Multiagent and Grid Systems, vol. 2, no. 2, 2006, pp. 115 - 133.
The original publication is available at www.iospress.nl

A New Fault-Tolerance Framework for Grid Computing

Youcef Derbal
School of Information Technology Management
Ryerson University
350 Victoria Street, Toronto, ON, M5B 2K3
Tel: 1(416) 979-5000 x7918
yderbal@ryerson.ca

Abstract:
Fault detection and propagation in a computational grid requires a comprehensive framework that takes in consideration the various grid environmental conditions such as the asynchronous nature of communication and the uncertainty on the disseminated fault information. The paper presents a fault-tolerance framework that provides the necessary models to manage the local faulty behavior associated with the operation of hosted services. The framework includes a quantification mechanism of the fault vulnerability of grid nodes and their hosted services. The resulting measures of fault vulnerability are globally disseminated to enable the synthesis of decentralized fault-tolerant decision making strategies.

Keywords: Computational Grid, Fault-Tolerance, Fault Detector, Reliability, Service Request.

1. Introduction

Computational grids (CGs) are large scale networks of geographically distributed aggregates of service providing resource clusters that often span distinct management domains. As such they are susceptible to a wide spectrum of potential faults where some of which may lead to failures in service provision. The nature of these faults and their causes have been analyzed as part of a taxonomy for dependable and secure computing developed for communication and computing systems such as CGs [5]. A more focused categorization of faults in grids has been articulated to include hardware, software and network related classes of faults [16]. The cumulative effect of these faults translates into job failures, delayed job executions, denials of service, non-compliance with user defined quality of service such as deadline for job execution, and violation of service level agreements. With the adoption of a Service Oriented Architecture (SOA) as supported by the Web Service Resource Framework (WSRF) standard [25], a gird service can be defined as an arbitrarily composed hierarchy of other grid services. The reliable consumption of composed grid services depends on the development of fault-tolerant workflow management strategies that take in consideration the provision reliability of the entire chain of service composition with respect to the classes of faults encountered in grid systems. In this respect, every grid node ought to be able to estimate, in a timely fashion, the reliability of other nodes (and their hosted services) so as to synthesize decisions that are tolerant of their peer's faults. The large scale, dynamical and distributed nature of CGs across geographically remote and distinct management domains poses considerable challenges to the development of such fault-tolerant decision-making mechanisms. Some of these challenges include the ubiquitous uncertainty on fault and resource state information, the intermittent participation of resources, the heterogeneity of resources, network latency, and the lack of central control. Current research has addressed some of these challenges with varied degrees of success [16, 17, 23, 34-36]. However, there are many issues that remain unaddressed; including the uncertainty associated with resource state and fault information, the lack of formulated mechanisms for the timely dissemination of fault information across distinct

management domains, and the absence of quantitative models of node and service reliability. This paper presents a decentralized fault-tolerance framework that accounts for the uncertainty on the resources' state as well as their fault behavior. It includes a distributed network of node-bound probabilistic models to quantify the extent of fault vulnerability of nodes and their hosted services. The resulting quantification of reliability is exchanged among neighboring nodes using a neighborhood-based dissemination mechanism to enable the synthesis of decentralized fault-tolerant management strategies without incurring the drawbacks of a centralized registry.

The paper is organized as follows: section 2 provides a literature review while section 3 presents the proposed fault-tolerance framework. The application of the framework to the development of a fault-tolerant scheduling strategy is given in section 4, followed by simulation results presented in section 5. Related works and conclusions are given in sections 6 and 7 respectively.

2. Literature Review

Research on fault-tolerant grid and high performance computing has traditionally focused on recovery strategies implemented principally through checkpointing and job migration applied to local networks of computing hosts under a centralized management [3, 18, 24, 28, 30-34]. While checkpointing and job migration techniques should be part of any fault-tolerant grid management framework, more emphasis needs to be given to monitoring, detection, and prediction of faults so as to include preventive approaches in addressing the challenges of the grid computing environment. This is particularly important since preventive maintenance measures would diminish the need for frequent checkpointing and complex recovery procedures which may involve rescheduling jobs on different execution environments [34]. Hence, it is not surprising that an increasing attention is being given to prediction-based fault prevention strategies as a lever to limit failure rate in the first place [16, 17, 23, 34-36]. In [34], a preemptive strategy is used to enable the graceful transition of a service/node from a probable fault state, defined by a set of operational conditions that indicate a susceptibility to pending failure, from a normal operating state. In [28], a series of measures are taken to facilitate the implementation of management operations that are robust to faults. One of these measures consists in a periodic observation of job execution servers. If the status of the execution server is not satisfactory, the computation is migrated to a new server. Other measures that were suggested are net yet implemented. These include restart and rollback based on a regular and coordinated checkpointing applied to different execution servers. Lee et al. [23] propose a fault recovery strategy where the migration decision is synthesized based on checkpointing information and the potential performance benefit of selecting a substitute execution site. In [16], dedicated agents associated with various categories of grid faults are used to rejuvenate the system accordingly. For example, once certain conditions about a pending memory shortage are observed, the associated agent migrate the jobs of the affected node to a different node. Other works on fault-tolerance in grids are focused on utilizing job replication to ensure the reliable execution of scheduled jobs [1, 4], while in [35] a scheme of high service availability is implemented through backup replications.

Most of the above surveyed works do not address the uncertainty associated with resource state and fault information. In addition, no clear mechanisms are proposed for the timely dissemination of fault information across distinct management domains. Furthermore, these works don't include any quantitative models of node or service reliability. All these issues are of critical importance to fault-tolerance since it has been shown that the lack of timely dissemination of the dynamic state of resources and their level of reliability is one of the reasons behind job failure or delayed job execution in real grid systems such as the Grid3 [17]. In light of the above survey, there is a need for a comprehensive fault-tolerance framework that integrates the necessary mechanisms for fault detection, propagation and estimation/prediction with proactive fault prevention as well as reactive

fault recovery strategies so as to cover what has been recognized as the central means to attain dependability [5] (see Fig. 1). The framework has to address the cited challenges of the grid environment and enable the development of assurance strategies of the quality of service (QoS). These would undoubtedly be needed for the realization of the grid commercial potential. Quality of service may be defined using performance metrics such as probability of service request handling within a given deadline, expected job failure rate, or mean wait time before service. In order to address the large scale grid distribution and the presence of distinct management domains, the framework has to provide the necessary infrastructure for the integration of decentralized and scalable fault-tolerant grid management strategies. The practical feasibility of such infrastructure depends on the ability of the fault detection and propagation mechanisms to enable the timely estimation of node and service reliability despite the distributed grid topology, the asynchronous and fault prone communication, and network latency. Research on general distributed computing systems has yielded some notable results that are relevant to these issues of fault detection and propagation in grids [7-13, 20-22, 27]. In particular, It was proven that for an asynchronously communicating set of processors it is impossible to reach a consensus about their states in the presence of even a single fault [13]. However, different algorithms have been proposed to yield approximate consensus under various assumptions on the degree of partial synchrony between processors and the rate of faulty to reliable processors [10-12]. Furthermore, [8, 9] introduced the notion of unreliable failure detectors that enables processors to eventually reach consensus about their states provided that some implicit assumptions are made on the synchrony between the parties involved [22]. This proven existence of realizable fault detectors indicates that it is indeed possible to develop grid node-bound mechanisms to detect faulty behavior and propagate the associated information to peer nodes in an asynchronous fashion and still eventually achieve an approximate consensus using probabilistic approaches similar to those proposed in [6, 7]. Achieving such consistent grid-wide view of the state of services and nodes, even in probabilistic terms, would enable an increased effectiveness of the collective resource exploitation through decentralized decision-making processes.

3. Grid Fault-Tolerance Framework

The grid under consideration is assumed to be a federation of service providers (nodes) each making up a distinct management domain. Furthermore, it is assumed that each node is capable of handling a service request that is either submitted by a user or delegated by a peer, provided that it hosts the required service with sufficient capacity.

Consider a User Service Request (USR) submitted to a grid node. Let us assume that such request requires for its handing the availability of a single grid service. Such availability would necessarily go beyond the assertion that the required grid service is indeed deployed. In particular, the hosting environment has to possess sufficient resource availability for the instantiation of the grid service in question, the subsequent invocation of its operations, and the maintenance of it state. The required resources may include CPU slots, RAM, other service components, special hardware devices, disk space, swap space, memory cache as well as any required licenses of application software that the service may need for its successful operation. If the service needs for its execution a specific operating system, some processor architecture, or the presence of the Microsoft .NET framework or the Java Virtual Machine (JVM) and possibly a required heap size, then these would be part of the set of required resources. This resource base supports the set $S = \{s_0, s_1, ..., s_{M-1}\}$ of services deployed on the grid node in question. Although the competing needs of the deployed services are managed through mechanisms of reservation and allocations, the running instances of the various deployed services may exceed their allocated share of resource usage. This may amount to a unintended violation of a Service Level Agreement (SLA) between the consumer of the service and its provider [2] . The reason lies in the inability to make a

categorical a priori assertion about the precise resource usage for the entire set of possible scenarios of run-time behavior associated with a running service instance for a given service request. This is exacerbated by the inaccurate or exaggerated job requirements provided by the user [26]. The run-time exception scenarios are obvious reasons for the excess resource usage beyond the allocated levels. One can conceive of a node control mechanism that monitors the resource usage and terminate the tasks or processes of any service instance that exceeds its allocated resource levels. However, such drastic approach may not be necessarily favorable towards the desire to build a reliable hosting platform that is robust against moderate deviations from some SLA. If these deviations are to be accepted, the mentioned mechanism of resource usage control would have to include some dynamic prediction of the expected profile of resource usage by the violating services. This would enable the synthesis of a more appropriate decision as to the termination of the associated tasks and processes. In all cases, momentary or transient violations of SLAs by one or more services are highly probable. As a result, running service instances are vulnerable to failure before completion of their tasks because of unexpected depletion of resources such as RAM or disk swap for example. The various sources of these failures are illustrated in the service fault tree given in Fig. 2 and inspired by the taxonomy given in [16].
One of the identified sources of faults in a grid environment is resource depletion. The competing needs of hosted services with many spawned instances, and the uncertainty on the specification of their requirements create unforeseen scenarios of contentious resource usage (see Fig. 3). These often lead to resource starvation of running service instances and would ultimately result in timeouts or run-time exceptions.
The multiple fault sources in a grid environment translate, from the service provision view point, into service unavailability, or service failures. In the first case, the service request handling may still be in progress but is hindered by contentious resource usage or depletion. This would ultimately result in a timeout if no corrective intervention is applied. For instance, it may be possible to save the current execution state through checkpointing and resumption of the request handling from the last checkpoint as soon as the service is restored to its normal operational state through graceful restart and re-initialization. In the second case, the service request handling has terminated abnormally before completion of the tasks being executed. As a result, the processing completed up to the failure event would be lost unless checkpointing is periodically applied throughout the lifecycle of the service operation.

3.1 Service and Node Reliability

The coupling between the potential sources of faults and their unpredictable manifestations in a grid environment in addition to the uncertainty on the resource state information and the service operating conditions suggest that deterministic models of fault predictions may be inappropriate and ineffective. This provides a sufficient motivation to view the operational behavior of a service instance as a stochastic process $\{X_n, n = 0,1,2,...\}$ that takes on different states from the finite set Ω of possible operational states described below (see Fig. 4):

- *Robust State (R*): This corresponds to service request handling void of any significant error conditions that might compromise the expected performance or lead to an abnormal termination of the service instance.

- *Vulnerable State (*V): The running service instance exhibits a regular behavior but with degraded performance or error conditions that might lead to failure. Usually, this state of fault vulnerability is reached as a result of process aging of the service instance which may be

caused by many environmental factors including memory leakage and cumulative data corruption [15].

- *Failure State (*F): The service instance is not responding to received requests because of crashed processes, resource depletions such as run-time memory shortage or software bugs in a supporting infrastructures such as application and database servers.

- *Maintenance State (*M): Associated with a preemptive re-initialization of the service instance after completing the associated pending service requests. Once the maintenance is triggered new requests are redirected to another instance of the service, if available, until the maintained service instance is returned to a robust state.

The proposed service maintenance is similar to the concept of rejuvenation of software applications that was first introduced in [15] to address the issue of fault-tolerance for long running applications such as those encountered in telecommunication systems. Service maintenance may be applied to all instances of a service and as such it may include de-fragmentation of disk storage, re-spawning of processes, and restarting of database and application servers. However, unlike software rejuvenation which is regularly scheduled, the proposed service maintenance is to be triggered whenever some identified fault-vulnerable operating conditions are observed by the node management system. Some of these conditions may include a pending run-time memory shortage or the number of open database connections exceeding some upper limit. One clear advantage of the proposed preventive maintenance is the ability to gracefully re-initialize a service instance after checkpointing in order to enable the resumption of request handling from the state where it was halted. As a result, request handling delays would be much less significant than if a service failure occurs unpredictably ensuing loss of work done up to the failure time unless an expensive periodic checkpointing is in place. In addition, the degraded performance of a service or its non-availability caused by a preventive maintenance initiated during periods of low service load would have a lower negative impact on the quality of service provisioning compared to the effect of unpredictable failures that may occur during periods of high service load.

Given the introduced stochastic model of service operation, let us assume that whenever the process enters state $i \in \Omega$ it remains there for a random amount of time having mean μ_i before it goes to state $j \in \Omega$ where it remains for a mean time of μ_j. The resulting process is a non-homogenous semi-Markov process [14, 29]. Furthermore, if whenever a process returns to state $i \in \Omega$ it is said that a cycle has been completed and a reward is given in the amount of time that was spent in state i during the cycle, then the process is also a renewal-reward process [29]. Hence, the probability p_i of being in state i is equal to the proportion of time that the process has spent in state i, namely [29]:

$$p_i = \frac{\mu_i}{\sum_{j \in \Omega} \mu_j} \qquad (1)$$

The mean time μ_i a process stays in state i is estimated at the time index $n \geq 0$ as follows:

$$\mu_i(n) = \frac{\sum_{k=0}^{N_i - 1} \Delta T_k^{(i)}}{N_i} \qquad (2)$$

Where N_i is the number of monitored transitions out of state i up to the discrete time index n, and $\Delta T_k^{(i)}$ is the observed time interval spent in state i prior to the $k-th$ transition. In order to quantify the reliability of a service, a reliability measure $\Phi(s,n):(\Sigma,\mathbb{N})\rightarrow[0 \quad 1]$ is defined as follows:

$$\Phi(s,n) = p_r + p_v \qquad (3)$$

Where p_r and p_v are the probabilities of being in the robust and vulnerable states respectively and are estimated using (1) and (2). Σ represents the set of services hosted by the grid, and $\mathbb{N}$ is the set of natural numbers. Note that given (1) and (2) we have $\sum_{i\in\Omega} p_i = 1$ and hence the relation $0 \le \Phi(s,n) \le 1$ is always true. The above stochastic model of service operation provides the foundation for the development of a grid wide distributed reliability model. Towards this purpose, assumptions on the topology of the target grid system need to be made before defining the proposed models of service and node reliability respectively.

***Definition 1*:** A grid neighborhood is any arbitrary grouping of nodes that regularly share their identity, and service availability information.

As a corollary of the above definition, two nodes are said to be neighbors if there exists a neighborhood of which they are both members. Nodes may belong to more than one neighborhood at a time and may join or leave a neighborhood at will. The identity information includes node ports and IP addresses as well as any other parameters necessary for the establishment of a communication link. The service availability information should, at a minimum, include the names as well as the descriptions and current capacities of hosted services. Preferably, the groupings are made among geographically proximate nodes, partitioning hence the grid into a logical collection of contiguous neighborhoods.

***Definition 2*:** A service s hosted by a grid node h is said to be suspect (unreliable) at time n if and only if $\Phi(s,n) \le \gamma$. Since $\Phi(s,n)$ is a probability measure, $\gamma \in (0 \quad 1]$ should be greater than 0.7 so as to avoid the region of uncertainty.

The above service reliability characterization allows grid nodes to classify the services provided by their peers as reliable or suspect. An appropriate dissemination of such information would enable the synthesis of fault-tolerant decision-making processes that involve the consumptions of the services in question. The reliability of a service as defined above depends on the ability of the hosting environment to apply fault preventative maintenance processes as well as timely recovery strategies to address the various sources of faults listed in the previous section. One reasonable observation might be that the effect of the node-bound fault-tolerance strategies would manifest themselves uniformly on the reliability of all hosted services. This suggests the relevance of defining the notion of node reliability based on the collective status of hosted services.

***Definition 3*:** Let $\Gamma_x^{(y)}(n)$ and $U_x^{(y)}(n)$ be the set of services hosted by node $x \in G$ and deemed reliable and suspect by node $y \in G$ respectively. Then node x is said to be suspect (unreliable) by node y at some discrete time n if and only if:

$$\left|U_x^{(y)}(n)\right| > \beta.\left|\Gamma_x^{(y)}(n)\right| \qquad (4.1)$$

$$\beta = \frac{1}{\left|\aleph^{(y)}\right|} \sum_{z \in \aleph^{(y)}} \frac{\left|U_z^{(y)}(n)\right|}{\left|\Gamma_z^{(y)}(n)\right|} \qquad (4.2)$$

G is the set of gird nodes, $|A|$ denotes the cardinality of set A, and $\aleph^{(y)}$ is the set of nodes neighboring $y \in G$. The choice of β is based on the neighborhood ratio of suspect versus reliable services. The categorization of nodes according to their reliability serves as a prescreening process that enables a fast convergence of decentralized grid decision-making mechanisms. For example, in the case of a decentralized grid scheduling, a suspect node may, in some cases, be eliminated from the list of candidate solutions if it is known to be unreliable. This would take place before any expensive search is undertaken to assert whether such a node hosts a sought after service with sufficient capacity and acceptable reliability.

In this paper, an assumption is made that information about hosted services is maintained in distributed service registries attached to nodes. Furthermore, the content of these registries is assumed to be shared with neighboring nodes at a regular interval of time so as to allow the implementation of a decentralized service discovery strategies. Inadequate update frequency and freshness of the service availability information would affect the performance of decision-making processes such as scheduling. Such effect may translate into service denials if the necessary resources are found to be unavailable once the request reaches the provider node after being delegated on the assumption that this last has the necessary capacity to handle the request. In the case of the proposed framework, the uncertainty on the knowledge of resource state is implicitly accounted for in the proposed models of reliability. Indeed, the observed occurrences of service denials, some of which may be caused by resource state uncertainty, are taken into account in the computation of the state probabilities of the models in question. The associated reliability information is then disseminated as a feedback to potential consumers so as to improve their future scheduling decisions in the face of uncertain grid state information.

3.2 Neighborhood Dissemination of Reliability Information

In a service oriented grid architecture, one of the principal strategies of grid resource exploitation is service request delegation. A user service request submitted to a given node is first assessed for local handling. If the local capacity of the involved service is insufficient, a delegation to a peer-node with sufficient service capacity is performed according to some grid scheduling strategy. In this respect, a delegating node needs to be aware of the reliability of services hosted by peer nodes so as to ensure the maximum likelihood that its delegated requests will be successfully handled even in the face of potential faults resulting from the hosting environment or the jobs associated with the service request itself. For this purpose, the service reliability information needs to be disseminated across the grid. The large scale distribution of a grid requires the dissemination strategy to take in consideration the environmental factors such as limited storage space, network latency and finite bandwidth. The influence of these factors is directly dependent on the density of disseminated information and the scope of dissemination. On one extreme end, a large amount of reliability information universally communicated among all grid nodes would increase the network traffic and latency, and deplete more local storage and processing capacity without necessarily

achieving the desired universal awareness since the increased network latency would render the disseminated information to distant nodes stale. The other extreme end of no dissemination of reliability information would be equally non-intuitive given the wide spectrum of conceivable strategies between the two extremes that may achieve a satisfactory tradeoff between awareness and scalability costs. The proposed dissemination strategy is devised to strike such a balance by limiting the density of communicated information to that of the service reliability measure and the set of suspect neighbors. Furthermore, the scope of dissemination is restricted to the neighboring nodes. In particular, referring to Fig. 5, where the lines linking the nodes define the pathways for the exchange of fault/reliability-related information, the dissemination strategy consists of two elements:

- Each node such as h disseminates the reliability measures associated with its hosted services to its immediate neighbors such as x which it will then use to update the set $\Gamma^{(x)} = \bigcup_{z \in \aleph^{(x)}} \Gamma_z^{(x)}$ of services deemed reliable by x, as well as the set $U^{(x)} = \bigcup_{z \in \aleph^{(x)}} U_z^{(x)}$ of services deemed suspect.
- Each node such as x disseminates the set of suspect nodes $\Upsilon_0^{(x)}$ to all its neighboring nodes. The set $\Upsilon_0^{(x)}$ is compiled based on the maintained set of suspect services and does not include the sets of suspect nodes that have been disseminated by neighboring nodes. Note that the set of suspect nodes known to a node x is equal to $\Upsilon^{(x)} = \Upsilon_0^{(x)} \bigcup_{z \in \aleph^{(x)}} \Upsilon_0^{(z)}$.

Given the above dissemination approach, the following relationships hold true for the bounds on the size of the lists of suspect nodes and suspect services respectively:

$$\left|\Upsilon^{(x)}(n)\right| \le N_{\max} + N_{\max}^2 \qquad (5)$$

$$\left|U^{(x)}(n)\right| \le N_{\max}.M_{\max} \qquad (6)$$

$M_{\max}$ is the maximum number of services that can be hosted by a node, and $N_{\max}$ is the maximum number of neighbors that a grid node may have. Relation (5) results from the fact that in the worst case, a node x would deem unreliable all of its neighbors and would receive from each one of its neighbors a list that reflects the worst case scenario where they in turn deem all their neighbors to be unreliable. The bound given in (6) sprung from the worst case scenario where every service hosted by every neighboring node is unreliable. Note that the set of reliable services can be inferred from the set of suspect services since each node has full knowledge about the service offerings of its neighbors. The same applies to the set of suspect nodes and the set of reliable nodes. As a result, only one category of information (suspect or reliable) needs to be maintained for nodes and services. Using relations (5) and (6), the size RSZ of the registry that holds the information about service and node reliability can be bounded as follows:

$$RSZ \le \left(N_{\max} + N_{\max}^2\right)v + N_{\max}M_{\max}u \qquad (7)$$

Where u is the required storage space associated with the reliability information of a single service entry which includes the service name, and the node ID. v is the storage space associated with the reliability information of a single node which in this case is no more than the ID of the suspect node. Allocating 32 Bytes for the service name and 4 Bytes for the storage of a node ID respectively would result in $u = 36$ Bytes and $v = 4$ Bytes. Fig. 6 illustrates the bound on the

required storage size of the registry as a function of the maximum number of services and the maximum number of neighbors respectively. In practical terms, the registry would be implemented using a relational database or an LDAP directory bound to the node. Judicious caching of the registry in the node's run-time memory would ensure a faster information retrieval. However, since the registry is local to the node and exclusively accessible to its management mechanisms, even in the absence of caching the queering performance would not degrade with the increase of users or gird size as would be the case for a centralized grid registry that holds reliability information.

3.3 Framework Integration

The grid fault-tolerance framework outlined conceptually in Fig. 1 provides a comprehensive approach to fault-tolerant provision and consumption of grid services. As outlined above, the strategies and models associated with the framework are decentralized requiring hence an independent integration with the grid node management system. Fig. 7 illustrates a nominal node architecture geared towards the realization of the framework. As a whole, the framework encourages local prevention and recovery measures with global awareness of the reliability of peer service offering. The current work is focused on some elements of the framework, in particular the models of reliability and the dissemination strategy. As a result, the reactive recovery strategies and issues are not addressed. However, one issue that needs clarification is the open nature of the framework towards the inclusion of various methods of fault prevention and recovery. This is possible partly because of the use of a reliability model that considers the overall operation of deployed services as opposed to individual jobs, processes or resources such as CPUs or RAM. However, such reliability model is still being driven by the monitoring of the individual resources and running processes underlying the operation of hosted services. Furthermore, there are no restrictions in the framework as to the information and the strategies that might be used to trigger a proactive maintenance. Similarly, the framework can accommodate any checkpointing and recovery strategy that can be implemented in a decentralized fashion.

4. Application to Fault-Tolerant Grid Scheduling

The proposed fault-tolerance framework and its service and node models of reliability provide a quantification of the hosting environment's ability to support a reliable provision of hosted grid services. This section explores the application of the framework to the synthesis of fault-tolerant grid scheduling decisions. For this, let us assume that a user submitted a service request, denoted u_{sr}, to node $h \in G$. u_{sr} requires the availability of a single instance of grid service s^* for which node h does not have the necessary capacity. A service discovery scheme has yielded a solution set Θ of grid nodes whose hosting environments meet the requirement of the submitted u_{sr}. Given the dissemination scheme of reliability information, the set Θ can be partitioned into seven subsets Θ_0, Θ_1, Θ_2, Θ_3, Θ_4, Θ_5, and Θ_6 as shown in Table 1. The partitioning is based on the fact that nodes don't have access to any reliability information about services and their hosting nodes if these are located within a distance greater than two hops away. Furthermore, nodes don't have access to the reliability information about services other than those hosted by neighbors. Given these constraints, the question then is: What are the nodes, among the elements of Θ, that would provide the most reliable handling of the service request if it is delegated to them?

Using the partition of the solution set obtained through a discovery process, the following fault-tolerant delegation algorithm implements a multi-step approach to the search for a target node y^* hosting a reliable instance of the required service.

```
Begin
Θ ← serviceDiscovery(h, u_sr)
► If SCHEDULING_TIMEOUT || Θ = ∅
    GOTO End
  End If
  (Θ_0, Θ_1, Θ_2, Θ_3, Θ_4, Θ_5, Θ_6) ← partition(h, Θ)
  If Θ_0 ≠ ∅ Then
    y* ← randElementChoice(Θ_0)
    GOTO End
  Else If Θ_1 ≠ ∅ Then
    y* ← randElementChoice(Θ_1)
    GOTO End
  Else If Θ_4 ≠ ∅ Then
    h ← randElementChoice(Θ_4)
    Θ ← Θ_4
    GOTO ►
  Else If Θ_5 ≠ ∅ Then
    h ← randElementChoice(Θ_5)
    Θ ← Θ_5
    GOTO ►
  Else
    h ← randElementChoice(Θ_6)
    Θ ← Θ_6
    GOTO ►
  End If
  End
```

The function *randElementChoice*(.) returns a randomly chosen element of the input set, and *serviceDiscovery*(.) returns a solution set of nodes that host the required service with a sufficient capacity. The underlying strategy of the algorithm views the reliability of the service to be more important than that of the node. This is motivated by the fact that a hosting environment may provide a more fault-tolerant handling of a select subset of services to the exclusion of all the other services. In this case, the node may in fact be classified as suspect, while the select subset of services is still categorized as reliable. Note that the scheduling of the service request is not performed if the solution set Θ does not include a node that offers a reliable service. Instead, a delegation to chosen nodes is performed based on their proximity. In this choice the subsets Θ_2 and Θ_3 are excluded since they are known to host suspect instances of the service in question. After delegation, a repartitioning of the reduced solution set is performed based on the reliability information available to the new home node and the selection process is repeated until the service request is scheduled or a scheduling timeout period elapses. The above scheduling approach is conservative and may be improved using a more optimal choice of the solution set of nodes. However, the issue at hand is not the performance of the scheduling algorithm but rather its fault-tolerance. In this respect, the above scheduling strategy is fault-tolerant in the sense that it limits

the scheduling of a service request to nodes for which there is an assurance about their expected reliability or that of the target services they host.

5. Simulation Results

The proposed fault-tolerance framework is validated through a simulation of the developed fault-tolerant scheduling approach. The Midland Grid Emulator developed by the Author is used for this purpose (see Fig. 8). The emulated grid may be configured for an arbitrary number of nodes and deployed services respectively. The stochastic model of service operation is executed as a Java daemon thread. An exponential distribution is used to simulate the inter-arrival time of events that compromise the operational integrity of the service. For a balanced fault injection the same rate is used for all nodes and services, while different randomly selected rates for different nodes and services are used for an unbalanced fault injection.

Service vulnerability is emulated using a counter that takes values between 0 and 100. Beyond 25% of the counter's maximum value the service is considered vulnerable and enters the failure state once the 100% level is reached (see Fig. 9). Each arrival of a compromising event increases the counter by a certain percentage of its maximum value while recovery and maintenance operations reset the counter after a random amount of time spent in the failure and maintenance states respectively. The proposed stochastic model of Fig. 4 includes a direct transition between the robust and failure states. However, this is not considered in the simulation with the assumption that real systems rarely succumb to catastrophic failures without experiencing a prior state of vulnerability observed through error and fault conditions.

In order to account for the bursty nature of real grid service request distributions, the number of simultaneously arriving service requests is simulated using a Pareto process where the range and the shape are set to 100 and 1 respectively. The inter-arrival time of the Pareto distributed service request load for the various nodes is simulated using a Poisson process.

The goal of the fault-tolerance framework is to maintain a desirable level of service availability in the presence of faults as well as to limit delays and abnormal terminations that are caused by unpredictable failures. These desirable operational properties of the hosting environment contribute to the overall reliability of service provisioning as experienced by consumer nodes. In order to quantify the performance of the framework with respect to these properties, the following fault-tolerance indicators are defined:

$$\eta_f = \frac{n_f(T)}{n_R(T)} \qquad (8)$$

$$\eta_d = \frac{n_d(T)}{n_R(T)} \qquad (9)$$

η_f is the failure ratio and η_d is the ratio of service denial resulting from the unavailability of requested services. n_R is the total grid-wide number of service requests submitted within the interval T of time. n_f is the total grid wide number of failed service requests during the interval T of time, and n_d is the total grid wide number of service requests that could not be handled because of service unavailability during the interval T of time.

The commercial grid potential relies on the ability to construct an open service provisioning system where new resources can be added as needed in order to handle increased user load or more computationally demanding applications. The realization of such potential is dependent on the scalability of the grid decision-making mechanisms such as service discovery, scheduling and load

balancing. In this respect, the contribution of the framework to the fault-tolerant properties of these mechanisms ought to lock steps with the scalability performance of these lasts in terms of the above defined metrics.

Using the defined fault-tolerance indicators, the performance of the proposed fault-tolerant random delegation scheduling (FTRDS) strategy is compared to a similar strategy of random delegation scheduling (RDS) that does not rely on the reliability information disseminated among neighbors. It is important to note here that random selection of available resources, on which both FTRDS and RDS are based, may not be the best scheduling strategy. However, the focus here is not on the performance of the scheduling strategy but rather on the performance differential that results from the utilization of the fault-tolerance framework.

In the first set of simulations, both RDS and FTRDS rely on a service discovery mechanism that supplies a candidate set of nodes that are believed to have the required service capacity at the time of the delegation decision. Figs 10 and 11 illustrate the ratios of failure and service denial for various grid sizes with a balanced fault injection throughout the grid. For an unbalanced fault injection the ratios of failure and service denial are given in Figs 12 and 13. The simulation results illustrate the substantial improvement induced by the utilization of the fault-tolerance framework. For both balanced and unbalanced fault injections the failure ratio is reduced by up to 50% from its level obtained for the RDS strategy. Similarly, the FTRDS strategy exhibits a dramatic improvement with respect to the ratio of service denials when compared to the RDS strategy. The reliance on neighborhood based delegation and dissemination of state information implies that both scheduling strategies use a resource set made up of closer nodes. Consequently, an increase in the size of the grid does not have drastic effects on the scheduling performance in general and the fault tolerant property in particular as illustrated in Figs 10-13.

The effectiveness of the preventive maintenance and the effect of the maintenance threshold on the ratios of failure and service denials are illustrated in Figs 14 and 15. The simulation is conducted for a grid of 30 nodes where the maintenance threshold was varied from 25% to 95% of the vulnerability level where a failure would immediately follow. The simulation results suggest that the lowest level of failure ratio is achieved when the maintenance threshold is closest to the lower bound of the vulnerability zone. This is plausible since a lower threshold of maintenance results in a high frequency of maintenance and would increase the service immunity against crossing the vulnerability threshold into the failure state. As the maintenance threshold is increased, such immunity is decreased resulting in an increased failure ratio for both balanced and unbalanced fault injections (Figs 14 and 15). For service denials which equally cost the provider in terms of reliability and dependability of their service provision, a higher frequency of maintenance (lower maintenance threshold) may be expected to result in more service denials incurred during maintenance downtime. This pattern, although weakly exhibited for the case of unbalanced fault injection, is overall not strongly manifested in the simulation results. One reason for this might be the cumulative and nonlinear effect of failures on service denials. Indeed, as the failure ratio increases with a decreased maintenance frequency, the increased occurrences of failures would result in the unavailability of corresponding services. Consequently, the anticipated decrease of service denials resulting from less maintenance downtime is not materialized because of the effect of service unavailability caused by failures.

6. Related Works

Some surveyed works focus on the synthesis of fault-aware decision making mechanisms while others are concerned with system recovery after failure. Given the costly checkpointing associated with the later approach, proactive fault-tolerant strategies that minimize unpredictable failures have also been explored in the literature. These works are related in various degrees to the

proposed approach of fault-tolerant grid management. However, there is a limited number of research efforts that are strongly related to this work in their focus on monitoring-based proactive fault-tolerance approaches [16, 17, 19, 26]. In [17] various scheduling strategies were considered where feedback information about the state of resources and the reliability of their operations is utilized. The monitored information includes the number of queued jobs, the number of cancelled jobs as well as the number of unfinished jobs associated with a given grid node. Using this information, measures such as node load and node reliability where defined and subsequently used to inform the scheduling process. Experimental tests on the Grid3 system showed that feedback-based scheduling reduces the number of job resubmissions and delivers a better job completion time compared to open-loop scheduling. These results illustrate the intuitively expected fault-tolerant quality of a feedback scheduling approach. However, given the small number of nodes making up the grid (25 nodes) the scalability of the approach cannot be asserted. Furthermore, the job related node performance information is utilized in an ad-hoc fashion. No attempt is made to develop a model that captures the dynamics of node reliability so as to provide a more effective prediction of its behavior. Such model would be necessary as grid systems are expected to provide services that rely on extensive infrastructures such as application servers, database servers, and specialized devices. In this respect, in addition to the job-related performance information, the model in question would have to accommodate resource state and reliability information such as server throughput, transaction rates, service up-time, and network health metrics. In addition to the absence of an integrated model of reliability that addresses the challenges encountered in grid systems, the approach reported in [17] is formulated for a single management domain avoiding hence the consideration, otherwise necessary, of the dissemination of state and reliability information across a distributed grid system that spans distinct management domains.
In [16], dedicated agents associated with the various categories of grid faults are used to rejuvenate the system accordingly. For example, once certain conditions about a pending memory shortage are observed, the associated agent migrate the jobs of the affected node to a different node. This agent-oriented approach is promising because of the potential for the development of a scalable, distributed network of agents able to independently correct local failures or reduce the vulnerability towards their occurrences while being guided by some broader guidelines of QoS formulated at the grid level. However, the results reported in [16] are limited to the consideration of a centrally managed cluster and an application bound implementation of the fault-tolerant services. In [19], the failure detection is based on a dynamic grouping of nodes with an elected leader that detects a failure in the group through a timeout applied to a heart beat mechanism. Once a failure is detected by a leader, this last broadcast the associated information to all group leaders in the grid as well as the members of its group. While the detection scheme is scalable, the dynamic grouping is susceptible to instability during node joining or leaving. Furthermore, the awareness of node failures is not sufficient to quantify node reliability with respect to partial failures associated with process crashes or failures of some classes of services hosted by the node. In other works, it has been noted that given the unavoidable system failures and the inaccurate or exaggerated job requirements provided by the user, it is impossible for a grid system to provide a 100% QoS guarantee [26]. This observation has provided the motivation for a proposed fault-aware scheduling approach for a supercomputing cluster whereby the user and the system negotiate a probabilistic QoS guarantee such as: “job x shall be completed by deadline d with a probability of p” [26]. In order to achieve such guarantee, the scheduling strategy utilizes a set of algorithms that predict critical events that lead to job failures based on monitored health information about node state and load. In particular, the prediction mechanism provides an estimate of the probability that a submitted job with an expected execution time will fail if scheduled on a given node. The job is scheduled if the QoS guarantee is deemed achievable otherwise the user is consulted on whether they wish to relax the QoS requirement in favor of a

higher probability of success. The simulation of a cluster made up of 125 nodes suggests an improved reliability and performance of the system. The underlying idea of negotiating a desirable risk strategy between the user and the supercomputing cluster is a pragmatic approach that gives due consideration to the uncertainty on the expected grid performance. However, its applicability to a grid system that spans multiple administrative domains would require a decentralized performance prediction strategy and an adjunct mechanism of dissemination of performance and reliability information.

Compared to these related works the proposed fault-tolerance framework equally recognizes that fault-tolerant decision making strategies benefit from state feedback information about resource usage, operational performance, fault conditions, and network health. However, instead of an ad-hoc utilization of state feedback information, monitored state information is used in the proposed framework to drive a synthesized model that characterizes the reliability of nodes and services. In contrast to the mentioned related works, the proposed fault-tolerance framework is decentralized in nature so as to address the presence of distinct administrative domains within a grid. Overall, the proposed framework is distinguished from the surveyed related works by it comprehensive approach to the quantification of service and node reliability in addition to the provision of a light mechanism for the dissemination of reliability information so as to enable the implementation of decentralized and scalable fault-tolerant management strategies.

7. Conclusion

A decentralized fault-tolerance framework for grid computing is proposed. It relies on a distributed network of node-bound probabilistic models of service and node reliability respectively. The associated reliability information is selectively exchanged among neighboring nodes using a dissemination mechanism that is shown to be light in its requirement for data exchange density and overhead storage. Comparative simulation results show that the proposed framework is scalable and improves the performance of a grid scheduling strategy with respect to fault-tolerance. Furthermore, the proposed node-bound proactive maintenance strategy is shown to reduce service failure ratio without increasing the ratio of service denial beyond the level that would be experienced in the absence of a preventive maintenance of service operation. For future works, further elaboration of the model of service operation is the next logical step towards the practical implementation of the framework. In particular, it would be valuable to study the effect on service denials and failures of a dynamically set maintenance threshold trigger that is based on some appropriate model of the hosting environment's variables such as load, resource utilization, and past operational behavior. This may, for example, be used to implement a maintenance regime that avoids high load periods and achieve as a result a lower service downtime.

Acknowledgments

I am grateful to Professors Aziz Guergachi and Ojelanki Ngenyama for gracefully sharing the computing resources used to run the simulations reported in this paper.

References

[1] J. H. Abawajy, Fault-tolerant scheduling policy for grid computing systems, in: Proceedings - International Parallel and Distributed Processing Symposium, IPDPS 2004, IEEE Computer Society, Los Alamitos, United States, 2004, pp. 3289-3295.

[2] R. Al-Ali, A. Hafid, O. Rana, and D. Walker, An approach for quality of service adaptation in service-oriented Grids, Concurrency Computation Practice and Experience 16 (2004), 401-412.
[3] G. Allen, W. Benger, T. Goodale, H.-C. Hege, G. Lanfermann, A. Merzky, T. Radke, E. Seidel, and J. Shalf, Cactus Code: a problem solving environment for the grid, in: IEEE International Symposium on High Performance Distributed Computing, Proceedings, Institute of Electrical and Electronics Engineers Inc., Piscataway, NJ, USA, 2000, pp. 253-260.
[4] C. Anglano and M. Canonico, Fault-tolerant scheduling for bag-of-tasks grid applications, in: Lecture Notes in Computer Science, Springer Verlag, Heidelberg, D-69121, Germany, 2005, pp. 630-639.
[5] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Transactions on Dependable and Secure Computing 1 (2004), 11-33.
[6] M. Ben-Or, Another advantage of free choice (Extended Abstract): Completely asynchronous agreement protocols, in: Proceedings of the second annual ACM symposium on Principles of distributed computing, 1983, pp. 27 - 30.
[7] G. Bracha and S. Toueg, Asynchronous consensus and broadcast protocols, Journal of the ACM 32 (1985), 824-840.
[8] T. D. Chandra and S. Toueg, Unreliable failure detectors for reliable distributed systems, Journal of the ACM 43 (1996), 225-267.
[9] T. Deepak Chandra, V. Hadzilacos, and S. Toueg, Weakest failure detector for solving consensus, Journal of the ACM 43 (1996), 685-722.
[10] D. Dolev, C. Dwork, and L. Stockmeyer, On the minimal synchronism for distributed consensus, Journal of the Association for Computing Machinery 34 (1987), 77-97.
[11] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl, Reaching approximate agreement in the presence of faults, Journal of the Association for Computing Machinery 33 (1986), 499-516.
[12] C. Dwork, N. Lynch, and L. Stockmeyer, Consensus in the presence of partial synchrony, Journal of the Association for Computing Machinery 35 (1988), 288-323.
[13] M. J. Fischer, N. A. Lynch, and M. S. Paterson, Impossibility of distributed consensus with one faulty process, Journal of the Association for Computing Machinery 32 (1985), 374-382.
[14] S. S. Gokhale, T. Philip, and P. N. Marinos, Non-homogeneous Markov software reliability model with imperfect repair, in: Proceedings -IEEE International Computer Performance and Dependability Symposium, IPDS, IEEE, Los Alamitos, CA, USA, 1996, pp. 262-270.
[15] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, Software rejuvenation: analysis, module and applications, in: Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing (FTCS-25), 1995, pp. 381 - 390.
[16] M. T. Huda, H. W. Schmidt, and I. D. Peake, An Agent Oriented Proactive Fault-Tolerant Framework for Grid Computing, in: Proceedings of the First International Conference on e-Science and Grid Computing, 2005, pp. 304- 311.
[17] J.-U. In, P. Avery, R. Cavanaugh, L. Chitnis, M. Kulkarni, and S. Ranka, SPHINX: A fault-tolerant system for scheduling in dynamic grid environments, in: Proceedings - 19th IEEE International Parallel and Distributed Processing Symposium, Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States, 2005, pp. 12b-12b.

[18] K. A. Iskra, F. Van Der Linden, Z. W. Hendrikse, B. J. Overeinder, G. D. Van Albada, and P. M. A. Sloot, The implementation of dynamite - An environment for migrating PVM tasks, Operating Systems Review (ACM) 34 (2000), 40-55.

[19] A. Jain and R. K. Shyamasundar, Failure detection and membership management in grid environments, in: Proceedings - IEEE/ACM International Workshop on Grid Computing, IEEE Computer Society, Los Alamitos, CA 90720-1314, United States, 2004, pp. 44-52.

[20] P. Jayanti, T. D. Chandra, and S. Toueg, Cost of graceful degradation for omission failures, Information Processing Letters 71 (1999), 167-172.

[21] P. Jayanti, T. D. Chandra, and S. Toueg, Fault-tolerant wait-free shared objects, Journal of the ACM 45 (1998), 451-500.

[22] M. Larrea, A. Fernandez, and S. Arevalo, On the implementation of unreliable failure detectors in partially synchronous systems, IEEE Transactions on Computers 53 (2004), 815-828.

[23] H. M. Lee, S. H. Chin, J. H. Lee, D. W. Lee, K. S. Chung, S. Y. Jung, and H. C. Yu, A resource manager for optimal resource selection and fault tolerance service in grids, in: 2004 IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2004, Institute of Electrical and Electronics Engineers Computer Society, 2004, pp. 572-579.

[24] M. J. Litzkow, M. Livny, and M. W. Mutka, Condor - a hunter of idle workstations, in: Proceedings - International Conference on Distributed Computing Systems, IEEE, Piscataway, NJ, USA, 1988, pp. 104-111.

[25] OASIS, Web Services Resource Framework (WSRF) - http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-01.pdf, (2005).

[26] A. J. Oliner, L. Rudolph, R. K. Sahoo, J. E. Moreira, and M. Gupta, Probabilistic QoS guarantees for supercomputing systems, in: Proceedings of the International Conference on Dependable Systems and Networks, Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States, 2005, pp. 634-643.

[27] M. Pease, R. Shostak, and L. Lamport, Reaching Agreement in the presence of faults, Journal of the ACM 27 (1980), 228-234.

[28] J. S. Plank, H. Casanova, M. Beck, and J. J. Dongarra, Deploying fault tolerance and task migration with NetSolve, Future Generation Computer Systems 15 (1999), 745-755.

[29] S. M. Ross, Introduction to Probability Models, Academic Press, Inc., San Diego, CA, 1989.

[30] S. S. Vadhiyar and J. J. Dongarra, Self adaptivity in Grid computing, Concurrency Computation Practice and Experience 17 (2005), 235-257.

[31] S. S. Vadhiyar and J. J. Dongarra, SRS: A framework for developing malleable and migratable parallel applications for distributed systems, Parallel Processing Letters 13 (2003), 291-312.

[32] L. Wang, K. Pattabiraman, Z. Kalbarczyk, R. K. Iyer, L. Votta, C. Vick, and A. Wood, Modeling coordinated checkpointing for large-scale supercomputers, in: Proceedings of the International Conference on Dependable Systems and Networks, Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States, 2005, pp. 812-821.

[33] N. Woo, H. Jung, H. Y. Yeom, T. Park, and H. Park, MPICH-GF: Transparent checkpointing and rollback-recovery for grid-enabled MPI processes, IEICE Transactions on Information and Systems E87-D (2004), 1820-1828.

[34] G. Wrzesinska, R. V. Van Nieuwpoort, J. Maassen, and H. E. Bal, Fault-tolerance, malleability and migration for divide-and-conquer applications on the grid, in: Proceedings - 19th IEEE International Parallel and Distributed Processing Symposium, Institute of

Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States, 2005, pp. 13a-13a.

[35] X. Zhang, D. Zagorodnov, M. Hiltunen, K. Marzullo, and R. D. Schlichting, Fault-tolerant Grid services using primary-backup: Feasibility and performance, in: Proceedings - IEEE International Conference on Cluster Computing, ICCC, Institute of Electrical and Electronics Engineers Inc., New York, NY 10016-5997, United States, 2004, pp. 105-114.

[36] M. Zulkernine and R. E. Seviora, A compositional approach to monitoring distributed systems, in: Proceedings of the 2002 International Conference on Dependable Systems and Networks, IEEE Computer Society, 2002, pp. 763-772.

Multiagent and Grid Systems, vol. 2, no. 2, 2006, pp. 115 - 133.
The original publication is available at www.iospress.nl

Youcef Derbal, B. Eng., M.Sc., Ph.D. (Electrical and Computer Engineering, Queen's University). In his dissertation, Dr. Derbal investigated the stability of nonlinear dynamic systems and the design of neuro-adaptive control structures. For over a decade, Dr. Derbal worked in various information and computing industries. His research interests are focused on the various decision-making mechanisms underlying grid systems, as well as the application of grid computing solutions to the simulation of large scale models of environmental and biological systems.

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="2">Neighboring Node</td><td colspan="2">Non-Neighboring Nodes
(located two hops away)</td><td>Distant Nodes
(located more than two hops away)</td></tr>
<tr><td>Reliable</td><td>Suspect</td><td>Reliable</td><td>Suspect</td><td rowspan="3">Θ_6</td></tr>
<tr><td rowspan="2">Required Service</td><td>Reliable</td><td>Θ_0</td><td>Θ_1</td><td rowspan="2">Θ_4</td><td rowspan="2">Θ_5</td></tr>
<tr><td>Suspect</td><td>Θ_2</td><td>Θ_3</td></tr>
</table>

Table 1: Partitioning of a scheduling solution set based on the available reliability information.

Fig. 1: Overview of a Fault-Tolerant Grid Management Framework.

Fig. 2: Service Fault Tree.

Fig. 3: The dynamic behavior of running service instances are coupled because of their dependence on the shared resources of the hosting environment.

Fig. 4: Stochastic model of service operation.

Fig. 5: Dissemination of reliability information.

Fig. 6: Bound on the storage size of the registry as a function of the number of hosted services illustrated for a maximum number of neighbors set to 25, 50, 75 and 100.

Fig. 7: Integration of the fault-tolerance framework within the grid node management system.

Fig. 8: Midland Grid Emulator.

Fig. 9: Emulation of service operation.

Fig. 10: Failure Ratio as a function of grid size for balanced fault injection.

Fig. 11: Ratio of service denial as a function of grid size for balanced fault injection.

Fig. 12: Failure ratio as a function of grid size for unbalanced fault injection.

Fig. 13: Service denial ratio as a function of grid size for unbalanced fault injection.

Fig. 14: Ratios of service denials and failures as a function of the maintenance threshold for balanced fault injection.

Fig. 15: Ratios of service denial and failures as a function of maintenance threshold for an unbalanced fault injection.

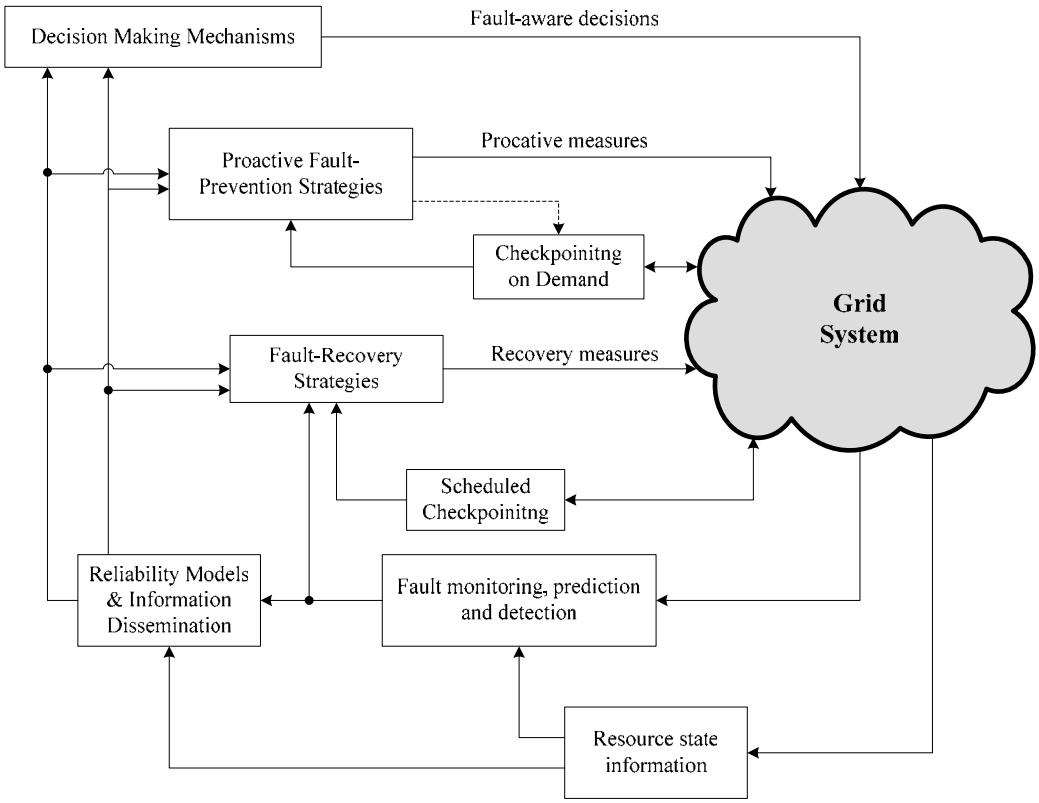
Decision Making Mechanisms
Fault-aware decisions
Proactive Fault-Prevention Strategies
Procative measures
Checkpoinitng on Demand
Grid System
Fault-Recovery Strategies
Recovery measures
Scheduled Checkpoinitng
Reliability Models & Information Dissemination
Fault monitoring, prediction and detection
Resource state information

Fig. 1

Multiagent and Grid Systems, vol. 2, no. 2, 2006, pp. 115 - 133.
The original publication is available at www.iospress.nl

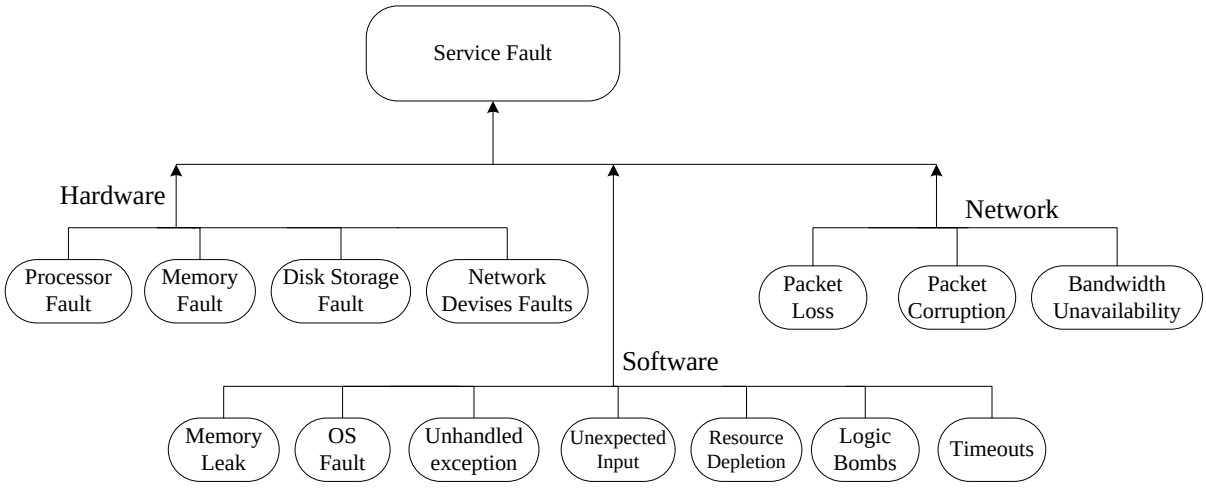

Fig. 2

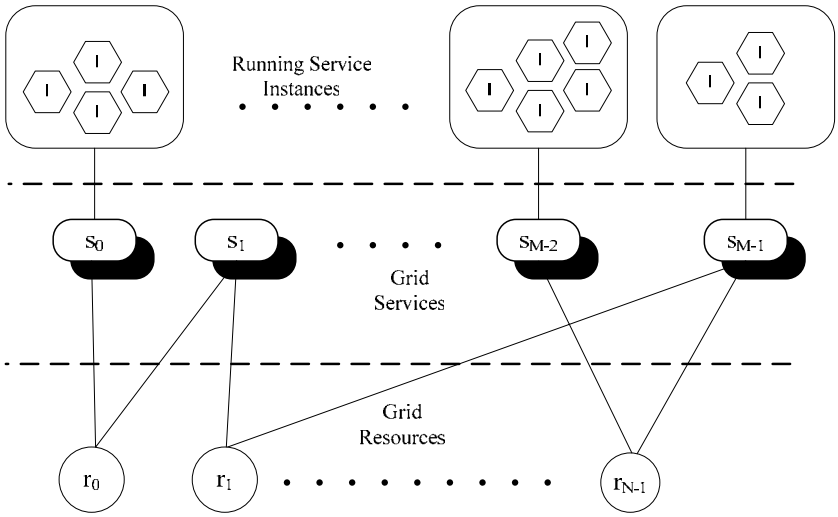
Running Service
Instances
s_0
s_1
s_{M-2}
s_{M-1}
Grid
Services
Grid
Resources
r_0
r_1
r_{N-1}

Fig. 3

Multiagent and Grid Systems, vol. 2, no. 2, 2006, pp. 115 - 133.
The original publication is available at www.iospress.nl

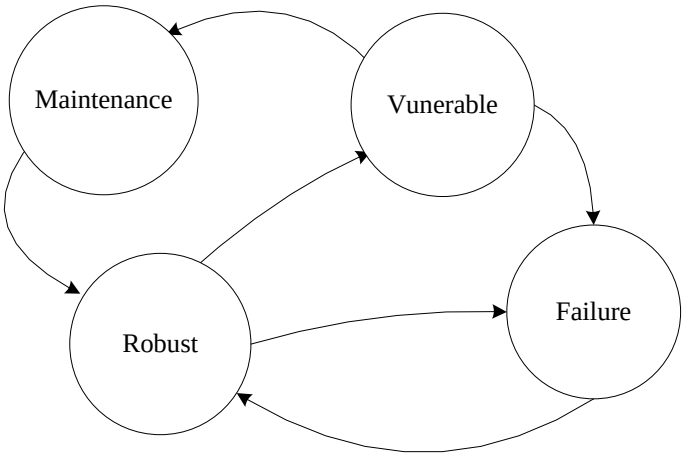

Fig. 4

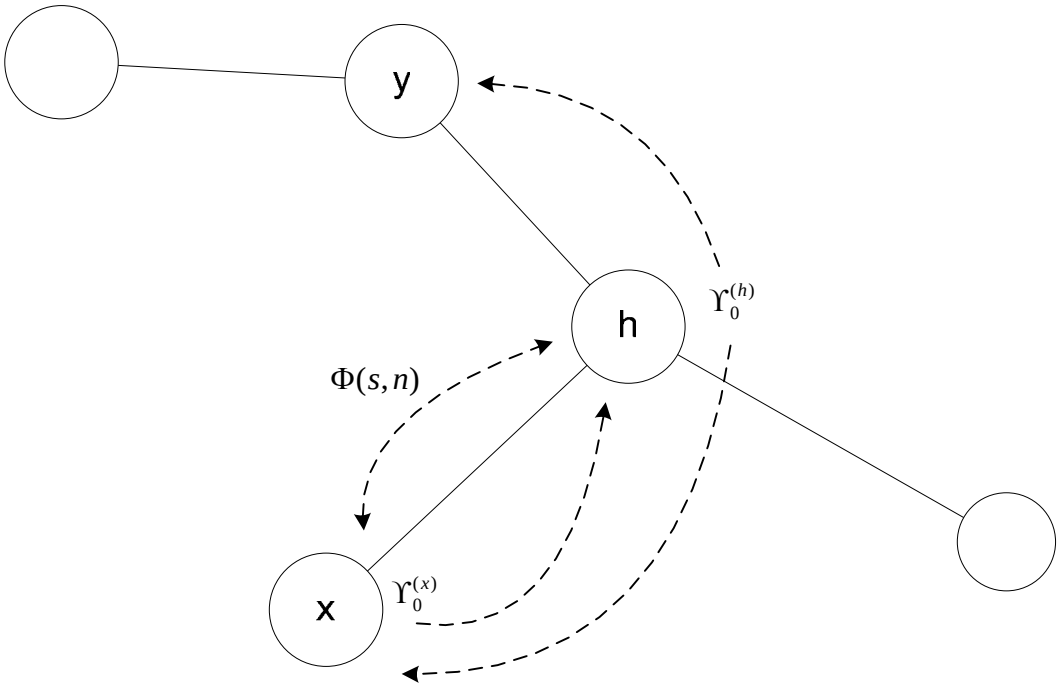

Fig. 5

Multiagent and Grid Systems, vol. 2, no. 2, 2006, pp. 115 - 133.
The original publication is available at www.iospress.nl

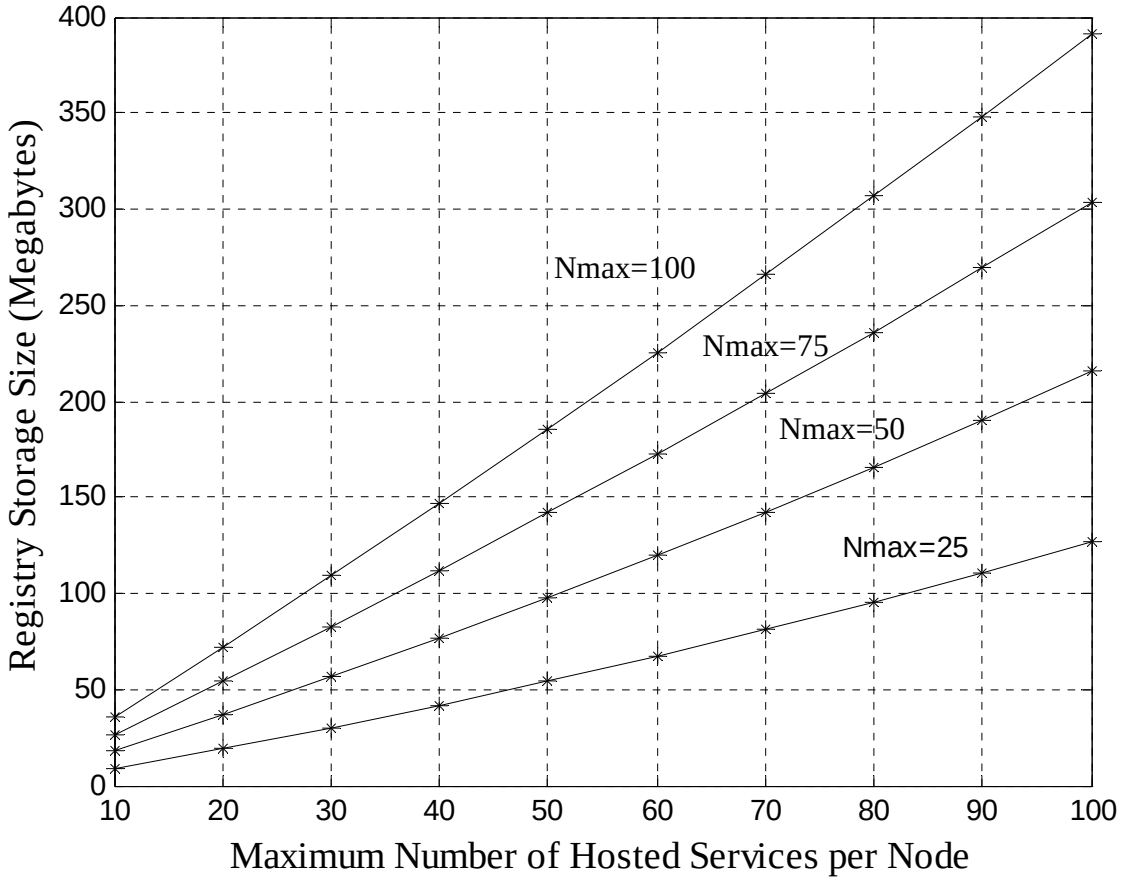

Fig. 6

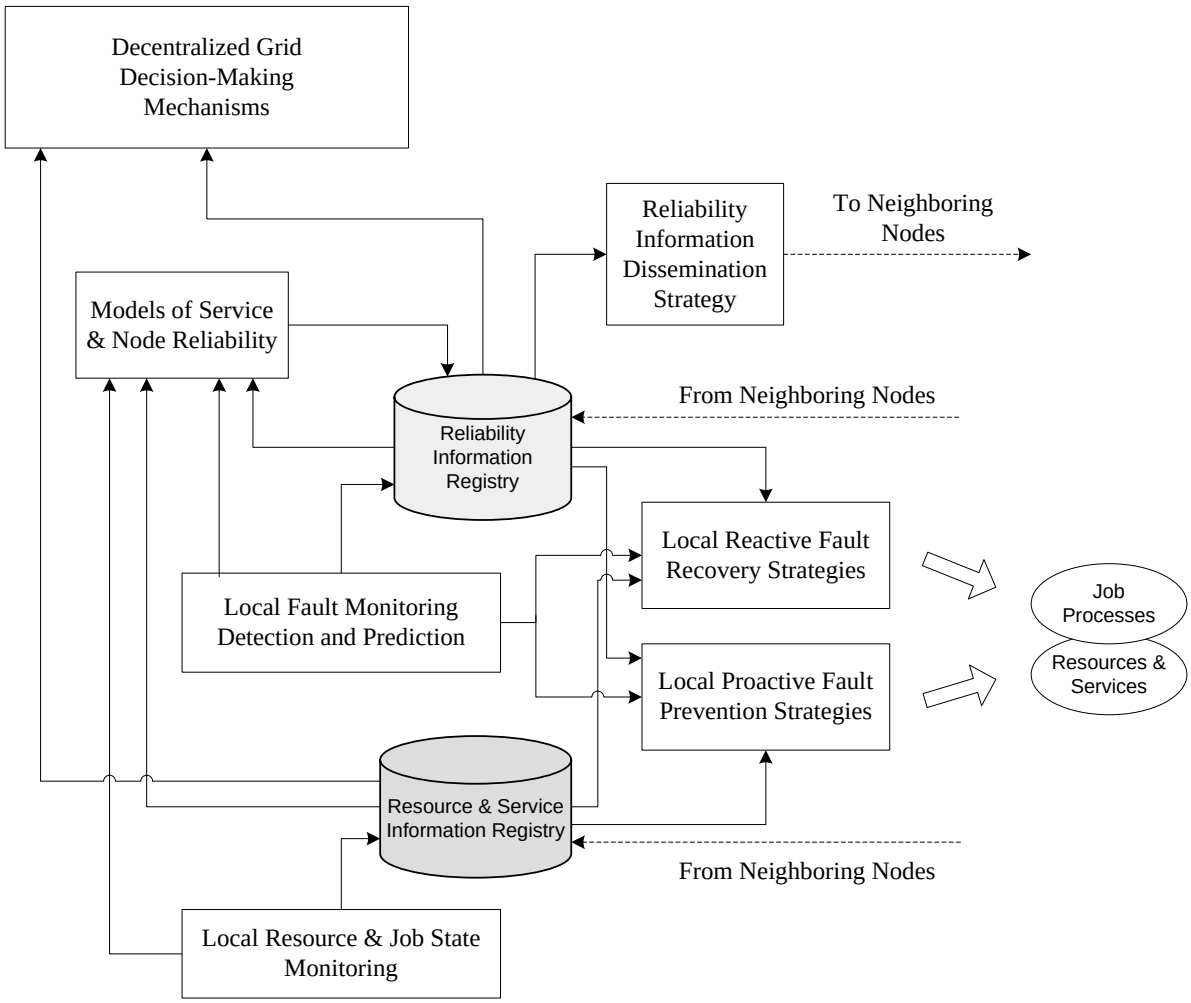
Decentralized Grid Decision-Making Mechanisms
Reliability Information Dissemination Strategy
To Neighboring Nodes
Models of Service & Node Reliability
From Neighboring Nodes
Reliability Information Registry
Local Reactive Fault Recovery Strategies
Job Processes
Local Fault Monitoring Detection and Prediction
Local Proactive Fault Prevention Strategies
Resources & Services
Resource & Service Information Registry
From Neighboring Nodes
Local Resource & Job State Monitoring

Fig. 7

Multiagent and Grid Systems, vol. 2, no. 2, 2006, pp. 115 - 133.
The original publication is available at www.iospress.nl

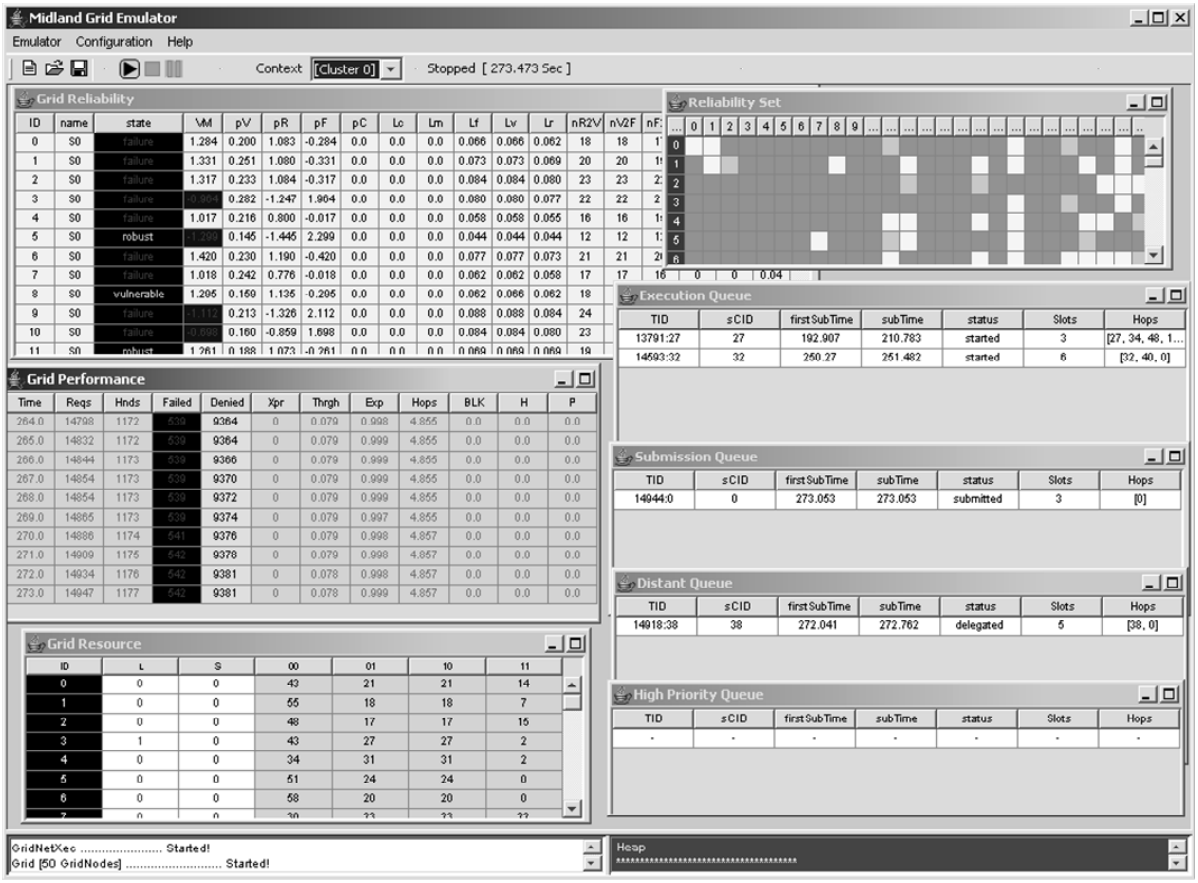

Fig. 8

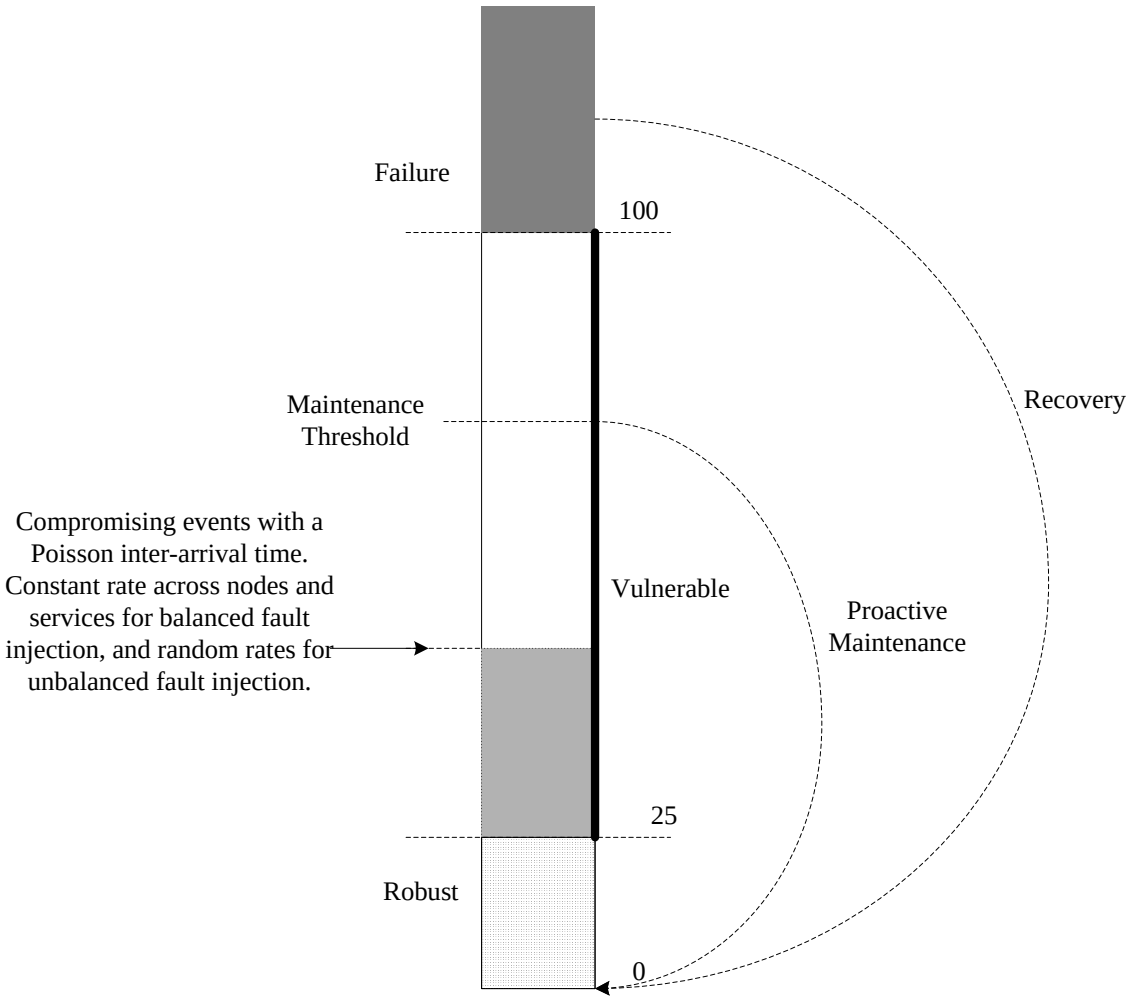
Failure
100
Maintenance
Threshold
Recovery
Compromising events with a
Poisson inter-arrival time.
Constant rate across nodes and
services for balanced fault
injection, and random rates for
unbalanced fault injection.
Vulnerable
Proactive
Maintenance
25
Robust
0

Fig. 9

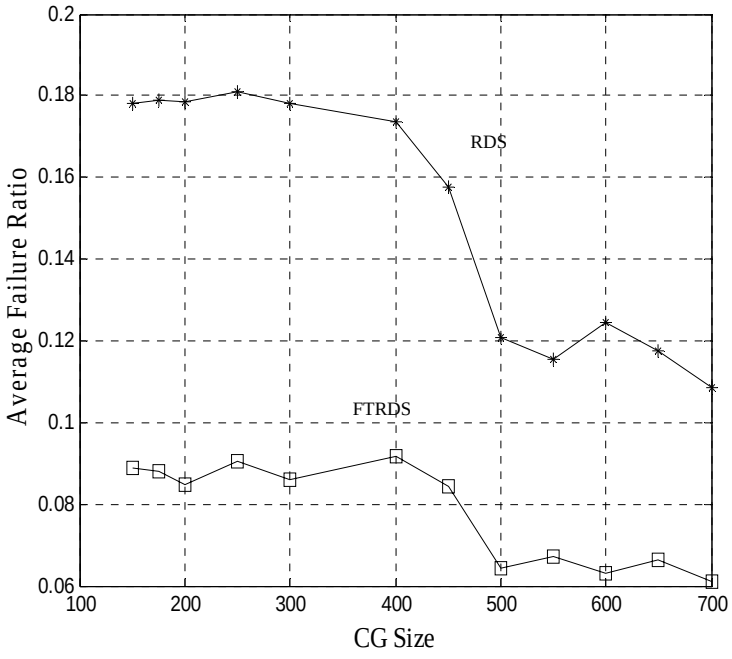
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
Average Failure Ratio
RDS
FTRDS
100
200
300
400
500
600
700
CG Size

Fig. 10

Multiagent and Grid Systems, vol. 2, no. 2, 2006, pp. 115 - 133.
The original publication is available at www.iospress.nl

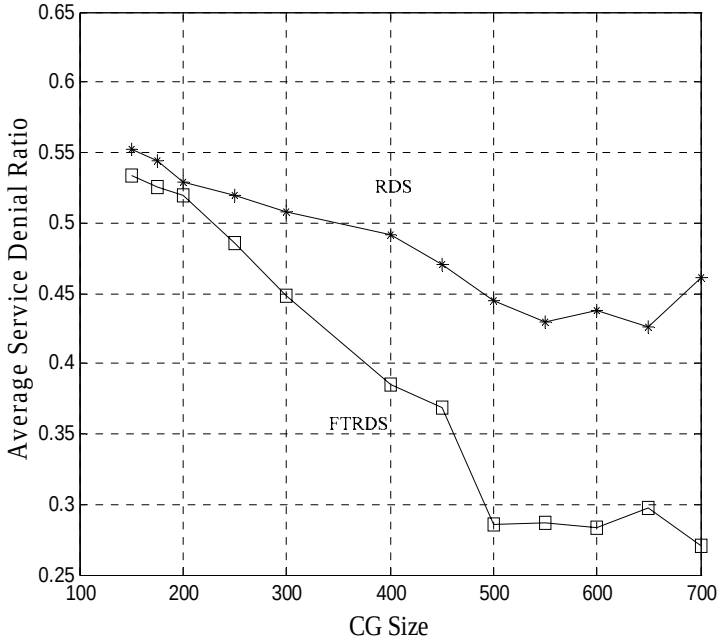

Fig. 11

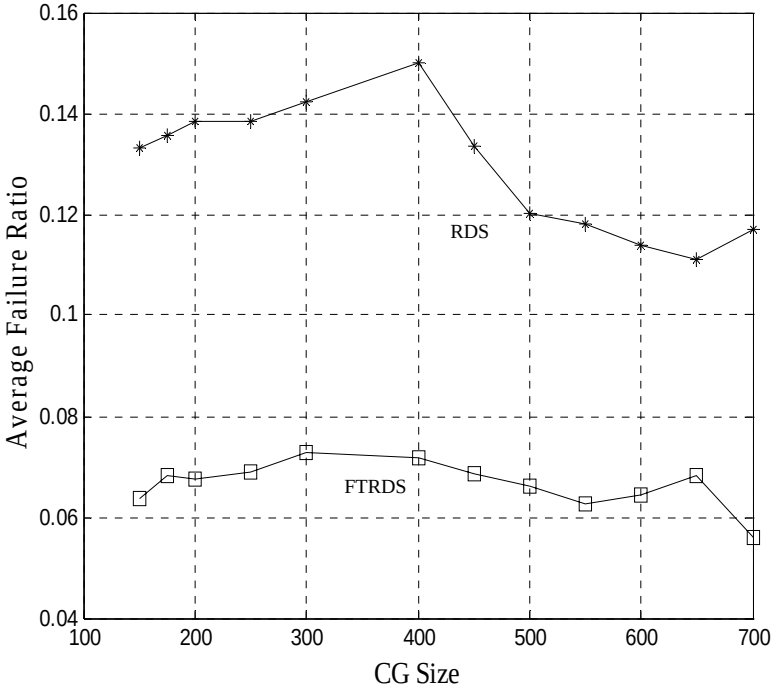

Fig. 12

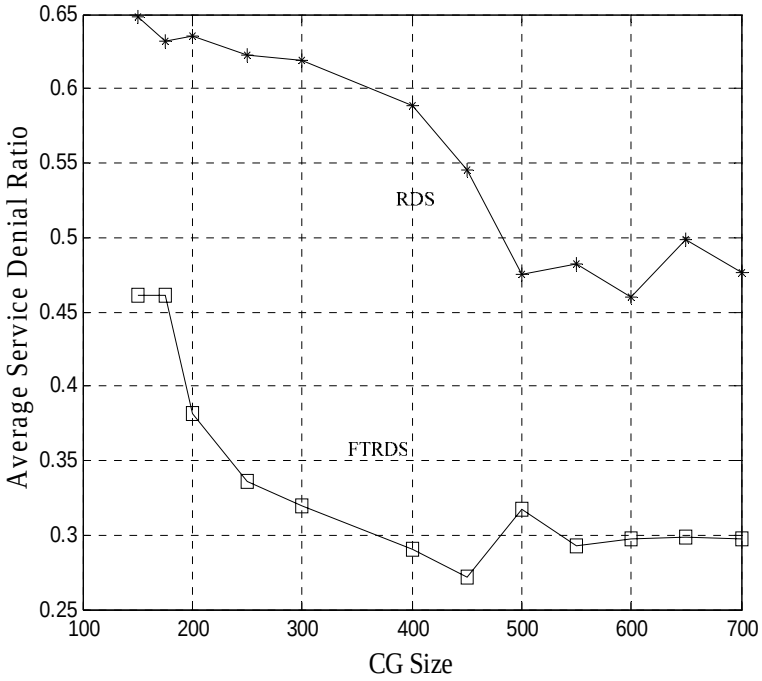
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
100
200
300
400
500
600
700
Average Service Denial Ratio
CG Size
RDS
FTRDS

Fig. 13

Multiagent and Grid Systems, vol. 2, no. 2, 2006, pp. 115 - 133.
The original publication is available at www.iospress.nl

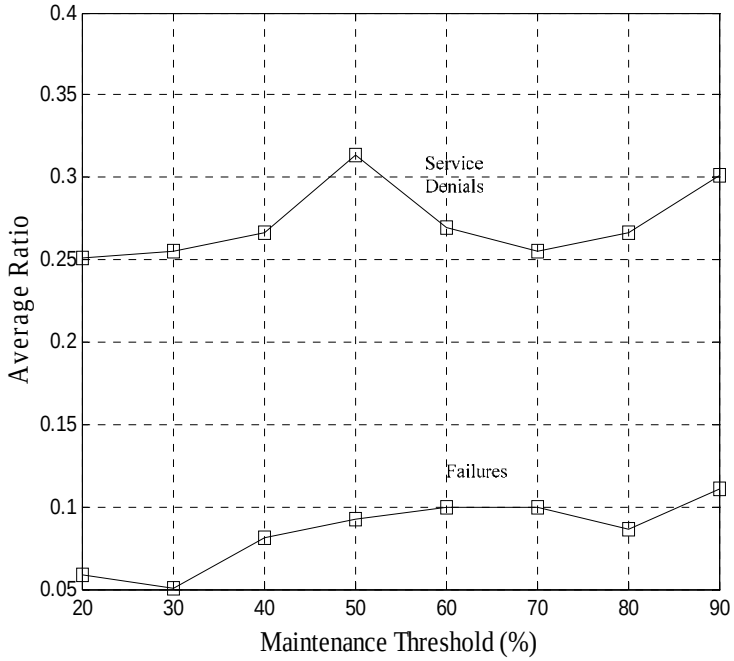

Fig. 14

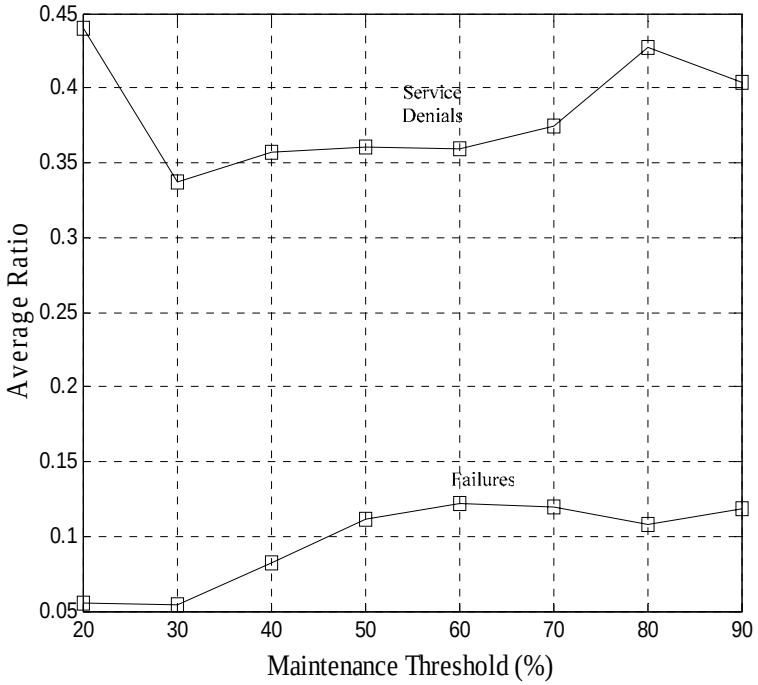
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
Average Ratio
Service
Denials
Failures
20
30
40
50
60
70
80
90
Maintenance Threshold (%)

Fig. 15