that allow users to discover grid resources that meet the requirements of their applications; e.g., service discovery through directories or other facilities. It also includes services for scheduling use of grid resources, job submission and monitoring, providing notifications of status of grid resources or executing tasks, grid usage and accounting services, and security (authentication, authorization, encryption, etc.). As with grid resources, the reliability of infrastructure and management services can be improved by applying fault tolerance methods described in section 3, including fault detection (section 3.1), recovery methods (section 3.2), and testing methods (section 3.3). However, in contrast to grid resources, infrastructure and management services have a wider scope and more central function. For the grid to operate, ensuring reliability of these services is critical, and so, infrastructure and management services can be viewed as a separate area for fault tolerance research. While there have been efforts directed specifically towards making these essential services fault tolerant, there has perhaps been less work in this area than is needed. For example, few of the grid resource management systems surveyed in [159] were reported to have built-in fault-tolerant capabilities. This section reviews the available research directed toward making infrastructure and management services more fault tolerant.

Work on fault-tolerant data replica management services [103, 105-107] has been described above. The OGF *Grid Monitoring Architecture* document [35] describes generic requirements for monitoring grid systems. The document states that monitoring subsystems should be fault-tolerant, secure, scalable, and interoperable across heterogeneous grid resources. A few experimental systems have been developed that were intended to realize this goal. Use of redundancy to achieve fault tolerance in grid monitoring systems was reported in [160] and in [161] which achieved fault tolerance by replicating monitoring data at multiple nodes. In [162], resources being monitored registered information about themselves in registries that could be accessed by interested parties to learn status of resources. Here, registries were replicated for fault-tolerance purposes. In [163], a fault-tolerant service discovery system was described that is built on top of the Jini Service Discovery protocol [164]. This approach exploited the inherent redundancy of Jini lookup services to build a distributed, hierarchical index of grid resources in which index nodes are 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 results from a testbed were

provided to document system performance; however, tests of fault-tolerance capabilities were not reported.

Another important function in grid systems is co-allocation or co-scheduling of tasks that must run concurrently on different resources. In [165] a co-allocation scheduling service was described that is based on the Paxos three-phase transaction commit protocol [166] which achieves fault-tolerance through use of distributed, redundant transaction coordinators. Earlier work on increasing survivability of secure communications services in distributed environments through use of redundancy was reported in [167]. This work described a variety of redundancy techniques for making security services resistant to attack, which can be used in grid environments.

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 reliability metrics to measure the reliability of a large number of grid resources and, thus indirectly, the grid itself. For instance, in [168], 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 [169]. In [85], failure history of computing resources was used to create quantitative reliability ratings that could be used in resource allocation and scheduling decisions (see above). In [104], data replica management was enhanced by use of data availability measures to evaluate data replica optimization algorithms. Data availability was expressed as ratio of files unavailable to files requested and the ratio of bytes unavailable to bytes requested. In [170] a framework for evaluating quality-of-service (QoS) provided by a grid system was described that introduced metrics for service accessibility and availability.

6. GRID CONNECTION AND TRANSPORT RELIABILITY

Grid networks enable message exchange and data transport in grid systems. 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 1Gb/s per flow). Because grid applications often require large-scale data transport capabilities for extended durations, connections between 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 for parallel processing. 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 will require use of many network components. This in turn increases the probability of failures that necessitate rerouting connections using functioning components. For these reasons, ensuring reliable transport in large-scale grid networks is an important and difficult research problem. This section surveys work on this problem. The section first considers standards for reliable connectivity and data transport that are coming into use in grid environments and discusses work intended to strengthen their fault-tolerance features. Then, the section examines methods being investigated for establishing grid networks that ensure reliable connectivity and data transport. Two methods are of special interest: overlay, or virtual, networks and dedicated networks, both of which have been used in operational grid systems. Finally, work on the important topic of reliable multicast transmission in grid systems is addressed.

6.1 Specifications for Reliable Connection and Transport

To date, there are 3 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. The Transmission Control Protocol (TCP), which provides general-purpose, fault-tolerant, point-to-point connectivity in network systems, is also employed for connection establishment in grid environments. In this role, TCP is used in combination with other transport protocols. In [171], an OGF survey of available TCP alternatives found none by itself fully met the requirements of grid networks.

The *Web Services Reliable Messaging Protocol* (*WS-ReliableMessaging*) [172] was originally 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 that use this protocol. 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* delivery of a message. The *WS-Reliability* specification [173] provides similar capabilities. Both specifications prescribe a binding that allows its messages to be encoded and transmitted using the XML-based Simple Object Access Protocol (SOAP) protocol [174]. Both specifications are extensible using the Web Service Description Language (WSDL) [175], to allow combination with other web service specifications to define new services.

In [176], a comparison of the two reliable messaging specifications concluded that *WS-ReliableMessaging* provides more flexible features for re-initiating erroneous transmissions and also provides more extensive capabilities for reporting faults that occur during transmission. Another comparison reported in [177] found the two specifications were designed to respond to different types of faults, which produced differences in error handling in implementations. Initial efforts to implement *WS-ReliableMessaging* [176, 178] and *WS-Reliability* have not yet yielded information on the effectiveness of using these specifications in production grid environments. A comprehensive analysis based on actual operational experience would be needed to evaluate reliability aspects of these specifications for web-based grid applications and to bring to light specific issues that need attention. Subsequent standardization of *WS-ReliableMessaging* [172] by OASIS [2] indicates growing use of this specification.

GridFTP, version 2 [179] is an important specification developed by the Globus Alliance and OGF. GridFTP extends the File Transfer Protocol (FTP) [180] 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 the underlying TCP and employs a checksum technique to detect data loss that may have occurred during transfer. In [181], GridFTP was compared with other bulk data transfer protocols and found to be highly reliable and scalable in relation to other protocols. However, a known problem in GridFTP is that failure of a client necessitates restart of data transmission, a disadvantage for transferring large data sets. This is overcome by fault-tolerance mechanisms provided in [182] and the Globus toolkit [90], 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 [183] and by [184] 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 [185, 186], a messaging infrastructure was presented for support of 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), SOAP messaging, and web service specifications for addressing [187] and event notification [188]. 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 supported guaranteed delivery of messages and fault tolerance in the face of broker failure, failure or disconnect of communicating services, and link failure. Flexible reconfiguration of the broker topology through operations specified in WS-Management [189] enhanced scalability of the network. The viability of this approach was shown in the implementation of the Grid-FTP recovery mechanism referred to above [182] and prototype grid applications involving streaming audio and video data. The Globus toolkit [90] also includes a service for transparent, fault-tolerant transfer of data by GridFTP through an intermediate distributed DBMS that performs a function analogous to the Narada broker network. Still another approach to this problem was taken in [190], who proposed and demonstrated a service for bulk data transfer between heterogeneous storage systems that use dissimilar data access protocols, relying on redundant data transfers across intermediate hops to improve fault-tolerance. In [191], 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. An important aspect of routing is traffic engineering [192], the selection of paths through a network to maximize data flow within available bandwidth without violating administrative constraints. Effective routing mechanisms are important to dynamically reroute grid data flows around failed network components. Identification of routing mechanisms that will perform well in grid environments is an important topic of research, since current interior gateway protocols, such as the combined Open Shortest Path First, Intermediate System to Intermediate System (OSPF/IS-IS) [193, 194], and exterior gateway protocols, such as the Border Gateway Protocol (BGP) [195], have been regarded as insufficient as insufficient [14, 80]. One solution involves creating virtual overlay networks on top of existing physical networks using Multi-Protocol Label Switching (MPLS) [196, 197] standard to provide better bandwidth availability and predictable performance. Clapp et al [198] 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 the underlying network resources. This issue is important because the reliability of the overlay depends on the

reliability of the underlying resources. Overlays may wish to be notified of the resource failure in order to take appropriate recovery actions or request additional allocation of physical resources for rerouting purposes. Thus, from a reliability standpoint, interactions between overlays and the supporting network layers is an area of needed research.

Given the research on promoting fault tolerance at different logical network layers represented by such systems as NaradaBrokering [185] (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 higher levels of availability in a grid network? Similarly, would fault tolerance be enhanced by mapping long-distance connections between the service islands described in [80] 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 for grid environments. It is important to know how management of overlay networks might differ for grid applications, particularly with respect to key functions relating to fault tolerance. Finally, it is necessary to investigate allocation of dedicated network resources for use in grid networks, rather than sharing resources as occurs in overlays. Given the heavy demands for data transport in grid systems, ultimately it may be preferable to fulfill grid networking requirements by employing dedicated network resources, as some have done. An example of this is the interlinking of the TeraGrid research grid [199] using optical network backplane [200]. However, to date, dedicated networks have been used for grids that involve a limited and stable number of participants. An important question to answer is whether this solution would scale in very large grids where membership is dynamic.

6.3 Reliable Multicasting of Large Datasets

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. Perhaps the best-known current example of the use of multicasting in grid applications is the Access Grid [201], where large audio and video datasets are broadcast regularly to many participants. However, in these and other uses of multicasting in grid networks [186], best effort multicast is used, which provides high-throughput and low end-to-end delay, but does not provide guaranteed delivery [202].

Reliable multicast protocols have been the subject of research for years, both within grid settings and for more general purpose use [203, 204]. The Nack-oriented reliable multicast (NORM) is currently being developed as an IETF standard [205]. In [202] a series of trials were conducted in a scaled grid network testbed to compare performance of NORM, the Multicast Dissemination Protocol (MDP) [206], and a variant of TCP extended for multicasting. The results revealed significant design problems, bottlenecks and limitations in NORM and MDP that hampered throughput. These protocols were also determined to be less robust than TCP in real-world production networks.

Other researchers have proposed different multicast solutions, as for instance [207] which provides good performance for moderately-sized multicast groups in a simulated wide area network, but does not provide guaranteed reliable delivery. In [208], experiments were

reported on using the TRAM (*Tree-based Reliable Multicast)* protocol [209] to multicast across a small-scale, heterogeneous compute grid for task farming. These efforts also provided evidence that, at present, no multicast protocol has been identified that provides delivery of large data sets that is reliable, scalable, and meets the performance requirements of grid systems. Despite this, 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 [211].

7. RELIABILITY CONCERNS FROM AN OVERALL SYSTEM PERSPECTIVE

This section discusses approaches to grid system reliability that are based on an overall system-wide view, rather than focusing on individual functional areas or types of components. The purpose in adopting an overall perspective is to gain insights 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 high-level design or structure of the grid to improve the overall reliability, for instance, by identifying architectural alternatives that foster fault tolerance. The second involves using quantitative methods 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[2] or sites and their interconnections. Nodes 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 in which sites are organized in a logical tree contrasts with a decentralized architecture having no central point of control, which as a result, may also have fewer points of failure. Architectures can also be distinguished by choice of geographic location for sites or by the distribution of resources across sites; e.g., a few sites with many resources versus many sites having few resources. Distribution of resources across sites could impact resource availability in the event 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 again could influence if single points of failure or if 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. Both the distribution of these services and the design of the software system that implements them may affect their ability to reliably

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

carry out management functions. To date, only a few researchers have used architectural concepts to analyze grid reliability, as for instance [122, 123], who developed a method for identifying centralized "hub" software components in which faults are most likely to impact overall system reliability. In [212], a grid system architecture based on the Open Grid Service Architecture (OGSA) [7] was proposed for creating dependable grid systems. Preliminary inquiries into the influence of architecture on grid reliability were also made in [185, 213].

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, 214] for overviews). Quantitative assessments of overall reliability are based on, and constrained by, architectural considerations, since computing reliability of arbitrarily structured distributed systems has been shown to be intractable [215]. More recently, there has been work on this problem for grid systems. A method for measuring the overall reliability of a grid system based on known component failure rates was presented by Xie, Dai, and Poh in [214]. The method considers a system model consisting of a set of nodes, links, grid resources, and a resource management system. A set of workflow tasks is then allocated to grid resources on particular nodes. Given that each of these model components have known failure rates, the reliability of a grid system can be estimated by the probability that a set of user applications, executing on the grid as workflows will complete. In subsequent work, Dai, Pan, and Zou [216] revised this approach into a layered hierarchal grid model, where reliability analysis considers probability of different kinds of failures at each layer. The analysis incorporates Markov chain modeling of resource request queues in which blocking and time-out failures occur. In related work, [217] presented a model for measuring reliability of a set of grid components that share common communications links and are controlled by a single resource manager. Using this model as a basis, an algorithm for optimizing resource allocation was presented in [218]. In [219], an algorithm for computing reliability of grid systems having a star architecture was described. Developing quantitative metrics for assessing overall system reliability will be important to adoption and use of grid technology because they provide tools for evaluating operational grid systems in mission-critical settings. They therefore are an important area of research.

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 are not necessarily 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. Understanding causes of emergent behavior provides a basis for developing decentralized methods of control, which when implemented by components across a grid, lead to desirable global fault-free states. However, developing tractable methods to understand causes of emergent global 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 [220]. Other work uses simulation studies to demonstrate the importance of viewing a grid

as a complex system. In [221], 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 [222], 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 [223] demonstrated in more detail the feasibility of using grid standard specifications to produce viable resource allocation in a grid compute economy. These examples show that using complex systems methods to understand global behavior of grid systems can be used to improve grid reliability. As with architectural analysis and quantitative measurement, the importance of work in this area is likely to increase as 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. CONCLUSIONS

Developing reliability methods for large-scale, heterogeneous, dynamic grid systems remains a challenge that must be met if the vision of future grid systems is to be fully realized. Many currently deployed grid systems utilize fault-tolerance methods that have been adapted from contemporary high-performance, distributed computing. At present levels of scale, these methods can provide reliability to grid systems used by individual organizations or groups of cooperating organizations, as evidenced by a number of currently deployed grids. However, current methods must be further evolved and new methods must be developed for future, larger-scale, highly dynamic environments where managed resources are no longer contained within limited, known administrative domains.

To date, reliability work in grid systems has centered on fault tolerance. This study has shown that while significant work has occurred in developing fault-tolerance methods that are scalable and work in heterogeneous, dynamic environments, this work still remains in the research stage. Fault-tolerance work has 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 across these areas, and in each area, different problems still remain to be solved. Progress has been made in developing efficient fault detection methods for large-scale grids. Similarly, there has been work on developing methods for checkpointing and process migration that are used in research grid systems, while experimental methods for resource replication appear to have been deployed less often. Significant efforts have also been devoted to improving fault tolerance in grid data transport through the use of overlay networks and dedicated networks. There has also been notable progress in ensuring reliable connectivity and bulk transport of large data sets.

A number of key specifications need to be further examined to determine if they should be extended to support reliability. In some cases, needed specifications are incomplete, such as reliable multicasting and application checkpoint and recovery. Comprehensive guidelines are also needed for implementing fault-tolerant methods in grid environments. Much less work has occurred on test methods for fault prevention and workflow design and management tools with fault-tolerance features. The recent emergence of Web 2.0 network services [138, 139] and its potential use in grid systems has not yet been investigated from the standpoint of

grid reliability. More research needs to be devoted to the important topic of developing methods to measure overall grid system reliability. Similarly, it will be important to consider the roles of architecture and complex systems methods in improving grid systems reliability, subjects that will assume increasing importance as grid systems scale. Given the criticality of reliability to the continued advancement of grid technology, it is important that work in this area continues and is expanded, and that promising experimental methods be evolved for use in production environments.

ACKNOWLEDGEMENTS

I wish to thank Matti Hiltunen of AT&T his many insightful comments that helped improve this manuscript.

REFERENCES

1. Open Grid Forum http://www.ogf.org [January 22, 2008].
2. Organization for the Advancement of Structured Information Standards (OASIS) http://www.oasis-open.org [January 23 2008].
3. Internet Engineering Task Force http://www.ietf.org [January 22, 2008].
4. Avizienis A. et al. 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, et al. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. http://www.globus.org/research/papers/ogsa.pdf, 2002. [December 20, 2007].
7. Foster I. et al. (ed.). *The Open Grid Services Architecture, Version 1.5*, GFD.80, Open Grid Forum, July 2006.
8. Raffo D. eBay Grid, redundancy, and home-cooked management help site survive. *Byte and Switch*, November 22, 2006.
9. Carr D. How Google Works. *Baseline Magazine*, July 6, 2006, http://www.baselinemag.com.
10. Mogilevsky D, Koenig G, Yurcik W. Byzantine Anomaly Testing for Charm++: Providing Fault Tolerance a nd Survivability for Charm++ Empowered Clusters. *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops (CCGRIDW'06)*, 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, *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 (GCC 2006)*, 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 11th European Conference on Parallel Processing (Euro-Par 2005)*, (*Lecture Notes in Computer Science*, vol. 3648), Cunha J, Medeiros P. (eds.). Springer-Verlag: 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, in *Wiley Encyclopedia of Computer Science and Engineering*, Volume 14, Webster J. (ed.). Wiley: New York, 1999: 213-218.
19. Medard M, Lumetta S. Network Reliability and Fault Tolerance, in *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) Final Release, December 2000. http://www.ieeetfcc.org/
21. Milojicic D. et al. "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. et al. A Survey and Comparison of Peer-To-Peer Overlay Network Schemes. *IEEE Communications Surveys*, 2005; **7** (2): 72-93.

27. Bianchini R, Buskens R. "An Adaptive Distributed System-Level Diagnosis Algorithm and Its Implementation," *Proceedings of the Twenty-First International Fault-Tolerant Computing Symposium (FTCS-21)*, June 1991. IEEE Computer Society Press: Los Alamitos, CA, 1991; 222-229.
28. 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*, October 1994. (Lecture Notes In Computer Science, vol. 852), Echtle K, Hammer D, Powell D. (eds.). Springer-Verlag, London, 1994; 599-616.
29. Gupta I, Chandra T, Goldszmidt G. On scalable and efficient distributed failure detectors. *Proceedings of 20th Annual ACM Symposium on Principles of Distributed Computing*, August 2001. ACM Press, New York, 2001; 170–179.
30. Presuhn R. et al. *Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)*. RFC 3416, Internet Engineering Task Force, December 2002.
31. 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.
32. Chandra T, Toueg S. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the Association for Computing Machinery*, 1996; **43** (2): 225-267.
33. Hayashibara N, Cherif A, Katayama T. Failure Detectors for Large-Scale Distributed Systems. *Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems (SRDS'02)*, October 2002. IEEE Computer Society Press: Los Alamitos, CA, 2002; 404-409.
34. Zanikolas S, Sakellariou R. A Taxonomy of Grid Monitoring Systems. *Future Generation Computer Systems*, 2005; **21** (1):163–188.
35. Tierney B. et al. *A Grid Monitoring Architecture*, Informational Document GFD-I.7, Open Grid Forum, January 2002.
36. Stelling P. et al. A Fault Detection Service for Wide Area Distributed Computations. Cluster Computing, 1999; 2 (2): 117-128.
37. Jain A, Shyamasundar R. Failure Detection and Membership Management in Grid Environments. *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, November 2004. IEEE Computer Society Press: Los Alamitos, CA, 2005; 44-52.
38. Horita Y, Taura K, Chikayama T. A Scalable and Efficient Self-Organizing Failure Detector for Grid Applications. *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, November 2005. IEEE Computer Society Press: Los Alamitos, CA, 2006; 202-210.
39. 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 (DSN'02), June 2002. IEEE Computer Society Press: Los Alamitos, CA, 2002; 303–312.
40. Abawajy J. Fault Detection Service Architecture for Grid Computing Systems, *International Conference on Computational Science and Its Applications (ICCSA)*, *Part II* (*Lecture Notes in Computer Science*, vol. 3044), Lagan A et al. (eds.). Springer: Berlin, 2004; 107-115.
41. Chan K. et al. A fault taxonomy for web service compositions. University of Pretoria, Department of Computer Science, Polelo Research Group, December 2006.
42. Bruning S, Weissleder S, Malek M. A Fault Taxonomy for Service-Oriented Architecture *Proceedings of the 10th High Assurance Systems Engineering Symposium (HASE '07)*, November 2007. IEEE Computer Society Press: Los Alamitos, CA, 2007; 367-368.
43. Hofer J, Fahringer T. A Multi-Perspective Taxonomy for Systematic Classification of Grid Faults. To appear in *Proceedings of 16th Euromicro International Conference on Parallel, Distributed and network-based Processing (PDP'08)*, February 2008. IEEE Computer Society Press: Los Alamitos, CA.
44. 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 (IPDPS 2007)*, March 2007. IEEE Computer Society Press: Los Alamitos, CA, 2007; 1-8.
45. 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.[
46. 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 (GCCW'06)*, October 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 181-188. DOI 10.1109/GCCW.2006.69.
47. Jin H. et al. DRIC: Dependable Grid Computing Framework," *IEICE Transactions on Information and Systems*, 2006; **E89-D** (2): 612-623.
48. Röblitz T. et al. Autonomic Management of Large Clusters and Their Integration into the Grid. *Journal of Grid Computing*, 2005; **2** (3): 247–260.
49. Tanimura Y. et al. Implementation of Fault-Tolerant GridRPC Applications. *Journal of Grid Computing*, 2006; 4 (2): 145–157.
50. [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 (AINA'06)*, April 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 69-74.
51. DataSynapse GridServer. http://www.datasynapse.com/en/products/gridserver.php [February 28, 2008].
52. Hewlett-Packard Grid Computing, 2007 http://h20331.www2.hp.com/enterprise/cache/125369-0-0-225-121.html [December 10, 2007]
53. Netcool + Tivoli: delivering service management innovation, IBM, 2007 http://www-306.ibm.com/software/tivoli/products/netcool-omnibus/ [December 10, 2007].
54. IBM alphaWorks Grid Computing, http://www.alphaworks.ibm.com/grid [December 10, 2007].
55. *Administering Platform Process Manager*, Version 3.0, Platform Computing Corporation, March 2005.
56. *N1 Grid Engine User's Guide*, Sun MicroSystems, Inc., May 2005.
57. 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-Verlag: London, 2001; 273-283.
58. Goodale T. et al. The Cactus Framework and Toolkit: Design and Applications. *Fifth International Conference on Vector and Parallel Processing (VECPAR '2002)* (*Lecture Notes in Computer Science*, vol. 2565). Springer: Berlin, 2003; 15-36.

59. Lanfermann G. et al. Nomadic Migration: Fault Tolerance in a Disruptive Grid Environment. *Proceedings of the Second IEEE/ACM International Symposium Cluster Computing and the Grid*, May 2002. IEEE Computer Society Press: Los Alamitos, CA, 2002; 280-282.
60. Limaye, K. et al. Job-Site Level Fault Tolerance for Cluster and Grid environments. *Proceedings of IEEE International Conference on Cluster Computing*, September, 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 1-9.
61. Adding high availability to Condor Central manager. http://dsl.cs.technion.ac.il/projects/gozal/project_pp./ha/ha.html [December 20, 2007]
62. Bouteiller A. et al. MPICH-V: a Multiprotocol Automatic Fault Tolerant MPI. *International Journal of High Performance Computing and Applications*; **20** (3); 1999: 319-330.
63. *MPI-2, Extensions to the Message-Passing Interface*, Message Passing Interface Forum, November 2003, http://www.mpi-forum.org/.
64. Buntinas D. et al. Blocking vs. Non-Blocking Coordinated Checkpointing for Large-Scale Fault Tolerant MPI. Accepted for publication in *Future Generation Computer Systems*, Elsevier Press: Amsterdam, 2007.
65. Sankaran S. et al. The LAM/MPI Checkpoint/Restart Framework: System-Initiated Checkpointing. Proceedings of the LACSI Symposium, October 2003. Los Alamos Computer Science Institute: Santa Fe, New Mexico, 2003.
66. Yeom H. Providing Fault-tolerance for Parallel Programs on Grid (FT-MPICH). presented at the *OGF First Workshop of Reliability and Robustness in Grid Computing Systems*, Athens, Greece, February 2006. http://www.ogf.org.
67. Woo N, Yeom H, Park T. MPICH-GF: Transparent checkpointing and rollback-recovery for grid-enabled MPI processes. *IEICE Transactions on Information and Systems*, 2004; **E87-D** (7):1820-1828.
68. Kim H, Yeom H. A user-transparent recoverable file system for distributed computing environment. *Proceedings of Challenges of Large Applications in Distributed Environments, (CLADE 2005)*, July 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 45- 53.
69. Strom E, Yemini S. Optimistic recovery in distributed systems. *Transactions on Computer Systems*, 1985; **3** (3): 204–226.
70. [Woo N. et al. Performance Evaluation of Consistent Recovery Protocols Using MPICH-GF. *Proceedings of the 5th European Dependable Computing Conference*, April 2005 (*Lecture Notes in Computer Science, vol. 3463*), Dal Cin M, Kaaniche M, Pataricza A. (eds.). Springer-Verlag: Berlin, 2005; 167-178. DOI 10.1007/b107276.
71. Ramkumar B, Strumpen V. Portable Checkpointing for Heterogeneous Architectures. *Proceedings of the Twenty-seventh International Symposium on Fault-Tolerant Computing - Digest of Papers*, June 1997. IEEE Computer Society Press: Los Alamitos, CA, 1997; 58-67.
72. 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.
73. Vadhiyar S, Dongarra J. SRS - A Framework for Developing Malleable and Migratable Parallel Software. *Parallel Processing Letters*, 2003; **13** (2): 291-312.
74. 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 (PPoPP '06)*, March 2006. ACM Press, New York, 2006; 22-31.
75. 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 (MGC '05)*, November 2005. ACM Press, New York, 2006; 1-8.
76. 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.
77. Andrzejak A, et al. *Algorithms for Self-Organization and Adaptive Service Placement in Dynamic Distributed Systems*. Hewlet Packard Corporation, 2002. HPL-2002-259.
78. 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.
79. Verma, D. et al. SRIRAM: A scalable resilient autonomic mesh. *IBM Systems Journal*, 2003; **42** (1); 19-28.
80. Valcarenghi L, Piero C. QoS-Aware Connection Resilience for Network-Aware Grid Computing Fault Tolerance. *Proceedings of 2005 7th International Conference on Transparent Optical Networks-Volume 1*, July 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 417-422.
81. Lac C, Ramanathan S. A Resilient Telco Grid Middleware. *Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06)*, June 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 306-311.
82. Townend P, et al. FT-Grid: A Fault-Tolerance System for e-Science, *Proceedings of the 4th UK e-Science Fourth All Hands Meeting (AHM05)*, September 2005. Engineering and Physical Sciences Research Council: Swindon, United Kingdom, 2005.
83. 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.
84. 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.
85. 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.
86. Zhang X. et al. 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.
87. Lamport L. Paxos made simple. *ACM SIGACT News (Distributed Computing Column)*, 2001; **32** (4): 18-25.
88. Zhang X. et al. Fault–tolerant Grid Services Using Primary–Backup: Feasibility and Performance. *Proceedings of the 2004 IEEE International Conference on Cluster Computing (Cluster 2004)*, September 2004. IEEE Computer Society Press: Los Alamitos, CA, 2005; 105-114. DOI 10.1109/CLUSTR.2004.1392608.
89. Tuecke S. et al. (eds.). *Open Grid Services Infrastructure (OGSI), Version 1.0*, Proposed Recommendation, Open Grid Forum, July 2003.
90. *The Globus Toolkit*. http://www.globus.org/toolkit/ [December 20, 2007].

91. Dasgupta G. et al. QoS-GRAF: A Framework for QoS based Grid Resource Allocation with Failure provisioning. Fourteenth IEEE International Workshop on Quality of Service (IWQoS 2006), June 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 281-283.
92. Huedo E, Montero R, Llorente I. A framework for adaptive execution in grids. *Software—Practice and Experience*, 2004; **34** (7): 631-651.
93. 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.
94. In J. et al. SPHINX: A Fault-Tolerant System for Scheduling in Dynamic Grid Environments. *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05),* April 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 12b-12b.
95. IBM alphaWorks, Grid File Replication Manager, IBM, 2004. http://www.alphaworks.ibm.com/tech/gfrm [December 10, 2007].
96. *Data Grids and Service-Oriented Architecture, an Oracle White Paper*, Oracle Corporation http://www.oracle.com/technologies/grid/index.html, [December 10, 2007].
97. Chervenak A. et al. 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.
98. Hoschek W, et al. 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.
99. Stockinger H. et al. File and object replication in data grids. *Proceedings of Tenth IEEE Symposium on High Performance and Distributed Computing*, 2001. IEEE Computer Society Press: Los Alamitos, CA, 2001; 76-86
100. Bell W. et al. 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 (CCGrid 2003)*, May 2003. IEEE Computer Society Press: Los Alamitos, CA, 2001; 661-668.
101. Takizawa S. et al. A Scalable Multi-Replication Framework for Data Grid. *Proceedings of the 2005 Symposium on Applications and the Internet Workshops (SAINT-W'05)*, January 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 310-315.
102. 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 (CCGRID'06)*, May 2006. IEEE Computer Society Press: Washington, DC, 2006; 417-420.
103. Deris M, Abawajy J, Suzuri H. An efficient replicated data access approach for large-scale distributed systems. Proceedings of the 2004 *IEEE International Symposium on Cluster Computing and the Grid*, April 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004; 588-594.
104. 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 Symposium*, March 2007. IEEE Computer Society Press: Los Alamitos, CA, 2007; 1-8.
105. Chervenak A. et al. Performance and Scalability of a Replica Location Service. *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC* Chervenak A. et al. *-13)*, June 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004; 182-191.
106. Chervenak A. et al. "Wide area data replication for scientific collaborations. *Proceedings of the 6th International Workshop on Grid Computing*, November 2005. IEEE Computer Society Press: Los Alamitos, CA; 2005.
107. 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.
108. Zhang Q. et al. Dynamic Replica Location Service Supporting Data Grid Systems. *Proceedings of the Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*, September 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 61.
109. 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.
110. Research Triangle Institute, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, May 2002.
111. Barton J, Czeck E, Segall Z, Siewiorek D. Fault injection experiments using FIAT. *IEEE Transactions on Computers*;, 1990; **39** (4): 575-582.
112. Carreira J, Costa D, Silva J. Fault Injection Spot-Checks Computer System Dependability. *IEEE Spectrum*; **36**(8); 1999: 50-55.
113. 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; 404-14.
114. 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.
115. Voas,J. Miller K. Using Fault Injection to Assess Software Engineering Standards. *Proceedings of the Second IEEE Engineering Standards Symposium*, August 1995. IEEE Computer Society Press: Los Alamitos, CA, 1995; 139-145.
116. Looker N. et al. 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.
117. 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.
118. 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 (QEST 2006)*, September 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 375-384.
119. 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, November 2005; 194-201.

120. 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 - VECPAR 2006* (Lecture Notes in Computer Science, vol. 4395), Dayde M. et al. (eds.), Springer-Verlag: Berlin, 2007; 610-621, 2007.
121. 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 (DSN 2004)*, (*Lecture Notes in Computer Science*, vol. 3549), de Lemos R, Gack C, Romanovsky A. (eds.). Springer-Verlag: Berlin, 2005; 92-121.
122. Song, C. et al. Assessing Reliability of Grid Software Systems Using Emergent Features. *The 2nd Workshop on Reliability and Robustness in Grid Computing Systems, the 19th Open Grid Forum (OGF19)*, January, 2007. http://www.ogf.org.
123. Topkara U, Song C, Woo J, Park S, "Connected in a Small World: Rapid Integration of Biological Resources", *Grid Computing Environments Workshop*, November 2006. http://umut.topkara.org/papers/research_work.html [January 22, 2008].
124. Bitonti L. et al. Dynamic testing of legacy code resources on the grid. *Proceedings of the 3rd conference on Computing frontiers (CF '06)*, May 2006. ACM Press, New York, 2006; 261-268.
125. Chun G. et al. Benchmark Probes for Grid Assessment. *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04)*, April 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004; 276-283.
126. Chang B. et al. Trustless Grid Computing in ConCert. *Proceedings of the Third International Workshop on Grid Computing (GRID 2002)*, November 2002. IEEE Computer Society Press: Los Alamitos, CA, 2002; 112-125.
127. 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.
128. Munson J, Khoshgoftaar T. The detection of fault-prone programs. *Transactions on Software Engineering*, 1992; **18** (5): 423-433.
129. Graves T, Karr A, Marron J, Siy H. Predicting fault incidence using software change history. *Transactions on Software Engineering*, 2000; **26** (7): 653-661.
130. 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
131. Weyuker E, Ostrand T, Bell R. Adapting a Fault Prediction Model to Allow Widespread Usage. *Proceedings of the Second IEEE International Promise Workshop*, September, 2006. http://promisedata.org/repository/papers.html [January 22, 2008].
132. 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 (RIDE '97), High Performance Database Management for Large-Scale Applications*, April 1997. IEEE Computer Society Press: Los Alamitos, CA, 1997; 76-79.
133. Borgida A, Murata T. Tolerating exceptions in workflows: a unified framework for data and processes. *Proceedings of International Joint Conference on Work Activities Coordination and Collaboration (WACC'99)*, February 1999. ACM Press, New York, 1999; 59-68.
134. Hagan C, Alonso G. Exception Handling in Workflow Management Systems. *IEEE Transactions on Software Engineering*, 2000; **26**(10): 943-958.
135. Cardoso J, et al. Survivability Architecture for Workflow Management Systems, *Proceedings of the 39th Annual ACM Southeast Conference*, March 2001. ACM Press, New York, 1999; 207-216.
136. Alonso G, et al. Enhancing the Fault Tolerance of Workflow Management Systems. *IEEE Concurrency*, 2000; **8** (3): 74-81.
137. 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, 2005.
138. Fox, G, et al. Web 2.0 for Grids and e-Science. *Proceedings of the Laboratories, April 20072nd International Workshop on Distributed Cooperative (INGRID 2007 - Instrumenting the Grid)*, April 2007. Consorzio Nazionale Interuniversitario per le Telecomunicazioni{CNIT), Firenze, Italy.
139. Topcu A, et al. Integration of Collaborative Information Systems in Web 2.0. *Proceedings of the 3rd International Conference on Semantics, Knowledge and Grid (SKG2007)*, October 2007. IEEE Computer Society Press: Los Alamitos, CA.
140. Hwang S, Kesselman C, GridWorkflow : A Flexible Failure Handling Framework for the Grid. Proceedings of the 12th IEEE Internatioal Symposium on High Performance Distributed Computing (HPDC-12 '03), June 2003. IEEE Computer Society Press: Los Alamitos, CA, 2003; 126-137.
141. Yu J, Buyya R. *A Taxonomy of Workflow Management Systems for Grid Computing*University of Melbourne, Australia. Technical Report GRIDS-TR-2005-1. March 10 2005.
142. Tannenbaum T. et al. Condor - A Distributed Job Scheduler. In *Beowulf Cluster Computing with Linux*, Sterling T (ed.). The MIT Press: Cambridge, MA, 2001; 307-350.
143. Deelman E, et al. Mapping Abstract Complex Workflows onto Grid Environments. *Journal of Grid Computing*, 2003; **1** (1): 25-39.
144. Altintas I. et al. A Framework for the Design and Reuse of Grid Workflows. *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.
145. von Laszewski G, Hategan M. *Java CoG Kit Karajan/GridAnt Workflow Guide*. Technical Report, Argonne National Laboratory, Argonne, IL, USA, 2005.
146. Fahringer T. et al. ASKALON: a tool set for cluster and Grid computing. *Concurrency and Computation: Practice and Experience*, 2005; **17** (2-4): 143-169.
147. Oinn T. et al. Taverna: a Tool for the Composition and Enactment of Bioinformatics Workflows. *Bioinformatics*, 2004; **20** (17): 3045-3054.
148. Ra D. et al. Scalable Enterprise Level Workflow Manager for the Grid. *Proceedings of the Fifth International Conference on Quality Software (QSIC'05)*, September 2005. IEEE Computer Society Press: Los Alamitos, CA, 2006; 341-348.
149. Yu J, Buyya R. A Novel Architecture for Realizing Grid Workflow using Tuple Spaces. *The 5th IEEE/ACM International Workshop on Grid Computing (Grid 2004)*, November 2004. IEEE Computer Society Press: Los Alamitos, CA, 2005; 119-128. DOI 10.1109/GRID.2004.3.
150. Alves A. et al. (ed.) *Web Services Business Process Execution Language Version 2,* Organization for the Advancement of Structured Information Standards (OASIS), April 2007.

151. Tartanoglu F. et al. Coordinated Forward Error Recovery for Composite Web Services. *Proceedings of the 22nd International Symposium on Reliable Distributed Systems (SRDS)*, October 2003. IEEE Computer Society Press: Los Alamitos, CA, 2003; 167-176.
152. Emmerich W, et al. Grid Service Orchestration Using the Business Process Execution Language (BPEL). *Journal of Grid Computing*, 2006; **3** (3): 283–304.
153. Cybok D. A Grid workflow infrastructure. *Concurrency and Computation: Practice and Experience*, 2006; **18** (10): 1243–1254.
154. Turner K, and Tan, K. "Graphical Composition of Grid Services," Proceedings of the Third International Workshop on Rapid Integration of Software Engineering Techniques, May 2007 (*Lecture Notes in Computer Science*, vol. 4401), Guelfi N, Buchs D. (eds.). Springer-Verlag: Berlin; 1-17. DOI 10.1007/978-3-540-71876-5_1.
155. Wassermann B, Emmerich W. Reliable Scientific Service Compositions. *Second International Workshop on Engineering Service-Oriented Applications: Design and Composition (WESOA'06)*, *Proceedings of the 4th International Conference on Service-Oriented Computing (ICSOC)*, December 2006 (*Lecture Notes in Computer Science, vol. 4652*), Georgakopoulos D. et al. (eds.). Springer-Verlag: Berlin, 2007; 14-25. DOI 10.1007/978-3-540-75492-3.
156. Feingold M, Jeyaraman R. *Web Services Coordination (WS-Coordination), Version 1.1*. Organization for the Advancement of Structured Information Standards (OASIS), April 2007.
157. Furniss P. et al. *Business Transaction Protocol Version 1.1.0*, Committee Draft, Organization for the Advancement of Structured Information Standards (OASIS), November 2004.
158. Andrieux A. et al. *Web Services Agreement Specification (WS-Agreement)*, GFD.107, Open Grid Forum, May 2007.
159. 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 (DOI: 10.1002/spe.432).
160. Massie M, Chun B, Culler D. Ganglia Distributed Monitoring System: Design, Implementation, Experience. *Parallel Computing*, 2004; **30** (7): 817–840.
161. 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. et al. (eds.). Springer: Berlin, 2005; 922-931.
162. Byrom R. et al. Fault Tolerance in the R-GMA Information and Monitoring System. *Proceedings of Advances in Grid Computing - EGC 2005: European Grid Conference* (*Lecture Notes in Computer Science 2005, vol. 3470*), Sloot P. et al. (eds.). Springer: Berlin, 2005; 751-760.
163. 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.
164. Arnold K. et al. *The Jini Specification, V1.0*. Addison-Wesley: Boston, 1999.
165. MacLaren J, Keown M, Pickles S. Co-Allocation, Fault Tolerance and Grid Computing. *Proceedings of the UK e-Science All Hands Meeting*, September 2006. Engineering and Physical Sciences Research Council: Swindon, United Kingdom, 2006; 155–162.
166. Gray J, Lamport L. *Consensus on Transaction Commit*. Microsoft Research Corporation, 2004. MSR-TR-2003-96.
167. Hiltunen M, Schlichting R, Ugarte C. Enhancing survivability of security services using redundancy. *Proceedings of the 2001 International Conference on Dependable Systems and Networks*, July 2001. IEEE Computer Society Press: Los Alamitos, CA, 2001; 173–182.
168. Shang L. et al. 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 (CompFrame '07)*, October 2007. ACM Press: New York; 59 – 66.
169. Shafer, Glenn; *A Mathematical Theory of Evidence*, Princeton University Press: Princeton, New Jersey, 1976.
170. Colling D. et al. On Quality of Service Support for Grid Computing. *Proceedings of the Second International Workshop on Distributed Cooperative Laboratories and Instrumenting the GRID (INGRID 2007)*, April, 2007.
171. 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.
172. Davis D. et al. (ed.). *Web Services Reliable Messaging (WS-ReliableMessaging)*, Organization for the Advancement of Structured Information Standards (OASIS), June 2007.
173. Iwasa K. et al. (ed.). *WS-Reliability 1.1*, Organization for the Advancement of Structured Information Standards (OASIS, November 2004.
174. Gudgin M. et al. *SOAP Version 1.2, Part 1: Messaging Framework (Second Edition)*, World Wide Web Consortium, April 2007.
175. Christensen E. et al. *Web Services Description Language (WSDL) Version 1.1*, World Wide Web Consortium, March 2001.
176. 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-Volume1, (ITCC 2005)*, April 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 360-365.
177. Durand, J, Karmarkar A. Message Reliability Protocol Standards for Web Services: An Analysis. *Proceedings of the 3rd IEEE European Conference on Web Services (ECOWS 2005)*, November 2005. IEEE Computer Society Press: Los Alamitos, CA.
178. 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*, July 2004 (*Lecture Notes on Computer Science*, vol. 3095), Bussler C. et al. (eds.). Springer-Verlag, London, 2004; 89-104.
179. Mandrichenko I, Allcock W, Perelmutov T. *GridFTP v2 Protocol Description*, GFD-R-P.047, Open Grid Forum, May 2005.
180. Postel J, Reynolds J. *File Transfer Protocol.* RFC 959, Internet Engineering Task Force, October 1985.
181. Mattmann C. et al. A Classification and Evaluation of Data Movement Technologies for the Delivery of Highly Voluminous Scientific Data Products. National Aeronautics and Space Administration, Document 20060044153, 2006.
182. Lim S. et al. Web Service Robust GridFTP. *Proceedings of the 2004 International MultiConference in Computer Science and Computer Engineering*, June 2004. CSREA Press, 2004; 725-730.
183. Lowekamp B. et al. *A Hierarchy of Network Performance Characteristics for Grid Applications and Services*. GFD-R-P.023 (Proposed Recommendation), Open Grid Forum, May 2004.
184. Lowekamp B. et al. Enabling Network Measurement Portability through a Hierarchy of Characteristics. *Proceedings of the Fourth International Workshop on Grid Computing (GRID'03)*, November 2003. IEEE Computer Society Press: Washington, DC, 2004; 68-75.

185. Fox, G. Collaboration and Community Grids. *International Symposium on Collaborative Technologies and Systems*, May 2006. IEEE Computer Society Press: Los Alamitos, 2006; 419- 428.
186. 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.
187. Gudgin M, Hadley M, Rogers T. *Web Services Addressing Version 1.0*, World Wide Web Consortium, May 2006.
188. Box D. et al. *Web Services Eventing (WS-Eventing)*, draft submission, World Wide Web Consortium, March 2006.
189. Arora A. et al. *Web Services for Management (WS-Management), Version 1.0.0a*, Distributed Management Task Force, April 2006.
190. 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 (MGC '04)*, October 2004. ACM Press, New York, 2004; 63-68.
191. Maassen J, Bal H. Smartsockets: solving the connectivity problems in grid computing. *Proceedings of the 16th international symposium on High performance distributed computing (HPDC '07)*, June 2007. ACM Press, New York, 2007; 1-10.
192. Awduche D. et al. *Overview and Principles of Internet Traffic Engineering*. RFC 3272, Internet Engineering Task Force, May 2002.
193. Moy J. OSPF Version 2. RFC 2328, Internet Engineering Task Force, December 1999.
194. *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.
195. Rekhter Y. (ed.). A *Border Gateway Protocol 4 (BGP-4)*. RFC 1771, Internet Engineering Task Force, March 1995.
196. Rosen E, Viswanathan A, Callon R. *Multiprotocol Label Switching Architecture*. RFC 3031, Internet Engineering Task Force, January 2001.
197. Boyle J. et al. *Applicability Statement for Traffic Engineering with MPLS*. RFC 3346, Internet Engineering Task Force, August 2002.
198. Clapp G, Gannet J, Skoog R. Requirements and Design of a Dynamic Grid Networking Layer. *IEEE International Symposium on Cluster Computing and the Grid*, April 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004; 633-639.
199. TeraGrid, 2008 http://www.teragrid.org/ [January 16, 2008].
200. World's Fastest Network Launched to Connect TeraGrid Sites. *PSC News Center*, February 27, 2003. http://www.psc.edu/publicinfo/news/2003/ [January 16, 2008].
201. AccessGrid Home Page. http://www.accessgrid.org/ [17 December 2007].
202. Nekovee M, Barcellos M, Daw M. Reliable Multicast for the Grid: a Case Study in Experimental Computer Science. *Philosophical Transactions of the Royal Society A*, 2005; **10** (1098): 1775-1791.
203. Levine N, Garcia-Luna-Aceves J. "A comparison of reliable multicast protocols," *Multimedia Systems*, 1998; **6** (5): 334-348.
204. Mankin A. et al. *IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols*. RFC 2357, Internet Engineering Task Force, January 1998.
205. Adamson B. et al. *NACK-Oriented Reliable Multicast (NORM) Protocol (draft)*. Internet Engineering Task Force, March 2007.
206. Macker J. The multicast dissemination protocol (MDP) toolkit. *Proceedings of IEEE Military Communications Conference (MILCOM)*, vol. 1, November 1999. IEEE Computer Society Press: Los Alamitos, CA, 1999; 626–630.
207. Waters G, Crawford J, Lim S. Optimising Multicast Structures for Grid Computing. *Computer Communications*, 2004; **27** (14): 1389-1400.
208. Ranaldo N, Tretola G, Zimeo E. Hierarchical and Reliable Multicast Communication for Grid Systems. *Proceedings of the International Conference on Current & Future Issues of High-End Computing,* June 2005. Central Institute for Applied Mathematics: Jülich, Germany, 2005; 137-144.
209. Chiu D. et al. *TRAM: A Tree-based Reliable Multicast Protocol*. Sun Microsystems Laboratories, 1998. SMLI TR-98-68.
210. Popescu A. et al. 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. Reliable Multicast Transport (rmt) Working Group (Internet Engineering Task Force). http://www.ietf.org/html.charters/rmt-charter.html [January 15, 2008].
212. [212] Nguyen-Tuong A. et al. *Towards Dependable Grids*. Department of Computer Science, University of Virginia, Technical Report CS-2004-11, 2004.
213. [213] Bezzine S. et al. A Fault Tolerant and Multi-Paradigm Grid Architecture for Time Constrained Problems: Application to Option Pricing in Finance. *Second IEEE International Conference on e-Science and Grid Computing*, December 2006. IEEE Computer Society Press: Los Alamitos, CA, 2006; 49.
214. [214] Xie M, Dai Y, Poh K. *Computing Systems Reliability*, Kluwer Academic Publishers: New York, NY, USA, 2004.
215. [215] Lin M, Chang M, Chen D. Distributed-Program Reliability Analysis: Complexity and Efficient Algorithms. *IEEE Transactions on Reliability*, 1999; **48** (1): 87-95.
216. [216] Dai Y, Pan Y, Zou X. A Hierarchical Modeling and Analysis for Grid Service Reliability. *Transactions on Computer*s, 2007; **56** (5): 681-691.
217. [217] Dai Y, Levitin G. Reliability and performance of tree-structured grid services. *IEEE Transactions on* Reliability, 2006; **55** (2): 337-349.
218. [218] 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.
219. [219] 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.
220. [220] 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.
221. [221] Mills K, Dabrowski C. Investigating Global Behavior in Computing Grids. *Proceedings of the Second International Workshop on Self-Organizing Systems (IWSOS),*(*Lecture Notes in Computer Science*, vol. 4124), De Meer H, Sterbenz J. (eds.). Springer-Verlag: Berlin, 2006; 120-136, 2006.

222. [222] Mills K, Dabrowski C. Can Economics-based Resource Allocation Prove Effective in a Computation Marketplace? To appear in *Journa o f Grid Computingl*.
223. [223] Dabrowski C. *Investigating Resource Allocation in a Standards-Based Grid Compute Economy*. National Institute of Standards and Technology: Gaithersburg, MD. Interagency Report 7463, November 2007.

APPENDIX

This table 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 1. References listed by major topical area of grid system reliability.

Fault Tolerance of Grid Resources	
Fault Detection	[10-13, 15-17, 27-50].
Checkpointing and Process Migration	[16, 21, 22, 44, 47, 51-76].
Resource Replication	[16, 44, 47, 77-91].
Rescheduling	[92-94].
Data Replication	[8, 9, 95, 96-108].
Testing and Certification of Grid Resources	
	[109-131].
Reliability of Grid Applications and Workflows	
	[46, 132-158].
Infrastructure and Management Services	
	[35, 85, 103-107, 159-170].
Grid Connection and Transport Reliability	
Reliability Specifications	[2, 90, 171-182].
Fault Tolerance Methods	[14, 80, 90, 182-200].
Reliable Multicasting	[186, 201-211].
Overall System Perspective	
Architecture	[7, 122, 123, 185, 212, 213].
Measurement Methods	[24, 25, 214-219].
Complex Systems	[220-223].