

Supporting Self-Organization for Hybrid Grid Resource Scheduling

Değer Cenk Erdil
Department of Computer Science
Binghamton University (SUNY)
Binghamton, NY, 13902-6000, U.S.A.
erdil@cs.binghamton.edu

Michael J. Lewis
Department of Computer Science
Binghamton University (SUNY)
Binghamton, NY, 13902-6000, U.S.A.
mlewis@cs.binghamton.edu

ABSTRACT

Increasing scale, dynamism, and complexity of hybrid grids make traditional grid resource scheduling approaches difficult. In such grids, where resource volatility and dynamism is common, self-organization is a key technique for autonomous grid nodes to follow basic rules to minimize human participation, and administrative bottlenecks. This paper presents experimental results with a framework for distributed grid resource scheduling. In particular, we study information dissemination, which distributes information about dynamic grid resource states to remote schedulers. The framework helps each autonomous grid node self-organize by (1) self-configuring its operational parameters based on dynamic grid characteristics, and (2) self-adjusting its dissemination behavior by taking feedback from the system. The framework also helps distributed grid schedulers to find the tradeoff between two important performance parameters: dissemination overhead, and query satisfaction rates. We show by simulation that autonomous grid nodes that self-organize into small groups, and compare their local state to the states of other peer nodes, can perform comparable to both (1) similar dissemination protocols that are statically configured for each specific case, and (2) a theoretical central metascheduler that operates on complete knowledge of available resource and offered load states.

1. INTRODUCTION

Existing grids can be classified as *institutional* and *desktop* grids. Current institutional grids such as TeraGrid [2] and Open Science Grid (OSG) [1] impose administrative overhead by requiring resources to be centrally managed and coordinated. For example, resource scheduling in such grids focuses on mapping requests to resources, using all necessary information at a central location, such as the metascheduler.

Desktop grids [20, 17] and public resource computing

This research is supported by NSF Career Award ACI-0133838 and NSF Award CNS-0454298.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'08 March 16-20, 2008, Fortaleza, Ceará, Brazil
Copyright 2008 ACM 978-1-59593-753-7/08/0003 ...\$5.00.

systems [3, 6] scavenge unused computing cycles from idle desktop computers. They lower grid participation barriers to the levels of peer-to-peer systems, and thus increase the number of potential grid nodes from tens or hundreds of clusters, to thousands or millions of individual machines.

Future “hybrid” grids will include resources with characteristics of both desktop and institutional grid components. The increasing scale, dynamism, and complexity of hybrid grids make traditional grid resource scheduling approaches difficult, and suggest self-organizing alternatives.

One of the requirements for self-organization in hybrid grids is to minimize human participation such that software systems self-organize their activities, and impose minimal, decentralized control over participating autonomous agents [22]. Thus, hybrid grids require new approaches for self-organization. For example, each node may adaptively self-configure its operational parameters based on dynamic characteristics of its environment. Moreover, each node may self-adjust its behavior by taking some form of feedback from the system.

Earlier we proposed a scoring and adaptation framework for autonomous grid nodes to evaluate how well the scheduling policies of a particular grid system match the local policies of each autonomous grid node, in terms of resource and load characteristics, such as available resources, network topologies, and system saturation levels. We first showed that adaptive information dissemination protocols show promise in reducing the dissemination overhead compared to statically configured dissemination protocols [14]. Moreover, we also showed that adaptive protocols can find the balance between dissemination overhead and query satisfaction rates under dynamic resource and load characteristics in large-scale grids [13].

In this paper, we extend our work on the self-organizing aspects of autonomous grid nodes that use our scoring and adaptation framework. By simulation, we study the overhead characteristics of self-organization alternatives. We also compare the query satisfaction rates, and average distances of non-adaptive and adaptive dissemination protocols to those of a theoretical central metascheduler with complete knowledge of available resources and load on the grid.

This scoring and adaptation framework presents an enabling mechanism for self-organization in hybrid grids: each grid node takes basic adaptation steps based on local and neighborhood score comparisons, and resulting grid resource scheduling behavior is comparable to the performance of a hypothetical central metascheduler that (1) collects all available resource and offered load information at a central location, and (2) affects heuristically best sched-

ules.

2. RELATED WORK

Information dissemination is a technique to publish available resource states at remote locations. Distributed grid schedulers can then use this information to match resources with requests. Information dissemination can be structured or unstructured. Structured solutions impose some form of organization for grid nodes (e.g., hierarchical), and they provide better control of dissemination coverage and overhead. However, in most cases they are not scalable. Unstructured solutions on the other hand, have flat organization, where all grid nodes are peers. Unstructured solutions are better for large systems, such as grids. Based on coverage characteristics, information dissemination techniques can also be categorized as uniform and non-uniform. Uniform coverage is easier with structured approaches.

One example of structured information dissemination is Condor flocking [11], where a Condor pool sends resource information directly to all other pools, to form explicit relationships. Legion [4] and Globus [7] also use structured and uniform information dissemination. In both systems, nodes proactively publish, and periodically update, information about available resources at distributed information repositories. Yalagandula and Dahlin [23] propose Scalable Distributed Information Management Systems (SDIMS), for information dissemination, using distributed hash tables, and aggregation.

Epidemic protocols are common unstructured information dissemination techniques. Kempe et al. present theoretical results for gossiping protocols based on resource location and communication delay [18]. Haas et al. propose ad-hoc broadcast [21] to cover an entire network with probabilistic forwarding. Kermarrec et. al. [19] propose gossiping protocols where each node maintains a partial view of the membership. A multicast source disseminates a message that each infected node forwards to a random node set of size $O(\log(n))$. The message reaches all members with high probability after $O(\log(n))$ rounds.

Another epidemic protocol style is pairwise gossiping [8], where nodes randomly select other grid nodes to exchange limited subsets of information about potential resource providers. Drost et al. use rumor-mongering to robustly increase coverage of gossiping protocols in real grid settings [10], which is better suited for uniform dissemination coverage. For large-scale and dynamic grids with volatile resource states, we previously proposed non-uniform pairwise gossiping protocols [12], where each disseminating node covers a region of the grid, as opposed to entire grid. This alternative approach reduces overhead drastically while achieving comparable query satisfaction rates.

3. IMPLEMENTATION

This section describes our implementation to simulate the grid scheduling problem, and a corresponding solver that mimics a centralized metascheduler. Nodes may join and leave the grid at any time. Each node interacts as a resource provider, a resource requester, or both. Grid resource scheduling is distributed, i.e. each node has a scheduler component, and a query matching module that matches local queries with available resources, using information about remote providers that previously disseminate their available resource states.

3.1 Resource matching

Providers disseminate available resource state information on the grid, and each grid node keeps a subset of the disseminated information in its local information repository, which forms the potential providers list. Upon a local resource request, the query matching component sorts the list of potential providers by employing resource ranking criteria [9], and then iterates through the list, sending a *reservation request* to providers, in turn, until receiving a positive *reservation reply*.

Upon receiving the reservation request, a provider checks to see whether it has sufficient resources to satisfy the query, and responds accordingly. A positive reservation reply packet marks a successful schedule. A negative reservation reply causes the requester to reconsider the potential providers list, and contact the next potential provider. This process continues until the requester exhausts its list of potential providers. If all the potential providers respond negatively, the query is deemed unsatisfied. Figure 1 gives an overview of this process.

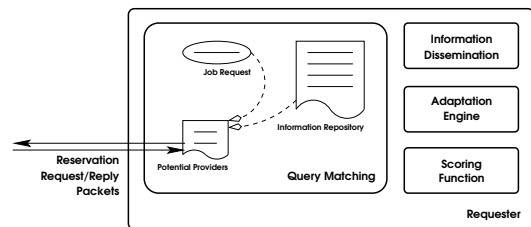


Figure 1: Decentralized query matching model in autonomous grid nodes. Upon receiving resource requests, the query matching component sorts the potential providers list, which is cached at a local information repository based on previously disseminated state information packets. The providers on the list are contacted in sorted order until either a provider confirms resource availability (successful schedule) or the list is exhausted (unsuccessful schedule).

3.2 Neighborhoods and self-organization

Nodes self-organize into neighborhoods: groups of nodes that share common interests, such as those that provide the same resource type. When a node joins the grid, it operates individually for a predefined period of time. During this period, it receives several marker packets (i.e., invitations to join a neighborhood) from nearby neighborhood leaders. If the node desires to join a neighborhood, it sends a response to the neighborhood leader to start the neighborhood membership process.

Alternatively, a node may form its own neighborhood by sending marker packets of its own. If it cannot attract the minimum number of members, n_{min} , after a predefined period of time, t , it simply gives up and destroys the neighborhood, releasing any new members it has recently attracted. After reaching the maximum number of neighborhood members, n_{max} , the neighborhood leader continues to send marker packets, but less frequently. Each neighborhood leader inspects marker packets of other similar neighborhoods, and may decide to merge its neighborhood with another neighborhood, by sending a merge request to the other neighborhood leader. A neighborhood leader may also decide to split the neighborhood into two smaller

neighborhoods, by electing a new neighborhood leader for the new neighborhood.

Moreover, neighborhoods can be relaxed or strict. In relaxed neighborhoods, nodes continue their autonomous operation on the grid, but compare their local scores to the neighborhood score to better adjust their behavior. In strict neighborhoods, the neighborhood leader keeps aggregate available resource state, and disseminates the aggregate state accordingly. Strict neighborhoods are most useful for dissemination proxies [15]. Figure 2 shows relaxed and strict neighborhoods.

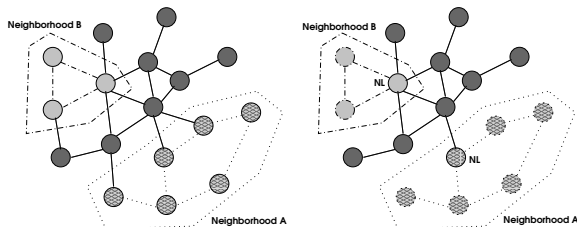


Figure 2: Relaxed and strict neighborhoods on the grid. In relaxed neighborhoods, nodes continue their autonomous operation on the grid, but compare their local scores to their neighborhood scores. Neighborhoods A and B have multiple disseminating links to other grid nodes. In strict neighborhoods, nodes elect a neighborhood leader, which then keeps aggregate available resource states, and disseminates the aggregate states accordingly.

3.3 Scoring and adaptation

One main property of the autonomous nodes is that they can *adapt* to dynamic grid conditions by adjusting their information dissemination behavior. This capability in turn, will yield better grid scheduling performance in terms of lower overhead, higher query satisfaction rates, or both. To adjust their behavior based on dynamic grid resource characteristics, nodes use two additional components: adaptation, and scoring.

Setting the level of dissemination to appropriate values is a difficult task for autonomous nodes, when the outside environment is dynamic. The main scoring intuition is that nodes should have a reference point, or a *metric*, that measures how well resource matching policies, and system resource distribution are meeting the particular local policies and goals of each autonomous node. In other words, a grid node score serves a purpose similar to that of a load index in classical distributed systems scheduling.

To adjust dissemination protocol behavior based on dynamic grid resource states, nodes calculate a local score: an integer number between 1 (lowest) and 100 (highest). A higher score is better. Each node calculates, and periodically updates, a local score, and compares it with an “outside” score. Based on the score comparisons, nodes take actions that are specified in the adaptation policies. Thus, scoring is the main enabling mechanism for adaptation.

Local score is different for providers and requesters. A provider score at the local level incorporates parameters such as local resource utilization, and incoming grid offered load. A requester score at the local level uses either query satisfaction rates, or yield, which is defined as the number of successful resource matches per each reserva-

tion attempt. Neighborhood score is the arithmetic mean of the neighborhood members’ local scores.

Nodes further define score ranges, and three operating regions: *low*, *normal*, and *high*. More details of the scoring algorithm and the adaptation framework are described elsewhere [13].

3.4 Solver study

Another contribution of this research is a higher level modelling of grid resource matching. The grid resource scheduling problem is a specific case of the bi-partite matching problem. Thus, we modified the solver code for Goldberg’s heuristic minimum-cost maximum-flow algorithm [16] for the bi-partite matching problem. We have also written general-purpose wrapper scripts for the solver and run the simulator and the solver in parallel with the same configurations and input parameters. This comparison helps us learn about the upper bounds for performance and thus better optimize our adaptive protocols. In particular, the solver helps us study the resource matching problem with complete scheduling knowledge under different resource characteristics, which shows the best possible (heuristic) answers.

4. RESULTS

Scalable Simulation Network Framework (SSF) [5] simulates the system, Goldberg’s heuristic solver for the minimum-cost maximum-flow bipartite matching problem mimics the central metascheduler, and GT-ITM [24] generates network topologies. We use a 600-node transit-stub topology. All simulations execute for 100 cycles; each data point represents the average of 40 runs with different random seeds. Similarly, the solver runs 100 times, in parallel (i.e. with the same configuration parameters) with the simulator. Since a mapping between a resource and a request may last for multiple cycles (up to n), the solver input files for the next n runs are adjusted accordingly after each run, based on the results of the solver, where n is the maximum duration of provider-requester matches, in simulation cycles.

We vary the ratio of available resources (AR) versus offered load (OL), and study three resource saturation levels:

- *over-saturated*, where $AR = OL/3$,
- *saturated*, where $AR = OL$, and
- *under-saturated*, where $AR = OL * 3$.

We report dissemination overhead in terms of the total number of hops travelled by information packets, query satisfaction rates as the ratio of satisfied queries over the total number of queries generated, and average distances as the number of hops between provider-requester pairs in terms of the number of hops on the overlay network.

4.1 Neighborhoods and Packet Overhead

This section compares the dissemination overhead of relaxed and strict neighborhoods explained in Section 3.2 in two different resource and load configurations, and three different resource saturation levels.

The first test places a total of 40 grid nodes into four neighborhoods, in groups of ten, where each neighborhood represents a different resource class (e.g., different CPU types). Thus, each grid node is preconfigured to join the neighborhood for its resource type, with a known neighborhood leader. Figure 3 shows that relaxed and strict neighborhoods have comparable dissemination overhead. Both

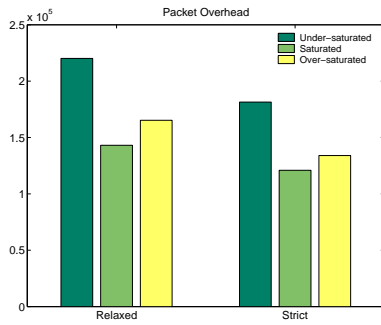


Figure 3: Packet overhead of relaxed and strict neighborhoods for over-saturated, saturated, and under-saturated systems. Isolated test cases. 600 total, 40 active grid nodes: 20 providers and 20 requesters self-organize into four neighborhoods of size ten each. Each neighborhood represents a different resource class, and thus each grid node joins the neighborhood for its resource class.

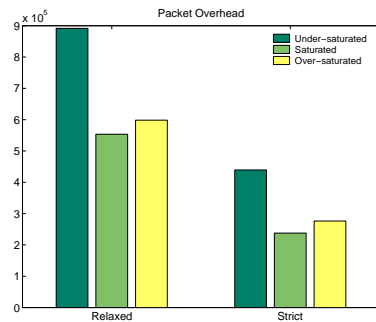


Figure 4: Packet overhead of relaxed and strict neighborhoods for over-saturated, saturated, and under-saturated systems. Random test cases. 600 total, 100 active grid nodes: 50 providers and 50 requesters self-organize into ten neighborhoods of size ten each. All grid nodes and neighborhoods are of the same resource class, and each grid node joins a random neighborhood.

self-organization methods result in the largest amount of overhead when the resource level is under-saturated, where nodes disseminate more aggressively to attract remote requests. This case also results in the biggest difference between the two alternatives.

The second test places a total of 100 grid nodes into ten neighborhoods in groups of ten. In this test, all grid nodes are of the same resource class (e.g., the same CPU type). Thus, each grid node joins a random neighborhood. Figure 4 shows that when nodes self-organize into random neighborhoods, relaxed neighborhoods result in more dissemination overhead compared to strict neighborhoods, due to individual dissemination of state information packets by each grid node. Similar to the previous configuration, the largest amount of overhead occurs in the under-saturated case. The overhead ratios of strict to relaxed neighborhoods in the under-saturated, saturated, and over-saturated cases are: 2.03, 2.33, and 2.17, respectively.

4.2 Dissemination Protocols

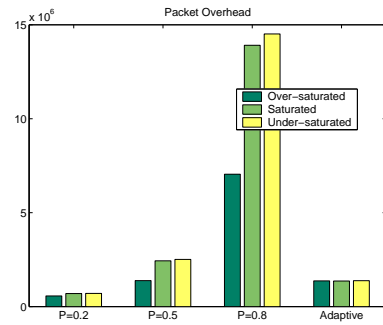


Figure 5: Packet overhead for over-saturated, saturated, and under-saturated systems. 600 total grid nodes: 300 providers and 300 requesters self-organize into 60 neighborhoods of size ten each.

This section compares the adaptive information dissemination protocol performance to (1) three separate dissemination protocols that are statically configured, and (2) a theoretical central metascheduler with complete knowledge of available resources and offered load on the grid. Two types of grid nodes: providers and requesters, self-organize into neighborhoods in the tests. Providers disseminate available resource state information to attract remote requesters. Upon local job requests, query matching components at requesters contact potential resource providers to match available resources with offered load. Results of three different system saturation levels are reported.

Tests report results for three epidemic gossiping protocols that are statically configured with gossiping probabilities of 0.2, 0.5, and 0.8, respectively. We also report results for the adaptive information dissemination protocol that uses the same epidemic gossiping protocol but adjusts its gossiping probability based on the score comparisons as described in Section 3, and for the heuristic solver for the bipartite matching problem.

Figure 5 reports the packet overhead for the three system saturation levels. Since the solver does not have a notion of overhead, it only report results for the epidemic gossiping protocols. Static epidemic protocol with gossiping probability $P = 0.8$ has the highest overhead across all the cases. The adaptive information dissemination protocol generates packet overhead comparable to the static protocol with $P = 0.5$ in the over-saturated system, and approximately half of the packet overhead in the saturated and under-saturated cases. Although the static dissemination protocol with $P = 0.2$ has the lowest packet overhead, its corresponding query satisfaction rates are also lower than other protocols.

Figure 6 reports the query satisfaction rates for the same three system saturation levels. The solver has the highest query satisfaction rates, since it has complete information. The static dissemination protocol with $P = 0.8$ has the next highest query satisfaction rates, but compared to its packet overhead, $P = 0.5$ is a better choice in all three saturation levels. The adaptive information dissemination protocol performs comparable to $P = 0.5$, but results in slightly lower query satisfaction rates. However, each adaptive grid node compares its local score with the neighborhood score, and adjusts its behavior accordingly, *without* previous knowledge of the system that it operates in.

Figure 7 reports the average distances for the same three



Figure 6: Query satisfaction rates (percent) for over-saturated, saturated, and under-saturated systems. 600 total grid nodes: 300 providers and 300 requesters self-organize into 60 neighborhoods of size ten each.

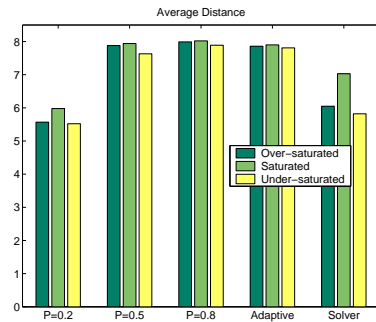


Figure 7: Average distances between any provider-requester pairs for over-saturated, saturated, and under-saturated systems. 600 total grid nodes: 300 providers and 300 requesters self-organize into 60 neighborhoods of size ten each.

system saturation levels. The solver, with complete knowledge, results in the lowest average distances. However, epidemic gossiping protocols also result in average distances that are close to those of the solver.

5. SUMMARY

In this paper, we present further experimental evaluation of a scoring and adaptation framework we previously introduce as an enabling mechanism for self-organization in hybrid grids. We show by simulation that those autonomous nodes that self-organize into neighborhoods, and elect a neighborhood leader to disseminate aggregate information, may reduce dissemination overhead by up to 50% compared to nodes that self-organize into neighborhoods but still disseminate individually. Moreover, we also show that resource provider nodes that disseminate available resource states based on the scoring and adaptation framework presented can help distributed grid schedulers achieve query satisfaction rates, and average distances that are comparable to both (1) similar dissemination protocols that are statically configured, and (2) a theoretical central metascheduler with complete knowledge.

6. REFERENCES

- [1] The open science grid.
<http://www.opensciencegrid.org>.
- [2] TeraGrid Project. <http://www.teragrid.org>.

- [3] D. P. Anderson. Boinc: A system for public-resource computing and storage. In *GRID '04: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pages 4–10, Washington, DC, USA, 2004. IEEE Computer Society.
- [4] S. J. Chapin, D. Katramatos, J. Karpovich, and A. Grimshaw. Resource management in Legion. *Future Generation Computer Systems*, 15(5–6):583–594, 1999.
- [5] J. Cowie, H. Liu, J. Liu, D. Nicol, and A. Ogielski. Towards realistic million-node internet simulations. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, 1999.
- [6] M. Cummings and A. Bazinet. The lattice project. <http://lattice.umiacs.umd.edu>.
- [7] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid Information Services for Distributed Resource Sharing. *Proceedings of the 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, 2001.
- [8] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 1–12, New York, NY, USA, 1987. ACM Press.
- [9] R. Desai, S. Tilak, B. Gandhi, M. J. Lewis, and N. B. Abu-Ghazaleh. Analysis of query matching criteria and resource monitoring for grid application scheduling. *Proceedings of CCGrid2006: IEEE International Symposium on Cluster Computing and the Grid*, 2006.
- [10] N. Drost, E. Ogston, R. V. van Nieuwpoort, and H. E. Bal. Arrg: Real-world gossiping. *Proceedings of the 16th IEEE International Symposium on High Performance Distributed Computing*, 2007.
- [11] D. H. J. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A worldwide flock of condors: load sharing among workstation clusters. Technical Report DUT-TWI-95-130, Delft, The Netherlands, 1995.
- [12] D. C. Erdil and M. J. Lewis. Grid resource scheduling with gossiping protocols. *Proceedings of the 7th IEEE International Conference on Peer-to-Peer Computing*, pages 193–200, Dublin, Ireland, 2007.
- [13] D. C. Erdil, M. J. Lewis, and N. Abu-Ghazaleh. An adaptive algorithm for information dissemination in self-organizing grids. *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing (eScience 2006)*, Amsterdam, the Netherlands, December 4–6, 2006.
- [14] D. C. Erdil, M. J. Lewis, and N. Abu-Ghazaleh. An adaptive approach to information dissemination in self-organizing grids. *Proceedings of the International Conference on Autonomic and Autonomous Systems (ICAS'06)*, Silicon Valley, CA, July 2006.
- [15] D. C. Erdil, M. J. Lewis, and N. B. Abu-Ghazaleh. Proxy-based grid information dissemination. *Proceedings of the Workshop on Large-Scale and*

- Volatile Desktop Grids (PCGrid2007)* (in conjunction with *IPDPS2007*), Long Beach, CA, March 2007.
- [16] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *J. Alg.*, 22(1):1–29, 1997.
- [17] P. Kacsuk, N. Podhorszki, and T. Kiss. Scalable desktop grid system. *Proceedings of Seventh International Meeting on High Performance Computing for Computational Science (VECPAR'06)*, July 2006.
- [18] D. Kempe, J. Kleinberg, and A. Demers. Spatial gossip and resource location protocols. *Annual ACM Symposium on Theory of Computing (STOC)*, 2001.
- [19] A.-M. Kermarrec, L. Massoulie, and A. J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 2003.
- [20] D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. Chien. Characterizing and evaluating desktop grids: An empirical study. *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'04)*, April 2004.
- [21] L. Li, J. Halpern, and Z. Haas. Gossip-based ad hoc routing. *IEEE Infocom*, 2002.
- [22] G. D. Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli. Engineering self-organizing systems: Nature-inspired approaches to software engineering. *Lecture Notes in Artificial Intelligence*, (2977), Berlin, Germany, 2004.
- [23] P. Yalagandula and M. Dahlin. A scalable distributed information management system. *Proceedings of ACM SIGCOMM*, Portland, OR, September 2004.
- [24] E. Zegura and K. Calvert. GT Internetwork Topology Models (GT-ITM). <http://www.cc.gatech.edu/projects/gtitm>.