# Decentralized QoS-Aware Checkpointing Arrangement in Mobile Grid Computing

Paul J. Darby III, *Student Member*, *IEEE*, and Nian-Feng Tzeng, *Fellow*, *IEEE*

**Abstract**—This paper deals with decentralized, QoS-aware middleware for checkpointing arrangement in *Mo*bile *G*rid (MoG) computing systems. Checkpointing is more crucial in MoG systems than in their conventional wired counterparts due to host mobility, dynamicity, less reliable wireless links, frequent disconnections, and variations in mobile systems. We've determined the globally optimal checkpoint arrangement to be NP-complete and so consider *Re*liability *D*riven (ReD) middleware, employing decentralized QoS-aware heuristics, to construct superior checkpointing arrangements efficiently. With ReD, an MH (*m*obile *h*ost) simply sends its checkpointed data to one selected neighboring MH, and also serves as a stable point of storage for checkpointed data received from a single approved neighboring MH. ReD works to maximize the probability of checkpointed data recovery during job execution, increasing the likelihood that a distributed application, executed on the MoG, completes without sustaining an unrecoverable failure. It allows collaborative services to be offered practically and autonomously by the MoG. Simulations and actual testbed implementation show ReD's favorable recovery probabilities with respect to *R*andom *C*heckpointing *A*rrangement (RCA) middleware, a QoS-blind comparison protocol producing random arbitrary checkpointing arrangements.

**Index Terms**—Checkpointing, computational Grids, mobile Grid systems, decentralized checkpointing, simulation and testbeds.

## 1 Introduction

Grid computing systems have seen their widespread adoption lately in not only academia but also in industry, as evidenced by commercial offerings from Sun [1], HP [2], and IBM [3], among others. While most existing Grids refer to clusters of computing and storage resources which are wire-interconnected for offering utility services collaboratively, *Mo*bile *G*rids (MoGs) are receiving growing attention and expected to become a critical part of a future computational Grid involving mobile hosts to facilitate user access to the Grid and to also offer computing resources [4]. A MoG can involve a number of *m*obile *h*osts (MHs), i.e., laptop computers, cell phones, PDAs, or wearable computing gear, having wireless interconnections among one another, or to access points. Indeed, a recent push by HP to equip business notebooks with integrated global broadband wireless connectivity [5], [52], has made it possible to form a truly *mobile Grid* (MoG) that consist of MHs providing computing utility services collaboratively, with or without connections to a wired Grid.

Current trends toward powerful multicore processors, efficient, small flash memory devices, and wireless technologies, such as IEEE 802.16 WiMAX (which are capable of delivering 10 Mbps or more over distances of miles), are seen as technological enablers of the practical MoG, while models for compensation, accounting, and regulation of these systems are being developed. For example, some authors have proposed a fair pricing strategy and optimal job allocation scheme for MoG computing including the Nash Bargaining solution to maximize Grid revenue from the user viewpoint [34]. Other researchers proposed a middleware using mobile agents for secure MoG services, addressed the heterogeneity concerns of this technology, and provided compatible interfaces to the Globus Toolkit [35]. Also underway are active research projects entertaining the use and practical applications of MoG concepts. In particular, the K*Grid project, aiming to provide a research environment for both industry and academia, has studied the use of idle resources for a great number of mobile devices and the development of a MoG platform [36]. Additionally, the AKOGRIMO project developed a blueprint for the Next Generation Grid based upon a Mobile Collaborative Business Grid model and the Mobile Dynamic Virtual Organization (MDVO) [37]. The distributed applications likely to run on these MoGs are not the traditional long-running heavy computing jobs commonly found in their wired counterparts, but instead, are lightweight, distributed mobile agents (e.g., location-aware and specifically designed, intelligent, reflective collaborative agents acting at the request of a client on the conventional Grid or within the MoG itself) whose functions will enhance both the values of the MoG and the entire global Grid [39], [40], [41], [42]. Because sensors (e.g., GPS, compasses, accelerometers, etc.) are commonly included in many mobile devices, lightweight applications will be capable of providing local, collaborative, and interpretive processing of disparate sensor data from a number of MHs, in order to intelligently surmise macro events for MoG or global Grid clients [50], [51]. For example, "Land Warrior" [56] is a poignant use case, providing integrated battle plan management to soldiers in the field. It incorporates wearable computers, GPS, cameras, and helmet-mounted flip-down displays capable of digitally displaying maps with locations of both friendly and enemy troops. Collaborative computing is required to update maps in real-time battle scenarios. Other use cases include collaborative mobile gaming,

• *The authors are with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, PO Box 43890, Lafayette, LA 70504-3890. E-mail: pauldarby@aol.com, tzeng@cacs.louisiana.edu.*

*Manuscript received 9 June 2008; revised 16 July 2009; accepted 4 Dec. 2009; published online 22 Apr. 2010.*
*For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2008-06-0224.*
*Digital Object Identifier no. 10.1109/TMC.2010.80.*

context-aware applications for internetworked cars, and military advantages such as "Cooperative Engagement Capability" [53], [54], [55].

Due to mobility and intermittent wireless link loss, all such scenarios call for robust checkpointing and recovery to support execution, minimizing execution rewind, and recovery rollback delay penalties. Depending upon the *application's* or *job's* tolerance for such delay (i.e., a QoS metric), its performance can be poor or it can be rendered totally inoperative and useless. Our *Re*liability *D*riven middleware, ReD, allows an MoG scheduler to make informed decisions, selectively submitting job portions to hosts having superior checkpointing arrangements in order to ensure successful completion by 1) providing highly reliable checkpointing, increasing the probability of successful recovery, minimizing rollback delay, and 2) providing performance prediction to the scheduler, enabling the client's specified maximum delay tolerance to be better negotiated and matched with MoG resource capabilities. Suitable for scientific applications, MoGs are particularly useful in remote areas where access to the wired Grid is infeasible, and autonomous, collaborative computing is needed [4]. MoG hosts have no guaranteed access to fixed, wired, Grid access points. Checkpointing is thus crucial for practical and feasible job completion, for without it, the MoG's potential is severely limited.

Checkpointing saves intermediate data and machine states periodically to reliable storage during the course of job execution. Various checkpointing mechanisms have been pursued for distributed systems (whose computing hosts are wire-connected) [6], [7], [8], [9], [10], [11], [12], [13]. However, they are not suitable for the MoG because their checkpointing arrangements 1) are relatively immaterial as checkpointed data from hosts can be stored at a designated server or servers, since connections to a server are deemed reliable, of high bandwidth, and of low latency, and 2) fail to deal with link disconnections and degrees of system topological dynamicity. In contrast, a MoG highly desires its checkpointed data to be kept all at neighboring MHs rather than remote ones who require multiple, relatively unreliable, hops to transmit checkpoints and to reach checkpointed data when it is needed. Earlier wireless checkpointing methods stored checkpointed information at fixed, stationary hosts, on the wired Grid (i.e., *b*ase *s*tations, (BSs)) via access points [14], [15], [19], [20], [21], [22], [23], [24], [25], [26], [27]. This is suitable only for systems where every MH can reach a BS in one hop, an unlikely scenario if the MoG is to be realized practically.

This paper proposes an autonomic, decentralized, QoS-aware, middleware, whose function is to establish checkpointing arrangements among MHs dynamically within the MoG, allowing its constituent MHs to support practical collaborative computation. As wireless links are less reliable and MHs move at will, a given job executed by multiple MHs collaboratively, relies on efficient checkpointing to enable execution recovery upon a MoG component failure by transferring recently saved intermediate data and machine states to a substitute MoG component, so that execution can resume from the last checkpoint, saved prior to the failure (instead of starting all over again from the very beginning). Our checkpointing methodology requires no BS to achieve its function as checkpointing is handled within the MoG by keeping checkpointed data from a given MH at immediate neighboring MHs. Our methodology also facilitates encapsulation of the checkpointing function within the MoG, making it transparent to the wired-Grid or mobile client being served. Thus, in order to limit the use of relatively unreliable wireless links, and further to minimize the consumption of wireless host's memory resources and energy, each MH sends its checkpointed data to one selected neighboring MH, and also serves to take checkpointed data from one approved neighboring MH, realizing a decentralized form of checkpointing. Each MH in a connected MoG concurrently runs both consumer and provider algorithm functions, and is thus capable of being both a provider and a consumer of checkpointing services for some other MHs. Obviously, there are numerous ways to assign checkpointing consumers and providers among all constituent MHs within a MoG, referred to as *checkpointing arrangements*.

We show in this paper that the choice of checkpointing arrangement has significant bearing upon MoG system reliability and thus QoS, *Q*uality *o*f *S*ervice (namely, the probability that an application will complete feasibly within the bounds of the client's a priori specified limit in terms of time frame, reliability, etc.) expected by a user's application. Different arrangements yield differing probabilities that the checkpointed data will survive and be recoverable in the presence of host failure, or more likely and more frequently, link failure or inadvertent and intermittent disconnection of a host or hosts from the MoG. As a MH may be connected to multiple neighboring MHs by wireless links with varying reliability figures, one may choose to save checkpointed data from a given MH to a neighbor connected by the most reliable link or to a neighbor which is best connected to the rest of the MoG so that checkpointed data will have the greatest likelihood of being accessible should it need to be recovered. Further, it is crucial to checkpoint to a well-connected and reliable neighbor, any MH with low remaining battery capacity or poor connections to the rest of the MoG, or any MH which may be in jeopardy of being disconnected from the MoG altogether (e.g., moving out of range). An efficient arrangement thus aims to identify superior assignments of providers and consumers of checkpointing services, among all MHs within the MoG in order to ensure collaborative job execution can complete reliably with smaller probability of experiencing unrecoverable failures. This paper shows that the globally optimal checkpointing arrangement is an NP-complete problem [32]. ReD is thus based upon a heuristic formulation that attains arrangements of superior reliability. Its efficient convergence is promoted by a derived and supportive, simple clustering algorithm, allowing concurrent operation on small clusters of hosts individually, rather than on a single large global system.

The rest of this paper is organized as follows: Section 2 outlines related checkpointing work in wire-connected Grid systems and wireless systems with BSs. Section 3 discusses our proposed Reliability Driven (ReD) middleware, its underpinnings and efficient algorithm, and describes our simulator and the testbed implementations used to evaluate its performance. Section 4 provides simulation and testbed results, comparing ReD with a QoS-Blind methodology, called *R*andom *C*heckpoint *A*rrangement protocol, (RCA). Section 5 concludes the article.

## 2 Related Work

At any time during job execution, a host or link failure may lead to severe performance degradation or even total job abortion, unless execution checkpointing is incorporated. Certain hosts may suspend execution while waiting for intermediate results (as input to their processes) that may never arrive due to the host or link failure.

Checkpointing forces hosts involved in job execution to periodically save intermediate states, registers, process control blocks, messages, logs, etc., to stable storage. This stored checkpoint information can then be used to resume job execution at a substitute host chosen to run the recovered application in place of the failed host. Upon host failure or inadvertent link disconnection, job execution at a substitute host can then be resumed from the last good checkpoint. This crucial function avoids having to start job execution all over again from the very beginning in the presence of every failure, thus substantially enhancing the performance realized by grid applications.

### 2.1 Checkpointing in Wired Grid Systems

Checkpointing in wired Grid systems has been investigated earlier with various methodologies proposed [6], [7], [8], [9], [10], [11], [12], [13], where hosts are connected by low latency, high-speed, wired links having low link and host failure rates [8], [11]. Diskless checkpointing sends checkpointed data to a cluster neighbor (instead of a local disk), in an attempt to reduce checkpointing time overhead on a LAN [9]. This works if message transmission time is less than disk-write time, a realistic possibility for a wired network. The method fails, however, to consider which neighbor or network path is best used to reach storage. Recent work on portable checkpointing for wired Grids assumes a centralized "middleware" support for applications, checkpointing, and recovery [10]. However, the MoG often requires decentralized ad-hoc support of checkpointing and recovery, due to its highly unreliable wireless connections and mobile environment.

Checkpointing in wired large-scale Grid systems generally assumes that the computation interval and checkpoint overhead are much smaller than the mean time between failures (MTBF) [11]. With the MoG, however, this assumption does not hold, since relatively high link and host failure rates (and disconnections) make its MTBF correspondingly low. Hence, maintaining the desired checkpoint interval and checkpointing overhead versus MTBF tradeoff is certainly more critical in MoGs. Existing work has considered various forms of (and their enhancements to) coordinated or uncoordinated checkpointing on large systems, placing application-driven control on the checkpoint interval [12]. Nonetheless, it failed to address checkpointing arrangement partly because in wired Grid systems, how to store checkpoint information and who is given priority to store it, is not deemed important, unlike the MoG case. One noteworthy point is that recent research in wired Grid systems proposes a methodology whereby potential alternative performance costs pertaining to checkpointing interval and designated repository choices are compared prior to automatically setting these parameters [48]. While this research relates checkpointing arrangement choices to job execution performance, and is potentially useful and practical in wired Grid systems, it utilizes statistical data from histories of wired computer resources offered in cycle sharing arrangements gathered over extended time periods, and is not suited to the MoG, where mobility and wireless dynamicity require nearly real-time checkpointing arrangement decisions leading to a fundamentally different controlling paradigm.

### 2.2 Checkpointing in Wireless Systems with BSs

Mobile devices will be an integral part of distributed computing as their computational and storage abilities grow. Wireless communications advances, leading to high bandwidth and robustness, will enable such devices to practically operate as part of the computational Grid. Hence, checkpointing in wireless computing systems has received growing attention, with solution approaches treated [14], [15], [17], [19], [20], [21], [22], [23], [24], [25], [26], [27]. Specifically, a checkpointing tool for the Palm Operating System has been developed [14] providing a set of APIs to enable checkpointing functionality on top of the Palm OS. It is useful because the Palm OS causes a reset of the handheld computer upon power loss. With this methodology, checkpointed data must be stored on stable safe storage (i.e., a computer server or PC, dubbed a *b*ase *s*tation, BS, on a wired network). The methodology is supported by recent routing mechanisms that interconnect inadvertently partitioned ad-hoc MH networks [18].

Almost all, earlier, work on checkpointing in wireless computing systems uses BSs for stable storage. The use of BSs to realize synchronous checkpointing is proposed to augment the checkpointing process [17]. Also, a method for reliable message logging has been suggested [27], where each MH client communicates with the BS through a proxy for checkpointed data storage, masking crashes. Because coordinated recovery among MH processes may slow down recovery, increasing the chance for multiple rollbacks [26], certain mitigating techniques have been adopted, using replication and other methods to augment and support checkpointing [16]. Replicas, storing checkpointed data, resume execution when primary hosts fail, but the replica selection method is not addressed.

Checkpointing wireless MHs to BSs has its own drawbacks, however, when not all MHs are adjacent to BSs or when BSs do not exist (like the MoG at hand). Mobility is a major impediment to moving checkpointed data from MH to BS. A complication is that routes between MH and BS change frequently due to varying wireless links, complete and intermittent disconnections, and mobility. The frequent need for multihop relays of checkpoint messages to access wired storage can lead to heavy traffic, significant latency, and needless power consumption due to collisions and interference. Consequently, a mechanism has been developed to reduce the number of checkpoint transmissions to only those actually requested and related to the needs of the distributed application [20].

Also, a moving MH does not associate with a fixed, unique BS, so successive checkpoints may be found on different BSs, creating a need to locate the BS(s) during recovery [15]. While simple and commonly adopted, the use of BSs for checkpointed data storage is not always advantageous or feasible as BS(s) may be absent, or completely out of range. BS use does not hide or encapsulate the MoG lower level checkpointing function within the MoG, needlessly allowing the MoG volatility and complexity to impact the wired Grid. This is incompatible with more recent MoG architectures where encapsulation of virtual MoG clusters allows for access through proxies, to manage MoG activities (i.e., real-time scheduling, and task migration by low-level brokers) [46], [47]. This prompts our investigation into keeping checkpointed data at MHs themselves, highly attractive if link bandwidth, MH capacity, and reliability are of key concern. We intend to make ReD either compatible with broker interaction via encapsulating proxy interfaces to OGSA compatible wired-Grid clients (e.g., using Globus) when BS access exists, or to operate autonomously otherwise.

## 3 Decentralized Checkpointing in the MoG

This work focuses on the MH checkpointing arrangement mechanism, seeking superior checkpointing arrangements to maximize the probability of distributed application completion without sustaining an unrecoverable failure. It deals with MoG checkpointing among neighboring MHs without any access point or BS. Our main focus in checkpointing arrangement lies in MH connectivity, with host failure being the failure of all of the host's connecting wireless links. As the MH battery exhibits a much larger MTBF, than the wireless links, its failure impact will be considered later. Let $MH_k \rightarrow MH_l$ define a checkpointing *relationship* between $MH_k$ and $MH_l$, functioning as the *consumer* and *provider* of checkpointing services, respectively. The set of all such directed relationships, on a MoG instance, is called a *checkpointing arrangement*. We define *stability*, in the context of a checkpointing arrangement, as one where *no consumer or provider prefers another provider or consumer, respectively, to its current partner in the relationship, and all consumers and providers have found providers and consumers, respectively*. Such an arrangement is called *stable*. Distinct arrangements result in differing MoG system reliability values and potential instabilities due to positional and wireless signal strength variations among MHs. Less reliable wireless links are more prone to frequent and intermittent disconnections. With MHs, relative locations, velocities, intervening obstacles, multipathing, interference, and other effects all partially determine link strength and reliability [38]. Our ***Re**liability **D**riven* methodology, is made QoS-aware, via wireless measurements, of the reliabilities of links between MHs in the MoG. It aims to enhance the service quality (QoS) received by a distributed application while executing on the MoG. It makes use of link reliability values to dynamically maintain superior checkpointing arrangements in the mobile environment.

### 3.1 Theoretical Considerations

Laptops, wearable computers, PDAs, cell phones, and other MHs may exhibit high mobility. Practically, it is desirable for our MoG to be flexible, ad-hoc, efficient, adaptive, and robust. In a mobile environment, intermittent signal degradation, interferences, and outright signal loss are major sources of both temporary and long term disconnection of certain MHs from the MoG. Battery powered and energy-limited, with low processing and memory capacity, MHs highly desire conservation of energy, CPU cycles, and memory. In this environment, any algorithm used for checkpointing arrangement must quickly converge to a superior solution. As no nearby BS is guaranteed, the MoG of interest calls for a decentralized checkpointing methodology, storing checkpoints at neighboring MHs.

Since the QoS delivered by the MoG is sensitive to the checkpointing methodology employed, it is essential to decide which MHs send checkpointed data to what other MHs for safe storage (*checkpointing arrangement*). When establishing the arrangement, it is intuitive for MHs with stronger wireless connections to other MHs to serve as the more desirable points of stable storage for checkpointed data. Also, those MHs, which are weakly connected, and possibly moving out of range (imminent failure), urgently need access to strongly connected peers to quickly store checkpointed data, preventing its irrevocable loss. MH battery failure can be considered similarly. To automatically and dynamically determine which checkpointing arrangements are more reliable and stable at a given snapshot time, $t$, each MH is assumed to be aware of the quality of its own wireless connections as well as the wireless connections of its neighbors (i.e., QoS-aware). Considering the aforementioned MoG operational requirements, we list below, the formulation upon which our desirable, QoS-aware, decentralized, checkpointing arrangement methodology is derived.

Problem formulation. Let $C_k$ represent the *consumer* function of Host $k$, transmitting checkpointed data to be handled by the *provider* function, $P_l$, of Host $l$. Further, let $S_j$ represent the *recovery execution* function at Host $j$, scheduled to resume execution of the checkpointed process affected by the potential failure of Host $k$. To ensure efficient handoff and resumption of the process on $S_j$, it is reasonable to assume that $j$ will be a neighbor of $l$. Let $M_i$ define an instance of a connected MoG at time $t_i$, composed of $N \geq 2$ MHs, under the worst case where all MHs are desired to collaborate via some wireless paths in order to execute a client's distributed application. In essence, $M_i$ is a connected graph with MHs as vertices and wireless links as edges. Let $\Delta t_{cp}$ and $\Delta t_{ps}$ be the minimum transmission times required to communicate checkpointed data from $C_k$ to $P_l$ prior to failure at $k$, and to recover checkpointed data from $P_l$ to $S_j$ after the failure at $k$, respectively. In both instances, hosts communicating the checkpointed data are neighbors, giving rise to $\Delta t_{cp} = \Delta t_{ps}$. Thus, we define a short time interval, $\Delta t_i = \Delta t_{cp} = \Delta t_{ps}$, centered on $t_i$, during which any wireless link must operate failure-free in order to communicate checkpoint data with the minimum delay. We designate $\phi_i$ as the mapped, link failure rate (for a given link) assessed at $t_i$, and reasonably assumed to be constant over $\Delta t_i$. Next, we define the term, "$\underline{\text{connectivity,}}$" $\psi_k$, of a host, $MH_k$, (functioning as *consumer* or *provider*) to be the parallel reliability of wireless links to all neighbors in $M_i$, i.e., $\psi_k = (\lambda_{k1,}||\lambda_{k2,}||\lambda_{kl,}||\cdots||\lambda_{kn,})$, $k \neq l$, where $\lambda_{kl}$ is the link *reliability* over interval $\Delta t_i$ from $MH_k$ to some neighbor $MH_l$ given as $\lambda_{kl} = e^{-\phi i \Delta t_i}$. Thus, $\psi_k > \psi_g$

implies that $MH_k$ is better connected on $M_i$ than is $MH_g$. *Connectivity* expresses the probability that the referenced host will never entirely lose all its wireless links to the MoG for the entire duration, $\Delta t_i$.

Given a worst case, well connected, MoG instance, $M_i$, at time snapshot $t_i$, consisting of $n$ MHs, the number, $\xi$, of complete checkpoint arrangements grows factorially in N. ReD maximizes

$$R_i = \prod_{k=0}^{N-1} [1 - (1 - \gamma_k)(1 - \rho_l \lambda_{kl})], \quad k \neq l, \tag{1}$$

where $\gamma_k$ is the connectivity of $MH_k$ functioning as the *consumer*, $\rho_l$ is the *connectivity* of $MH_l$ functioning as the *provider*, $\lambda_{kl}$ is the *reliability* of the wireless link connecting hosts $k$ and $l$ in $M_i$, $N$ is the total number of hosts in $M_i$ and $R_i$ is the arrangement reliability on the MoG instance. With feasible hardware support, $\gamma_k$, $\rho_l$, and $\lambda_{kl}$ can reasonably be assumed to be independent and generally uncorrelated. In essence, $R_i$ is the probability of the entire MoG operating successfully over time interval, $\Delta t_i$. By maximizing $R_i$ to achieve $R_i \max$, we maximize the probability that either no consumer fails (i.e., becomes disconnected from the MoG) during $\Delta t_i$, or if it does fail, the MoG will always succeed in access to the needed most current checkpoint, by one or more links, because the provider holding the checkpoint remains connected to the MoG.

On $M_i$, let $\Pi_i$ be a directed graph, $\Pi_i = G = (V, E)$, representing a *unique checkpointing arrangement*. It is a snapshot of all *consumer* $\rightarrow$ *provider* relationships on $M_i$ at time $t_i$. Each $MH_k$ is chosen from the set of vertices, as $V_k$ and each wireless link, $\lambda_{kl}$, is chosen from the set of edges, i.e., $E_{kl}$. Given $\Pi_i$, our problem initially appears to be that of finding the optimum checkpoint arrangement, $\Pi_i \max$, on $M_i$, yielding the greatest reliability, $R_i \max$. This problem, $\Pi_i \max$, seeks some checkpoint arrangement or arrangements, all yielding a reliability value, $R_i \max$, i.e., an optimal singularity. However, such a solution cannot be practically realized due to intrinsic measurement uncertainties in hardware, software, and the environment. In utilizing real MH network devices to measure link strengths so as to establish and propagate host *connectivities* in the MoG, we must allow for measurement uncertainties. We utilize $\mu_\tau$ to allow for temporal differences between the sampling time, $t_1$, and arrangement decision time, $t_2$, and it is important since data sampled at one host is often acted upon a little later by some other host in building checkpoint arrangements. Any device signal measurement inaccuracies, in the wireless environment, are accounted for by $\mu_\delta$. Finally, we consider $\mu_\sigma$, since special position and orientation contribute to measurement differences among hosts. The total, ever present, uncertainty, $\mu_T$, from all sources is the statistical combination of $\mu_\tau$, $\mu_\delta$, and $\mu_\sigma$ (see Table 1 for symbol explanation). Hence, while there are likely to be additional sources of uncertainty (e.g., interference and measurement, reliability assessment mapping, and temporal differences), the salient point here is that we must allow for this uncertainty in our calculations.

Allowance for $\mu_T$ requires that we revise our definition of "maximum" or "optimal." If we assume some small average $\mu_T$ at each MH, the uncertainty introduced into the calculation of the "optimal" arrangement will depend statistically upon $\mu_T$ and the number of MHs in the MoG instance. Thus, some uncertainty is introduced into any possible determination of the "optimal" arrangement. *Therefore, under such conditions, no optimal singularity can ever be determined.* Instead, the best we can hope for is that we can determine whether or not, under any instance, $M_i$, there exists a checkpoint arrangement, $\Pi_i$, whose calculated reliability, $R_i$, is within some desired range of the best possible pairing optimal, $\Pi_i \max$, assuming all measurements were precise. In other words we want to test whether $R_{th} \leq R_i \leq R_i \max$ is a true statement (where $R_{th}$ is the general lower bound threshold selected to allow for average uncertainty and to require arrangement *stability*, and $R_i \max$ is the theoretical upper bound maximum). A *stable* arrangement is defined as one where no consumer in $\Pi_i$, can improve arrangement reliability, from its perspective, by successfully pairing with a better-connected provider in $M_i$, or equally, one where no provider in $\Pi_i$ can improve arrangement reliability from its perspective, by successfully pairing with a more needy consumer in $M_i$. An unstable arrangement is likely to reorganize in a short order in favor

TABLE 1
Important Symbols

| Item | Quantity |
|---|---|
| $MH_k$ | Mobile Host, $k$ |
| $M_i$ | Instance, snapshot of a connected MoG at time, $t_i$ |
| $\Delta t_{cp}$ | Minimum checkpoint data transmission time $C_k$ to $P_l$ |
| $\Delta t_{ps}$ | Minimum recovery data transmission time $P_l$ to $S_j$ |
| $\Delta t_i$ | Min. checkpointed data transmission interval, during which $\phi_i$ is assumed constant $\Delta t_i = \Delta t_{cp} = \Delta t_{ps}$ |
| $\phi_i$ | Mapped, link failure rate, assessed at $t_i$ can be assumed constant over $\Delta t_i$. |
| $\lambda_{kl}$ | Wireless Link *Reliability*, $MH_k$ to $MH_l$ $\lambda_{kl} = e^{-\phi_i \Delta t_i}$ |
| $\psi_k$ | Connectivity of Host $k$; $\psi_k = (\lambda_{k1}, \Vert \lambda_{k2}, \Vert \lambda_{kl}, \Vert \cdots \Vert \lambda_{kn},)$, $k \neq l$ |
| $R_i$ | Reliability of checkpointing arrangement at time, $t_i$ $R_i = \prod_{k-0}^{N-1} [1-(1-\gamma_k)(1 - \rho_l \lambda_{kl})]$, $k \neq l$ |
| $\gamma_k$ | Connectivity of host, $k$, functioning as consumer |
| $\rho_l$ | Connectivity of host $l$, functioning as provider |
| $C_k$ | Consumer function of $MH_k$ |
| $P_l$ | Provider function of $MH_l$ |
| $S_j$ | Recovery execution function of $MH_j$ |
| $\Pi_i$ | Some unique checkpointing arrangement. A directed graph consisting of a unique collection of $C_k \rightarrow P_l$ relationships on $M_i$. $k \neq l$ |
| $R_i \max$ | Optimum checkpointing arrangement reliability |
| $\Pi_i \max$ | Checkpointing arrangement(s) on $M_i$, yielding $R_i \max$ |
| $R_{th}$ | General stable, lower bound reliability threshold, suboptimal, selected to allow for uncertainty, $\mu_T$. |
| $\mu_T$ | Statistical combination of $\mu_\tau$, $\mu_\delta$, $\mu_\sigma$ |
| $\mu_\tau$ | Temporal Sampling Uncertainty |
| $\mu_\delta$ | Device Sampling Uncertainty |
| $\mu_\sigma$ | Spacial Sampling Uncertainty |
| $\Pi_{ithresh}$ | Decision problem, where we ask is arrangement stable and is $R_i \geq R_{th}$? (yes/no) |
| $\Delta G_{ka}$ | Pairing gain of consumer $k$ on prospective provider, $a$ |
| H | Host Scheduling Level: In an $N$ host MoG, $H \leq N$ is the number of MHs scheduled to process executable portions of a client's job. |

of more stable pairings. The practical introduction of naturally occurring uncertainty and the requirement for stability simplifies our problem to that of a decision problem, where the decision is: "have we found a stable checkpointing arrangement, $\Pi_i$ whose calculated value holds the relationship, $R_i \geq R_{th}$ (yes/no)? This problem is called $\Pi_{ithresh}$.

In seeking to find a stable solution to the problem $\Pi_{ithresh}$, we note that it bears some similarities to the well-known NP-complete decision problem: "Stable Marriage Problem with Incomplete Lists and Ties, i.e., SMPILT [43]." We show that SMPILT can be transformed in a polynomial way into $\Pi_{ithresh}$. Theorem. $\Pi_{ithresh}$ *is NP-Complete*. See Appendix for proof.

## 3.2 ReD's Heuristic Basis

In light of the NP-complete search associated with the problem of finding a stable, decentralized, checkpointing arrangement on the MoG, efficient global algorithms for determining $\Pi_{ithresh}$ are not known to exist, and even for small numbers of hosts, the number of enumerations required are factorial in $N$ and prohibitive. So our research focused on the development of a distributed and practical heuristic algorithm for determining decentralized checkpointing arrangements dynamically on a highly mobile MoG platform. ReD seeks to determine the best possible checkpointing arrangement to maximize the probability of application (job) recovery without experiencing an unrecoverable failure ($R_i \max$).

ReD's algorithm takes into account desired behavioral controlling heuristics in the following ways. First, we require the MoG to be capable of autonomous operation without an access point or BS and further to reduce the use of relatively unreliable wireless links. ReD ensures this by storing checkpointed data only at neighboring MHs, within the MoG, and not requiring BS access or checkpoint transmission over multiple hops. Second, in a MoG, dynamicity ensures that a checkpointing arrangement must be converged rapidly and efficiently, even though it may only be close to optimal. While it is true that poor checkpointing arrangements play a role in reducing the $R_i$, we seek to maximize, so too do unconverged arrangements (i.e., arrangements where a significant percentage of consumers are still seeking to establish checkpointing relationships with providers). To ensure convergence within a reasonable time, ReD employs four strategies:

1. ReD is supported by a clustering algorithm, which partitions the global MoG into clusters, allowing ReD to quickly and concurrently find superior arrangements within each cluster instead of having to labor toward a global MoG solution. While many clustering algorithms have been proposed for general ad-hoc networks [28], [29], [30], [31], a simple clustering algorithm is devised and adopted both in our simulator and in our working testbed as a functional support layer for ReD,
2. ReD makes decisions about whether to request, accept, or break checkpointing relationships, locally (at the MH level) and in a fully distributed manner, instead of attempting a high-level centralized or global consensus,
3. ReD keeps checkpoint transmissions local, i.e., neighbor to neighbor, not requiring multiple hops and significant additional transmission overhead to achieve checkpointing relationships, and
4. ReD allows a given consumer or provider to break its existing checkpointing relationship (when a provider breaks a checkpointing relationship, a *break message* is transmitted to the consumer) only when the arrangement reliability improvement is significant, thus promoting *stability*.

Thus, in comparing stability versus reliability gains, we seek a tradeoff to improve the average attained $R_i$. Third, because each MH *consumer* transmits its own checkpoint independently and concurrently with all others in the MoG, our first approach focused on a reasonable design, involving both simple checkpointing and tracking methods, and fewer recovery coordinating messages, made possible since we need only to track and contact one host in order to recover the stored checkpoint. Other checkpointing arrangement methods can be conceived, e.g., one consumer to multiple providers, multiple consumers to one provider, etc. While these are indeed possible, they entail a degree of added complexity for tracking and recovery and are considered outside the scope of this paper. ReD's philosophy is thus to have one *consumer* checkpoint to one and only one neighboring MH *provider*, and for a *provider* to accept a checkpoint from one and only one neighboring MH *consumer*, resulting in a simple, yet decentralized, memory and energy efficient process. As a compromise, ReD often arrives at close-to-optimal checkpoint arrangements (rather than globally optimal ones), trading off potential arrangement reliability perfection for speed in arrangement formation. So, depending upon the network configuration and $N$ (the total number of hosts in the MoG), not all solutions are *perfect (i.e., not all solutions can ensure that all consumers find providers and vice versa)*.

In essence, ReD utilizes certain underpinnings of the widely known Gale-Shapley algorithm with some enhancements designed and tailored for operation in a dynamic mobile environment [45]. Our ReD enhances the Gale-Shapley algorithm to arrive at highly superior and stable system reliabilities, while minimizing convergence time, the number of transmissions, and the memory footprint encountered by the average host.

## 3.3 ReD's Methodology

An executing host is considered to be in "failure," if wireless connections to all of its neighbors are disrupted temporarily or permanently, resulting in its isolation and inability to achieve timely delivery of intermediate or final application results to other hosts. Executing MHs with poor *connectivity*, have greater likelihood of experiencing *failure* than do those with greater *connectivity* and are thus in greater need of checkpointing to the best, most reliably connected providers. In order to evaluate and compare the strength of progressive checkpointing arrangements, we calculate the reliability, $R_i$, of the whole arrangement on the MoG structure ($M_i$), as depicted in Fig. 1, where each of the symbol labels on the model's reliability diagram blocks are as previously defined (see Table 1). Link signal strength

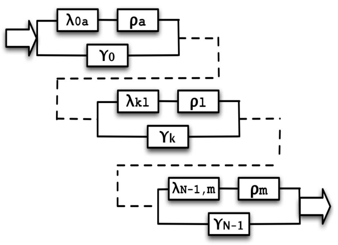

Fig. 1. ReD reliability model.

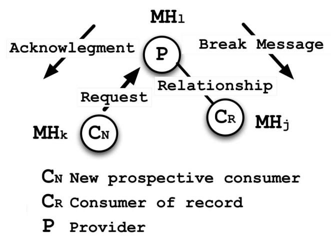

Fig. 2. Protocol function.

decreases inversely with the *square* of the distance between linked hosts. Reliability mapping for the link is thus based on this assumed signal strength profile with failure rate, $\phi_i$, assumed to be constant for the small time interval, $\Delta t_i$, typically a few milliseconds in the mobile environment.

We conclude from Fig. 1 that no unrecoverable failure occurs under the following two cases:

*Case 1:* No failure at participating hosts (i.e., no isolation of any host when its results are required). A given application will run to completion without restarting or rewinding. This is true whether or not checkpointing is done. *Case 2:* Failure of a participating host (isolation). The application will run to completion, with recovery assistance, provided that upon a host failure, its latest saved checkpoint data exists on the provider host, which is not itself isolated and can then be accessed by the MoG during the recovery process. This is possible only if 1) the link for sending checkpoints to the provider host did not fail when being used to checkpoint, and 2) the provider host did not fail (i.e., did not itself become isolated from the rest of the MoG) during the attempt by the MoG to access saved checkpointed data upon recovery.

Our checkpoint arrangement protocol, ReD, makes use of the underpinnings in both cases, resulting in the basis model Fig. 1 depicts. As long as all hosts running the distributed application have high connectivity (i.e., low likelihood of separation from the MoG), robust safe storage or reliable transmission to safe storage is not really needed. Only poorly connected hosts (e.g., hosts on the fringes of the MoG) need robust safe storage and reliable transmission to that safe storage. Furthermore, since host failures happen most frequently to poorly connected hosts, it makes sense to reserve the most robust checkpoint storage (providers) for these hosts, while leaving the least robust storage for the best-connected hosts. This way puts valuable and scarce robust checkpointing resources to work where they are most needed in reducing unrecoverable failures effectively. Essentially then, ReD continually seeks to 1) match providers having the greatest connectivity to consumers having the lowest connectivity and 2) to send checkpointed data from the consumer to the provider over the most reliable link possible in the process as host connectivities and link reliabilities vary in a dynamic and mobile MoG. Based upon Fig. 1, we assume that a consumer can checkpoint to any provider where there are $N$ choices for picking a consumer, $k$, and $N-1$ choices for picking a provider, $l$, once the initial consumer is selected. ReD thus seeks to maximize (1), as previously defined. Because of its NP-completeness, the global optimization of this model on the dynamic MoG is not deemed feasible. Even in small clusters, e.g., 10 hosts or so, we did not consider a centralized approach for optimizing $R_i$ to be practical. In addition to optimizing $R_i$, two precepts considered nearly as important in the design of ReD were convergence rate and *stability*. It is the average series of $R_i$ values obtained during the course of execution that ultimately determines the QoS level presented to a user application. In a mobile environment, where checkpointing arrangements are constantly forming and being torn down, both rapid convergence to superior checkpointing arrangements and *stability* once those arrangements are reached, are considered to be equally important as the $R_i$ values converged upon. ReD's heuristic method ensures that checkpointing arrangement decisions are made locally and individually at the host level, promoting rapid convergence, while a threshold mechanism is included in order to provide stability control.

## 3.4 ReD's Algorithm Description

Upon initiation or refresh, if some consumer, $MH_k$, does not have a designated provider, it begins to look for one. In doing so, it examines and compares $\lambda_{ka} \times \rho_a$ products of each of its $n$ neighbors ($a = \exists\ neighbor\ ID$), $MH_a$ (where $\rho_a = \psi_a$), sorting the products in decreasing order (max to min). Next, it transmits a checkpoint request, first to the list top host (having the greatest $\lambda_{ka} \times \rho_a$ product), e.g., $MH_l$. In essence, a checkpoint request asks the provider, $MH_l$'s permission to send checkpointed data to it. If prospective provider, $MH_l$ has no consumer of record, it readily grants permission and sends a positive acknowledgment back to $MH_k$, establishing a $MH_k \rightarrow MH_l$ relationship. On the other hand, if a relationship, say, $MH_j \rightarrow MH_l$ already exists, $MH_l$ checks to see if the requesting consumer's *pairing reliability gain* is greater than that for its existing paired consumer, i.e., is $(\psi_k || \lambda_{kl} \times \rho_l)\ /\ \psi_k alone > (\psi_j || \lambda_{jl} \times \rho_l)\ /\ \psi_j alone$ a true statement? If so, it breaks its relationship with $MH_j$ by sending it a break message, and then grants permission to $MH_k$ by sending it an acknowledgment. If, on the other hand, the statement proves false, $MH_k$ is sent a negative acknowledgment, and $MH_l$ maintains the relationship, $MH_j \rightarrow MH_l$, unless otherwise severed due to mobility, or weak signal. Fig. 2 depicts ReD's protocol messages while the pseudocode used is listed in the Appendix. We call this apparent gain in reliability, achieved by the prospective pairing of some consumer, $MH_k$, with some provider, $MH_a$, the *pairing gain* of $k$ on $a$, i.e., $\Delta G_{ka}$, or $G(MH_k \rightarrow MH_a)$, where $\Delta G_{ka} = (\psi_k || \lambda_{ka} \times \rho_a)\ /\ \psi_k alone$. ReD has been enhanced since our initial preliminary work [49] to include pairing gain considerations when comparing prospective relationships as opposed to just strictly comparing alternative relationship reliabilities directly. In summary, ReD, essentially greedily, attempts to globally maximize $R_i$, through local, decentralized, $MH_k \rightarrow MH_a$ pairing decisions. To allow for mobility, tables of connectivity and link reliabilities are updated and aged via the softstate process. A consumer periodically refreshes, checks its $\lambda_{ka} \times \rho_a$ sorted product list, and determines if it might do better to find another provider, whereupon it takes action. A provider, upon loss of relationship with a consumer, for any reason, deletes its consumer pointer and admits requests from other consumers. Finally, upon receiving a break message, a paired consumer initiates the process of finding a checkpoint provider all over again. Note that ReD is designed to be **IID** (meaning to run on each host *i*ndependently and *id*entically). Because messages can be lost in transmission, especially over poor wireless links, tables of host connectivity, link reliabilities, and *consumer* and *provider pointers* are maintained at hosts by the soft state registration process, e.g., due to mobility or a weak signal, a host may declare that it has lost its provider (or consumer), attempting to find a new one (or ready to admit a checkpoint request).

## 4 Simulations, Testbed, and Results

Sections 4.1 and 4.2 describe the simulator and the testbed environments, respectively, utilized to independently evaluate ReD. A series of experiments were conducted in both environments with exhaustive studies being done on the simulator and subsequent studies done in the testbed placing hosts in various relative positions to each other and averaging the results.

### 4.1 Simulator Description

The simulator is a custom designed, layered environment, including a *Physical Layer* (wireless 801.11b employing 64 byte frames) , *Link State Layer* (to calculate link reliability), *Connectivity Layer*, (using link state reliability calculations to compute host connectivity), and the *Clustering Layer* (to establish and maintain clusters). The simulated checkpoint arrangement protocol, either ReD or a compared *R*andom *C*heckpoint *A*rrangement (RCA) protocol, then operates at the Checkpoint Arrangement Layer within each of the formed clusters. RCA is not QoS-aware as is ReD, and was designed to determine consumer $\rightarrow$ provider relationships within clusters according to randomly assigned relative strengths instead of actual host connectivity values. So while having access to QoS-aware system information, i.e., link reliabilities and host connectivities, provided by the lower link-state and connectivity layers, RCA effectively chooses to ignore these when making its checkpointing arrangement decisions, rendering it QoS-blind. So, a consumer seeking checkpointing services arbitrarily chooses a provider, while a provider uses an arbitrary method to determine whether to provide services to a requesting consumer, e.g., greatest Node ID (or randomly assigned strength). Otherwise, RCA utilizes the same supporting layer functions, checkpointing arrangement mechanisms, messages, constraints, and algorithm as ReD, in order to preserve a fair comparison. It is reasonable to compare ReD to RCA since QoS-aware decentralized checkpointing arrangement in MoGs is an emerging area of research with little comparative, or competing algorithms in existence in the literature. Thus, RCA was designed to be QoS-blind, not a lower bound, permitting a reasonable comparison, so that ReD's improvement over any generic *arbitrary arrangement choice* methodology could be clearly demonstrated. The simulator gathers statistics via the *Measurement Module* about the performance of ReD and RCA, allowing us to adjust host density, velocity, and signal range. Gathered data included checkpointing arrangement reliability, and message counts along with average numbers of consumer hosts finding providers.

### 4.2 Testbed Implementation

In order to augment and further validate the simulator results, we implemented an actual testbed, of from six (6) to twelve (12) identically configured laptop computers to fairly compare ReD versus RCA in a $10 \times 10$ m indoor field. Numerous (10 to 24 hour) tests were run for both ReD and RCA in identical positional configuration permutations. Each computer, via *iwspy*, and Orinoco Silver 802.11b PCMCIA cards, measured received 802.11b wireless signal strength, smoothed them via moving average, and then indexed them by neighbor MAC address (ID). Hosts then mapped each measurement to respective link-reliability figures, storing them in their dynamic link-reliability arrays. Mapping calculations were based upon actual field tests of signal strength host-to-host in various positions and angular orientation with respect to each other. Ethernet signal strength measurements obtained corresponded well to those found in other work utilizing the same measurements to facilitate robot locations [33].

Utilizing this data, each host calculated its own connectivity to the rest of the MoG testbed. Hosts dynamically exchanged calculated connectivities with neighbors via short UDP/IP packets, allowing connectivity tables, indexed by neighbor ID, and sorted max to min, to facilitate both cluster formation and ReD's checkpointing arrangement decisions. Performance data for ReD versus RCA, were obtained with received signal strength being found to vary both positionally, and temporally.

### 4.3 Evaluation Results

Over 1,600, 240 hour, simulations, with 32 hosts, walking randomly (0.5 m/s) in a 100-m square area, were conducted, to obtain statistically broad and valid data. Simulation utilized the BigRed computer cluster (consisting of 180 nodes), located in the Center for Advanced Computer Studies, at the University of Louisiana at Lafayette. First, the performance of ReD was compared to that of RCA in a series of simulations under the identically configured conditions. Simulation data clearly showed ReD's significant improvement over RCA. Subsequently, we verified ReD's performance superiority over RCA through our working testbed. Finally, we hypothesized an intuitive and practical stabilization mechanism for ReD, utilizing pairing reliability gain threshold (i.e., we require a certain minimum threshold of pairing reliability gain before permitting established consumer $\rightarrow$ provider relationships to be revised) could significantly improve $R_i$ values. Simulations validate our hypothesis.

#### *4.3.1 Simulation Results*

Initial simulations compared ReD to RCA to establish MoG-system checkpointing arrangement reliabilities, $R_i$s, (temporal averages) attained with respect to host scheduling

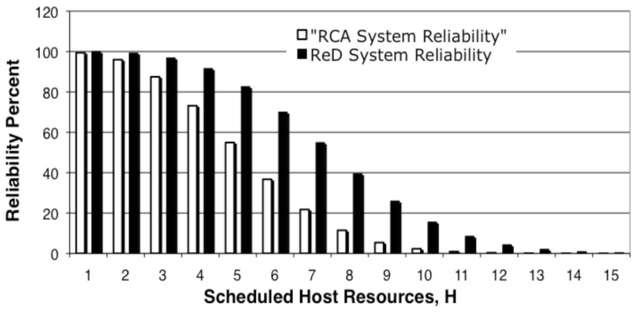

Fig. 3. ReD versus RCA system reliability under wireless range 20 m.

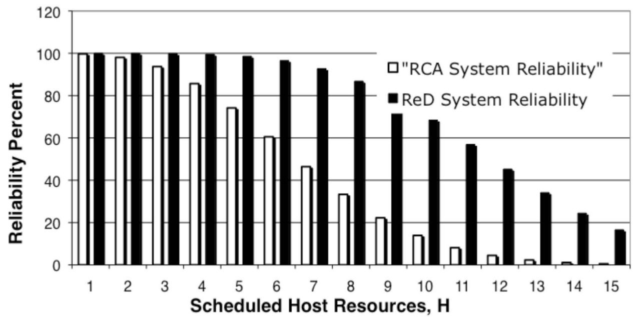

Fig. 4. ReD versus RCA system reliability under wireless range 30 m.

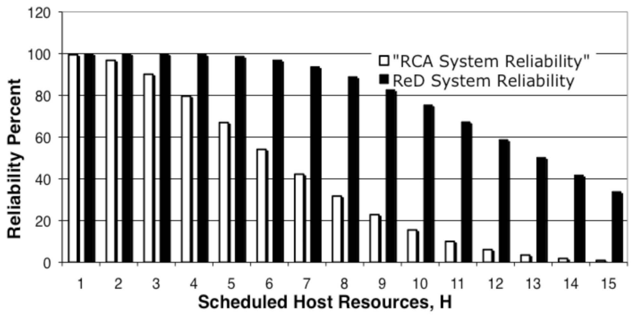

Fig. 5. ReD versus RCA system reliability under wireless range 40 m.

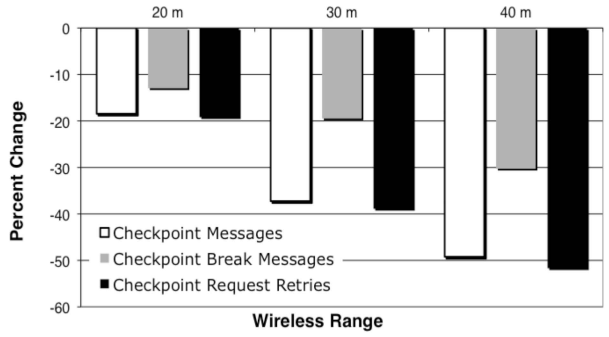

Fig. 6. ReD versus RCA in checkpoint message percent change.

levels, H (refer to Table 1), ranging from 1 to 15, over wireless ranges of 20, 30, and 40 m, as shown in Figs. 3, 4, and 5, respectively. Fig. 6 shows checkpointing arrangement message-count percent reductions for ReD versus RCA. Simulation results show ReD to enjoy significant improvements over RCA in the following respects:

1. the average achieved MoG checkpointing arrangement reliabilities, $R_i$s at most H levels.
2. $R_i$ scaling with H.
3. ReD's percent reduction of checkpoint message transmissions, showing far better scaling with wireless range increases than RCA.
4. ReD enjoyed 12.9 percent, 19.5 percent, and 30.3 percent fewer break message transmissions than did RCA. And
5. improvements of 43 percent, 74 percent, and 106 percent in the average number of maintained consumer $\rightarrow$ provider relationships for wireless ranges of 20, 30, and 40 m, respectively, an independent indicator of superior stability (not shown in graphs).

Fig. 5 shows ReD's arrangement reliability figure at 59 percent versus RCA's at 6 percent for a scheduling level of $\mathrm{H} = 12$ scheduled hosts, within a 32 host MoG system. This figure gives the probability, $p$, of executing a distributed job on the arrangement for an entire checkpoint interval (e.g., 2 seconds) without sustaining a single unrecoverable failure. Conversely, on average, unrecoverable failures occur during job execution with probability $1 - p$. Therefore, unrecoverable failures will on an average, be much more significant in the background under RCA than under ReD. Every unrecoverable failure imposes significant job rewind delay during application execution. These rewind delay differences are magnified by the degree to which distributed job portions are data dependent (and not embarrassingly parallel). The adverse impact of these reliability figures means that lengthy rewind delay to a proposed job's maximum execution time will be significantly greater under RCA control. Thus, the MoG job scheduling function will be able to commit to a much earlier job completion time under ReD. If part of the client's QoS specifies some maximum (e.g., near real time) tolerable delay, and prospective execution time under RCA exceeds that value frequently, a significant number of jobs would not be successfully completed under RCA. ReD can provide reliability versus H level to other MoG functions such as the MoG job apportionment scheduler. Note that at lower H-levels, the reliability of the arrangement subset is better since we can choose the most reliable consumer $\rightarrow$ provider pairs from the arrangement to form our execution group. So, practically, the job scheduler would utilize projected real time reliability figures to determine at what H levels, some portions of the jobs could be run to complete within the specified maximum delay tolerance (i.e., QoS).

Delineations between ReD and RCA are statistically significant exceeding 95 percent confidence intervals. Further, previous simulations of 64 and 81 host systems have shown, that as the number of hosts and host density increase, ReD's advantage over RCA consistently scales better, with even greater superiority in both average MoG-system reliabilities, i.e., average of $R_i$s, and reduced checkpointing arrangement message count.

To understand the root causes underlying the results obtained, consider that with increases in range, the number of hosts, and host density within the MoG, greater degrees of freedom with respect to $MH_k \rightarrow MH_a$ pairing choices, exist. ReD, being QoS-Aware is better able to capitalize upon this increased freedom degree, weighing connectivity metrics to

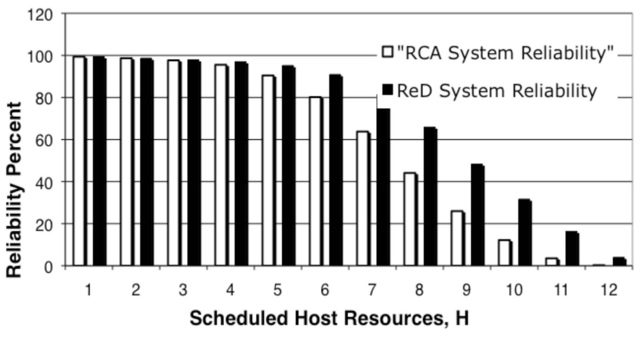

Fig. 7. Comparison of ReD versus RCA testbed system reliabilities.

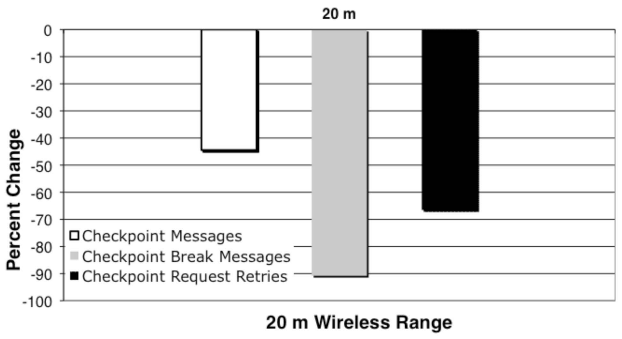

Fig. 8. ReD versus RCA in testbed message percent change.

increase $R_i$s by surgically implementing relationships with the greatest pairing reliability gain instead of arbitrary ones.

To further understand results, however, we need delve into the concept of *stability* as earlier defined, in the context of MoG checkpointing arrangements. In the presence of mobility, the associated relocation of hosts, and changes in their resultant link reliabilities and connectivities, the established MoG checkpointing arrangement must be dynamically and continually revised in order to flexibly track optimal system reliability. Too stable (i.e., rigid) and we can't track mobility in order to seek optimal stability, not stable enough and we risk a large degree of arrangement reshuffling, which also reduces reliability (i.e., reliability tracking versus stability tradeoff). ReD, in making a greater percentage of correct $MH_k \rightarrow MH_a$ pairing choices than RCA, ensures less need to break (and thus revise) existing relationships in order to form new ones. This important characteristic is demonstrated by ReD's overall reduction in the number of break messages (Fig. 6). In the process of continually revising arrangements some consumers will inevitably be unpaired (in any system snapshot), and in the process of finding a provider to pair with. Upon obtaining a provider, these unpaired consumers sometimes cause that provider to break with its existing consumer, causing that consumer to likewise seek a provider, etc., with a resulting cascading reshuffling of the system checkpointing arrangement. Reshuffling has a negative affect on average system reliability due to the increased message traffic and a certain background percentage of unpaired consumers. ReD, thus outperforms RCA due to both increased $R_i$s obtained and improved system stability. In summary, because RCA, is not QoS-aware, it tends to produce arbitrary checkpointing arrangements due to arbitrary $MH_k \rightarrow MH_a$ pairing choices with greater likelihood of connectivity loss under mobility, causing instability and suboptimal checkpointing arrangements. The superior performance shown by ReD can be attributed to both its resulting superior checkpointing arrangements and its superior stability characteristics. Due to its superior stability characteristics, ReD requires significantly fewer checkpointing arrangement maintenance messages as depicted in Fig. 6, with a decrease of more than 50 percent in the number of consumer initiated checkpoint repeat requests at 40 m effective wireless range. As expected, this advantage also scales better with increased range and has been shown previously to scale better with increased numbers of hosts and host density in simulations with up to 81 hosts.

### 4.3.2 Testbed Results

Results obtained from testbed implementation corroborate our simulation studies. Fig. 7 depicts the results showing ReD's superior system reliability as compared to RCA. Because, simulations were able to utilize a large number of hosts, and full mobility, a situation not afforded to the testbed implementation, ReD's superior performance results were more salient in the simulator than those depicted in Fig. 7. However, we nevertheless see improvement by approximately 200 percent corresponding with a scheduling level, H, of 10 MHs in a 12-host system. The testbed was run indoors, for numerous 24 hours periods, under a number of static positional permutations with results averaged to obtain the Fig. 7 results. Signal strength was found to vary both temporally and positionally.

Analogous to the results obtained in simulations, and in addition to its superior system reliability achievement, ReD requires 44.4 percent fewer checkpointing arrangement and maintenance messages on average than did RCA in the testbed implementation. As an indicator of superior stability, ReD enjoyed a break message decrease of more than 90 percent as depicted in Fig. 8.

### 4.3.3 ReD's Stability Control—Simulation Results

We hypothesized that by controlling ReD's system stability, we might achieve an enhanced level of average system reliability or control over same, through reductions in relationship reshuffling, trading off lower optimal arrangement reliabilities to achieve better stability. The premise is that smaller time intervals with unpaired consumers at the expense of optimal arrangement reliability achieved will actually improve average arrangement reliability. The mechanism we tested operates at the crucial juncture in a provider's decision to offer checkpointing services to a perspective consumer. As stated in ReD's earlier specified methodology, if a prospective provider $MH_l$, does not have a relationship with an existing consumer, $MH_j$, it simply agrees to pair with the requesting consumer, $MH_k$, sending it an acknowledgment. However, if such a relationship does exist, provider, $MH_l$, checks first to see if *pairing gain* of the proposed relationship is greater than that of the existing relationship by some threshold multiple factor, $F$ (i.e., $\Delta G_{kl} > F \times \Delta G_{jl}$). If the comparison holds true, provider, $MH_l$, dissolves the existing $MH_j \rightarrow MH_l$ relationship in favor of the prospective $MH_k \rightarrow MH_l$ relationship, sending $MH_j$ a break message and $MH_k$ an acknowledgment. Increased stability manifests itself by increases in average system reliability and fewer average numbers of break messages being transmitted.

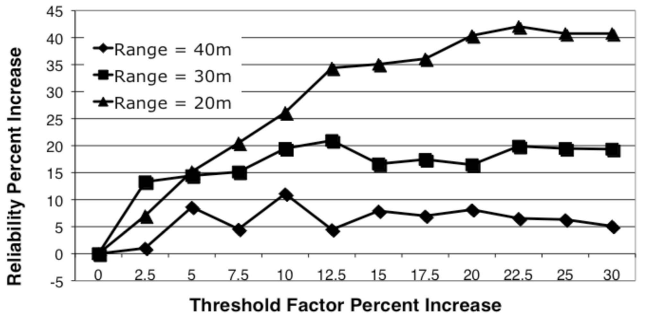

Fig. 9. Reliability percentage versus threshold factor percent increase under ReD.

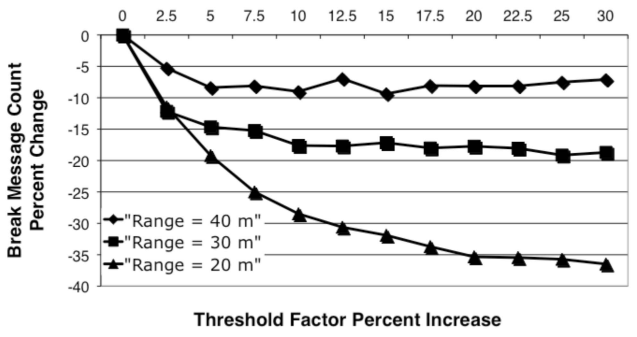

Fig. 10. Break message count percentage change versus threshold factor percent increase under ReD.

Fig. 9 shows simulation results from our implementation of this stability control mechanism for a 32-host system with wireless ranges of 20, 30, and 40 m, respectively, and a host scheduling level, H, of 15. The increases in system reliability shown are all normalized percentiles based on ReD's performance baseline with $F = 1$. Thus the horizontal axis, *threshold factor percent increase*, equates to $(F - 1) \times 100\%$, while the vertical axis average system reliability increase, equates to $(R_F - R_1)/R_1 \times 100\%$, where $R_1$ is ReD's achieved system reliability with $F = 1$ (i.e., no threshold increase), and $R_F$ is ReD's percent reliability increase with $F > 1$. We see that as $F$ increases, system reliabilities are shown to increase significantly, up to as much as 40 percent in the case of the curve based on 20 m wireless range. Fig. 10 plots $(F - 1) \times 100\%$ on the horizontal axis and $(B_F - B_1)/B_1 \times 100\%$ where $B_1$ and $B_F$ are the break message counts with $F = 1$, and $F > 1$.

From these figures, we see that greater increases in system reliability, and as an independent verification, greater decreases in the break message count, occur with decreases in wireless range. Relationships are in general, dissolved and reformed with decreasing frequency, as $F$ increases, making overall arrangements more stable. This is expected, since at smaller wireless ranges, the neighborhood density is less, and unpaired prospective consumers are less likely to cause relationship changes and any associated arrangement reshuffling, since the pool of neighboring consumer and provider alternatives is smaller (smaller neighborhood), producing more sensitive and salient increases in stability and the resultant average system reliability seen. However, at greater wireless ranges, with a larger neighborhood, more alternatives exist for a consumer seeking a provider. Consumers thus have broadened their search and are more successful at finding situations, where $\Delta G_{ka} > F \times \Delta G_{ja}$ is true, causing relationship changes and less dramatic increases in stability and system reliability with increases in $F$. Thus, our stability control enhancement to ReD, produces the best payoff at lower wireless ranges, resulting in greater average reliability attainment and correspondingly fewer break messages. This result is useful and practical, since under ReD, each host can determine its neighborhood density, indirectly through message exchange with neighbors, allowing it to moderate $F$, and thus yield effective stability control. Such moderation control decisions could be made individually, or in concert with neighbors.

The reason we don't arbitrarily increase $F$ is because of the reliability tracking versus stability tradeoff, identified earlier. High mobility environments necessitate some moderation in $F$ so that the system can flexibly adjust arrangements sufficiently to more optimally track reliability as host positions and network conditions change. This is born out by the flattening of the curves shown in the figures indicating a point of diminishing returns upon further increases in $F$. Therefore, both average host mobility and average provider density are inputs to the process of modulating $F$ and thus stability versus reliability control. Using ReD's stability control mechanism, hosts could theoretically snoop broadcasts of general wireless traffic to monitor the level of break message activity, gather data about neighborhood density and mobility through exchange with neighbors about connectivity, and thus modulate $F$ in order to effectively and responsively control stability versus arrangement reliability.

## 5 Concluding Remarks

Nodal mobility in a large MoG may render a MH participating in one job execution, unreachable from the remaining MHs occasionally, calling for efficient checkpointing in support of long job execution. As earlier proposed checkpointing approaches cannot be applied directly to MoGs and are not QoS-aware, we have dealt with QoS-aware checkpointing and recovery specifically for MoGs, with this paper focusing solely on checkpointing arrangement. It has been demonstrated via simulation and actual testbed studies, that ReD achieves significant reliability gains by quickly and efficiently determining checkpointing arrangements for most MHs in a MoG. ReD is shown to outperform its RCA counterpart in terms of the average reliability metric and does so with fewer required messages and superior stability (which is crucial to the checkpoint arrangement, minimization of latency, and wireless bandwidth utilization). Because ReD was tailored for a relatively unreliable wireless mobile environment, its design achieves its checkpoint arrangement functions in a lightweight, distributed manner, while maintaining both low memory and transmission energy footprints.

This work has marked implications for resource scheduling, checkpoint interval control, and application QoS level negotiation. It fills a novel niche component of the ever-developing field of MoG middleware, by proposing and demonstrating how QoS-aware functionality can be practically and efficiently added.

## Appendix

## NP-Complete Proof

**Theorem 1.** $\Pi_{ithresh}$ *is NP-Complete.*

The following steps constitute our proof:

**Proof.**

1. SMPILT utilizes two finite equal-sized sets of players referred to as men and women. Each man, $m_i$, ranks women $(1 \leq i \leq n)$ forming his preference list. Similarly, each woman, $w_i$, ranks men $(1 \leq i \leq n)$ forming her preference list. Any man or woman's list can contain preference "ties" among members of the opposite sex. Any man or woman's list can in fact be an "incomplete" listing of the members of the opposite sex. In other words, $N_{List} \leq n$. A stable marriage is one where no man or woman prefers another woman or man, respectively, to his or her current partner in marriage, and all members of each sex have found a spouse. In such a case a solution exists to SMPILT. In the context of $\Pi_{ithresh}$, stability means that no consumer or provider in the arrangement can realize a greater reliability increase (pairing increase from consumer perspective) by pairing with another provider or consumer, respectively, all consumers have found providers and all providers have consumers.
2. **Lemma 1**. *SMPILT has been shown to be NP-complete* [43].
3. Given a certificate with a checkpointing arrangement on $\Pi_i$, we can, in polynomial time, determine if we do in fact have a solution to $\Pi_{ithresh}$. Therefore $\Pi_{ithresh}$ is in NP.
4. Transformation of SMPILT into $\Pi_{ithresh}$. Note that each MH can in fact be both a consumer and a provider. Given SMPILT, we map each man into a specific consumer and each woman into a specific provider. In $\Pi_{ithresh}$, each MH, i.e., each consumer and provider, can calculate pairing gain for each provider and consumer alternative, respectively, allowing preference lists to be established. SMPILT's preference lists for both sexes directly maps into this feature of $\Pi_{ithresh}$. Since no consumer or provider can have global knowledge of all providers and consumers, respectively, and, since we do not allow a consumer host to checkpoint to a provider on the same host, the "incomplete list allowance" of SMPILT directly maps into $\Pi_{ithresh}$. Due to the uncertainty $\mu_T$, identified previously, the "ties" allowance of SMPILT directly maps into $\Pi_{ithresh}$ allowing us to model uncertainty about perspective provider and consumer host alternatives. The stability requirement of SMPILT directly maps into $\Pi_{ithresh}$ in which we also require stability to promote rapid convergence. All transformations, included above, require no more than polynomial time to complete.
5. **Theorem [44].** *Let X be an NP-complete problem. Consider a decision problem Z, where $Z \in NP$, such that X is polynomially (Turing) reducible to Z. Then Z is also NP-complete.*
6. By Lemma 1, SMPILT is NP-complete. According to Item 3, $\Pi_{ithresh} \in NP$. Further, $SMPILT \leq_T^p \Pi_{ithresh}$, hence $\Pi_{ithresh}$ is NP-complete. This completes the proof. □

*ReD's Pseudocode:*

*Consumer Code:*

```
ReD basic pseudocode listing:
Node: while not refresh, as consumer, $\exists C_k$
  Sort neighbor host providers, $\forall P_a$ on list by $\lambda_{ka} \times \rho_a$
(max to min); send checkpointing request to list top,
e.g., $MH_l$;
  if(no reply after five tries $||MH_l$ sends NACK
  $||(MH_k \rightarrow \exists P_a$ && $P_a$ sends Break to this $C_k$)), then
  if((list of $\lambda_a \times \rho_a)! = \emptyset$), then
      send checkpointing request to next $P_a$ on list; (repeat)
    else -- sleep until next refresh period; // empty
  else if ACK from selected $P_a$
    then set toPointer $\rightarrow$ to selected $P_a$;
upon refresh: start consumer process anew
```

*Provider Code:*

```
Node: as a Provider, $\exists P_l$
if(receive checkpointing request from $\exists C_k$) Then
  if ($C_j \rightarrow P_l == \emptyset$) // i.e., if a relationship does not
  already exist
    then send ACK to requesting $C_k$; set fromPointer to$C_k$;
  else if($C_j \rightarrow P_l ! = \emptyset$), then if($\Delta G_{kl}$ requesting >
  $\Delta G_{jl}$ existing)
      send ACK to requesting $C_k$; // ack requesting
      consumer
      send Break to existing $C_j$; // break with existing
      consumer
      set fromPointer to requesting $C_k$;
  else send NACK to requesting $C_k$;
Repeat for each checkpointing request received
```

## Acknowledgments

This work is supported in part by the US Department of Energy (DOE) under Award Number DE-FG02-04ER46136, by the Board of Regents, State of Louisiana, under Contract Number DOE/LEQSF(2004-07)-ULL, and by the US National Science Foundation under Grant CCF-0916451.

## References

[1] SUN Microsystems, "Sun Grid Compute Utility," http://www.sun.com/service/sungrid, 2006.

[2] Hewlett-Packard Development Company, L.P., "Grid-Computing—Extending the Boundaries of Distributed IT," http://h71028.www7.hp.com/ERC/downloads/4AA03675ENW.pdf?jumpid=reg_R1002_USEN, Jan. 2007.

[3] "IBM Grid Computing," http://www-1.ibm.com/grid/about_grid/what_is.shtml, Jan. 2007.

[4] S. Wesner et al., "Mobile Collaborative Business Grids—A Short Overview of the Akogrimo Project," white paper, Akogrimo Consortium, 2006.

[5] Computerworld, "HP Promises Global Wireless for Notebook PCs," http://www.computerworld.com/mobiletopics/mobile/story/0,10801,110218,00.html?source=NLT_AM&nid=110218, Apr. 2006.

[6] J. Long, W. Fuchs, and J. Abraham, "Compiler-Assisted Static Checkpoint Insertion," *Proc. Symp. Fault-Tolerant Computing,* pp. 58-65, July 1992.

[7] K. Ssu, B. Yao, and W. Fuchs, "An Adaptive Checkpointing Protocol to Bound Recovery Time with Message Logging," *Proc. 18th Symp. Reliable Distributed Systems,* pp. 244-252, Oct. 1999.

[8] N. Neves and W. Fuchs, "Coordinated Checkpointing without Direct Coordination," *Proc. Int'l Computer Performance and Dependability Symp.,* pp. 23-31, Sept. 1998.

[9] W. Gao, M. Chen, and T. Nanya, "A Faster Checkpointing and Recovery Algorithm with a Hierarchical Storage Approach," *Proc. Eighth Int'l Conf. High-Performance Computing in Asia-Pacific Region,* pp. 398-402, Nov. 2005.

[10] R. de Camargo, F. Kon, and A. Goldman, "Portable Checkpointing and Communications for BSP Applications on Dynamic Heterogenous Grid Environments," *Proc. Int'l Symp. Computer Architecture and High Performance Computing,* pp. 226-234, Oct. 2005.

[11] L. Wang et al., "Modeling Coordinated Checkpointing for Large-Scale Supercomputers," *Proc. Int'l Conf. Dependable Systems and Networks,* pp. 812-821, July 2005.

[12] A. Agbaria and W. Sanders, "Application-Driven Coordination-Free Distributed Checkpointing," *Proc. 25th IEEE Conf. Distributed Computing Systems,* pp. 177-186, June 2005.

[13] A. Oliner, R. Sahoo, J. Moreira, and M. Gupta, "Performance Implications of Periodic Checkpointing on Large-Scale Cluster Systems," *Proc. 19th IEEE Int'l Conf. Parallel and Distributed Processing Symp.,* Apr. 2005.

[14] C. Lin, S. Kuo, and Y. Huang, "A Checkpointing Tool for Palm Operating System," *Proc. Int'l Conf. Dependable Systems and Networks,* pp. 71-76, July 2001.

[15] D. Pradhan, P. Krishna, and N. Vaidya, "Recoverable Mobile Environment: Design and Trade-Off Analysis," *Proc. Symp. Fault-Tolerant Computing,* pp. 16-25, June 1996.

[16] T. Park, I. Byun, H. Kim, and H. Yeom, "The Performance of Checkpointing and Replication Schemes for Fault Tolerant Mobile Agent Systems," *Proc. 21st IEEE Symp. Reliable Distrinuted Systems,* pp. 256-261, Oct. 2002.

[17] H. Higaki and M. Takizawa, "Checkpoint-Recovery Protocol for Reliable Mobile Systems," *Proc. 17th IEEE Symp. Reliable Distributed Systems,* pp. 93-99, Oct. 1998.

[18] C. Ou, K. Ssu, and H. Jiau, "Connecting Network Partitions with Location-Assisted Forwarding Nodes in Mobile Ad Hoc Environments," *Proc. 10th IEEE Pacific Rim Int'l Symp. Dependable Computing,* pp. 239-247, Mar. 2004.

[19] K. Ssu et al., "Adaptive Checkpointing with Storage Management for Mobile Environments," *IEEE Trans. Reliability,* vol. 48, no. 4, pp. 315-324, Dec. 1999.

[20] G. Cao and M. Singhal, "Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 2, pp. 157-172, Feb. 2001.

[21] A. Acharya and B. Badrinath, "Checkpointing Distributed Applications on Mobile Computers," *Proc. Third Int'l Conf. Parallel and Distributed Information Systems,* pp. 73-80, 1994.

[22] R. Prakash and M. Singhal, "Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 10, pp. 1035-1048, Oct. 1996.

[23] N. Neves and W. Fuchs, "Adaptive Recovery for Mobile Environments," *Comm. ACM,* vol. 40, no. 1, pp. 68-84, 1997.

[24] G. Cao and M. Singhal, "On the Impossiblity of Min-Process Non-Blocking Checkpointing and an Efficient Checkpointing Algorithm for Mobile Computing Systems," *Proc. Int'l Conf. Parallel Processing,* pp. 37-44, 1998.

[25] G. Cao and M. Singhal, "Low-Cost Checkpointing with Mutable Checkpoints in Mobile Computing Systems," *Proc. 18th Int'l Conf. Distributed Computing Systems,* pp. 464-471, 1998.

[26] B. Yao, K. Ssu, and W. Fuchs, "Message Logging in Mobile Computing," *Proc. IEEE Symp. Fault-Tolerant Computing,* pp. 294-301, 1999.

[27] B. Yao and W.K. Fuchs, "Proxy-Based Recovery for Applications on Wireless Hand-Held Devices," *Proc. 19th IEEE Symp. Reliable Distributed Systems,* pp. 2-10, Oct. 2000.

[28] M. Chaterjee, S. Das, and D. Turgut, "WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks," *Cluster Computing,* vol. 5, no. 2, pp. 193-204, Apr. 2002.

[29] S. Bandyopadhyay and E. Coyle, "An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks," *Proc. IEEE INFOCOM,* pp. 1713-1723, Mar./Apr. 2003.

[30] T. Kanungo et al., "An Efficient K-Means Clustering Algorithm: Analysis and Implementation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 7, pp. 881-892, July 2002.

[31] U. Brandes, M. Gaertler, and D. Wagner, *Experiments on Graph Clustering Algorithms,* pp. 568-579. Springer, 2003.

[32] S. Cook, "The Complexity of Theorem-Proving Procedures," *Proc. Third Ann. ACM Symp. Theory of Computing,* pp. 151-158, 1971.

[33] A. Howard, S. Siddiqi, and G. Sukhatme, "An Experimental Study of Localization Using Wireless Ethernet," *Proc. Fourth Int'l Conf. Field and Service Robotics,* pp. 2-10, July 2003.

[34] P. Ghosh, N. Roy, S. Das, and K. Basu, "A Game Theory Based Pricing Strategy for Job Allocation in Mobile Grids," *Proc. 18th Int'l Parallel and Distributed Processing Symp.,* pp. 82-91, Apr. 2004.

[35] S. Wong and K. Ng, "A Middleware Framework for Secure Mobile Grid Services," *Proc. Sixth IEEE Int'l Symp. Cluster Computing and Grid Workshops (CCGRIDW '06),* pp. 2-12, Nov. 2006.

[36] The K*Grid Project, http://www.gridcenter.or.kr/MobileGrid/index.php, 2010.

[37] The Akogrimo Project, http://www.akogrimo.org, 2010.

[38] P. Gummadi, D. Wetherall, B. Geenstein, and S. Seshan, "Understanding and Mitigating the Impact of RF Interference on 802.11 Networks," *Proc. SIGCOMM,* Aug. 2007.

[39] Disaster Handling and Crisis Management—Akogrimio DHCM, http://www.akogrimo.org/modules.php?name=AddFile&file=addfile&datei=scen_crisismgmt, 2008.

[40] "Bufferfly and IBM to Introduce First Computing Grid for Video Game Industry," http://www.hoise.com/primeaur/02/articles/weekly/AE-PR 06-02-23.html, 2008.

[41] A. Hampshire, "Extending the Open Grid Services Architecture to Imtermittently Available Wireless Networks," *UKeScience All Hands,* 2004.

[42] Integrated Project on Pervasive Gaming, http://iperg.sics.se, 2010.

[43] K. Iwama, D. Manlove, S. Miyazaki, and Y. Morita, "Stable Marriage with Ties and Incomplete Lists," *Proc. 26th Int'l Colloquium of Automata, Languages and Programming (ICALP '99),* pp. 443-452, July 1999.

[44] G. Brassard and P. Bratey, *Fundamentals of Algorithmics,* p. 451. Prentice Hall, 1995.

[45] D. Gusfield and R. Irving, *The Stable Marriage Problem, Structure and Algorithms.* The MIT Press, 1989.

[46] T. Phan, L. Huang, and C. Dulan, "Challenge: Integrating Mobile Wireless Devices into the Computational Grid," *Proc. ACM MOBICOM,* pp. 271-278, Sept. 2002.

[47] M. Messig and A. Goscinski, "Autonomic System Management in Mobile Grid Environments," *Proc. Australasian Symp. Grid Computing and Research (AUSGrid),* 2007.

[48] X. Ren, R. Eigenmann, and S. Bagchi, "Failure-Aware Checkpointing in Fine-Grained Cycle Sharing Systems," *Proc. 16th IEEE Int'l Symp. High Performance Distributed Computing (HPDC-16),* pp. 33-42, June 2007.

[49] P. Darby and N. Tzeng, "Peer-to-Peer Checkpointing Arrangement for Mobile Grid Computing Systems," *Proc. 16th IEEE Int'l Symp. High Performance Distributed Computing (HPDC-16),* June 2007.

[50] S. Song, H. Youn, and U. Kim, "A New Reflective and Reliable Context-Oriented Event Service Architecture for Pervasive Computing," *Proc. Int'l Conf. Computational Science and Its Applications (ICCSA-5),* pp. 139-148, May 2006.

[51] J. Lee, R. Mateo, B. Gerardo, and S. Go, "Location-Aware Agent Using Data Mining for the Distributed Location-Based Services," *Proc. Int'l Conf. Computational Science and Its Applications (ICCSA-5),* pp. 867-876, May 2006.

[52] "HP and Cingular Wireless Introduce First Global Broadband Notebook PC in U.S.," HP Press Release, http://www.hp.com/hpinfo/newsroom/press/2006/061211a.html?jumpid=reg_R1002_USEN, Dec. 2006.

[53] W. Woerndl and R. Eigner, "Collaborative, Context-Aware Applications for Inter-Networked Cars," *Proc. 16th Int'l Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '07),* pp. 180-185, June 2007.

[54] A. Chandler and J. Finney, "Rendezvous: Supporting Real-Time Collaborative Mobile Gaming in High Latency Environments," *Proc. ACM SIGCHI Int'l Conf. Advances in Computer Entertainment Technology,* pp. 310-313, 2005.

[55] W. O'Neil, "The Cooperative Engagement Capability (CEC) Transforming Naval Anti-Air Warfare," Case Studies in National Security Transformation, Center for Technology and National Security Policy, Number 11, Aug. 2007.

[56] P PEO Soldier, "Documentary Highlights Success of Land Warrior System," http://www.army.mil/-news/2009/05/27/21680-documentary-highlights-success-of-land-warrior-system, May 2009.

**Paul J. Darby III** received the BS degree in electrical engineering and the MS degree in telecommunications from the University of Southwestern Louisiana (now University of Louisiana at Lafayette) in 1979 and 1995, respectively. He is working toward the PhD degree in the Center for Advanced Computer Studies. He was actively engaged in both the aerospace and telecommunications industries for more than 20 years and a recipient of NASA's Silver Snoopy Award. In telecommunications, he served as Engineer Advanced for the State of Louisiana in the Department of Public Safety and the Office of Telecommunications Management including LaNet. He has been a registered professional engineer since 1989. Currently, he serves as an instructor of electrical and computer engineering in the Department of Electrical and Computer Engineering at the University of Louisiana at Lafayette. His current research interests include mobile wireless grid computing with a focus on checkpointing. He is a member of the IEEE.

**Nian-Feng Tzeng** received the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 1986. Currently, he is Lockheed Martin professor with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, where he joined in 1987. His current research interests include computer communications and networks, high-performance computer systems, and parallel and distributed processing. He was on the editorial board of the *IEEE Transactions on Computers*, 1994-1998, and on the editorial board of the *IEEE Transactions on Parallel and Distributed Systems*, 1998-2001, and was the chair of Technical Committee on Distributed Processing, the IEEE Computer Society, from 1999 to 2002. He served as the program chair of 10th IEEE International Conference on Parallel and Distributed Systems (ICPADS), July 2004, and has been on the program committees of various conferences. He is the recipient of the outstanding paper award of the 10th International Conference on Distributed Computing Systems, May 1990, and received the University Foundation Distinguished Professor Award in 1997. He is a fellow member of the IEEE.