

Proxy-based Grid Information Dissemination

Değer Cenk Erdil, Michael J. Lewis and Nael B. Abu-Ghazaleh
Department of Computer Science
State University of New York (SUNY) at Binghamton
{erdil, mlewis, nael}@cs.binghamton.edu

Abstract

Resource scheduling in large-scale, volatile desktop grids is challenging because resource state is both dynamic and eclectic. Matching available resources with requests is not always possible with existing approaches. Partial dissemination protocols, such as gossiping, may provide efficient schedules when resource requesters are located near providers that can meet their needs. However, when requesters are distant from available resources, regular information dissemination techniques can waste communication bandwidth with futile messages. Thus, it may be advantageous to attempt to advertise to select remote regions of the grid, without necessarily also going through all intermediate nodes. This paper proposes dissemination proxies to increase coverage footprints and reduce dissemination overhead. We incorporate selecting and adjusting the amount of proxy nodes into an adaptive dissemination algorithm, and show that dissemination proxies are able to reduce dissemination overhead, and handle available resource distribution scenarios where regular information dissemination approaches may not produce efficient protocols. We also report initial results that indicate that randomly selecting nodes to serve as proxies can perform as well as strategies that select seemingly better-qualified proxies.

1 Introduction

Resource matching and scheduling in grid environments is a difficult problem that becomes more challenging in desktop grids, due to the scale, heterogeneity, and volatility of system state. Existing solutions to resource matching and scheduling are not ideal since resource states can be more dynamic, and resources are not necessarily always connected. Unlike deterministic protocols, probabilistic information dissemination protocols do not guarantee finding resources. However, they provide robust, efficient and scalable approaches to resource matching in highly dynamic environments, such as large-scale, volatile desktop grids.

We previously described partial dissemination proto-

cols as alternatives to complete information dissemination [6]. The goal is to make coverage footprints—the sets of nodes that receive information about the state of particular resources—large enough to attract jobs that utilize the resource, but small enough to keep overhead low. Partial dissemination reduces the frequency, scope, and/or resolution of resource state information dissemination relative to broadcasting in full detail.

Depending on the base dissemination protocol, dissemination aggressiveness can be adapted in different ways. Probabilistic algorithms, for example, reduce the frequency and scope (but not the detail). In this model, dissemination aggressiveness is a function of its frequency, and of the forwarding probabilities at the intermediate nodes. The dissemination protocol can also be optimized to avoid disseminating information unnecessarily. For example, *change-sensitive protocols* [6] can filter dissemination packets containing similar, redundant, or unnecessary information.

Aggregation can also be used as a tool for reducing the overhead of dissemination. If specific information about individual nodes or clusters causes too much overhead, then summarized information about all of the nodes at a certain “level” or aggregated region may provide enough information for schedulers to increase the effectiveness of their decisions. Aggregation reduces the scope of dissemination by limiting the number of dissemination sources.

The most natural approach is to make the resource itself be the sole source of dissemination; this leads to coverage footprints with the freshest (and therefore the most accurate) information about a resource being located in that resource’s vicinity. This property is attractive because it can lead to schedulers that select nearby resources for the applications they are mapping onto the grid, which can reduce application startup costs and avoid problems such as flashcrowding [7].

However, when nearby resources cannot satisfy local queries, it may be advantageous to advertise to select remote regions of the grid, without necessarily also going through all intermediate nodes. Therefore, another partial dissemination possibility—one that is the subject of this paper—is to introduce *dissemination proxies* to alter the source of dissemination. A node may decide to target particular remote dissemination proxies, unicast information packets to them,

This research is supported by NSF Award ACI-0133838, DOE Grant DE-FG02-02ER25526, and NSF Award CNS-0454298.
1-4244-0910-1/07/\$20.00 ©2007 IEEE.

and have them distribute that information to their proximities. Adaptation can be built into this strategy by changing the number and location of proxies based either on the success of using them for previous disseminations, or on the intended extent of dissemination, for example.

Gossip-based information dissemination is *flat* by nature. Whereas this unstructured communication has benefits such as scalability, ease of implementation, and low maintenance, it lacks the efficiency of a structured approach. The goal of proxies is to get the benefits of a structured approach without having to pay the price of maintaining that structure. Knowing which nodes to pick as proxies would seem to be a fundamentally important aspect of the proxy approach, but making the knowledge of the right proxy locations available for all grid nodes introduces structure into the system. One question we seek to answer in this paper is whether there in fact *are* better (or best) proxy locations, or whether we may get the benefits of proxies without the cost of a structured approach.

2 Related Work

Condor [13], through ClassAds and its matchmaking framework [11] enables resource management and scheduling in distributed systems. In Condor, resource providers and requesters advertise characteristics and requirements in ClassAds. A matchmaking service finds satisfying matches based on the specified constraints, and then notifies the two parties. The requester then can use other Condor services to contact the provider to have the desired service performed. This mechanism requires ClassAds from both parties to make it to the same location to be compared and eventually matched. Whereas this framework has been successful for clusters of computers and in some inter-cluster configurations, it does not necessarily handle the highly dynamic and eclectic state of nodes in desktop Grids. Hopefully, our approach will eventually extend the set of environments for which Condor’s matchmaking framework effectively matches resource requesters to providers.

Yalagandula and Dahlin [15] propose a flexible system called SDIMS to address the aggregation problem in distributed systems. SDIMS aggregates global information while giving a detailed view of nearby information. Moreover, it enables administratively isolated aggregation, based on a distributed hash table (DHT) tree. Thus, multiple users may get information at different granularities. We use aggregation to reduce information dissemination overhead, whereas SDIMS uses it to dynamically adjust the granularity of information abstraction. To adjust the number of proxies used in our framework as well as the level of abstraction, we may borrow ideas from SDIMS in the future.

An alternative approach to resource discovery is to pre-structure the resources into a hierarchy. This structured approach can likely keep dissemination overhead lower. However, additional overhead is needed to maintain the structure, especially with dynamic resources. This paper explores the unstructured approach only.

Several existing approaches combine epidemic protocols with hierarchical multicast structures [5, 10, 12, 18], but their maintenance requirements make them difficult to scale to grid environments. Lin and Marzullo propose directional gossiping in a wide-area network, where a node forwards information it receives with higher probability to the nodes that have less connections to itself, and vice versa [9]. The underlying intuition is that the overhead of gossip protocols in a wide area network can be reduced by considering the network topology [14]. However, in wide-area networks, overhead with directional gossip approximates flooding.

Karp et al. study randomized rumor spreading to disseminate information to the entire network in the minimum possible number of communication steps [8]. Based on coverage characteristics of pull and push algorithms, they divide the entire coverage process into two parts. The *median-counter* algorithm uses push to exponentially increase coverage in the first part, and switches to pull to quadratically shrink non-coverage in the second part. Our work is similar in how remote nodes are selected and information is sent. However, in our work, the remote nodes disseminate to their vicinity after being selected, rather than randomly. Our approach also differs in terms of the targeted coverage: we aim to cover only some remote parts of the grid, not the entire network.

3 Implementation

In this section, we describe our approach to proxy-based grid information dissemination.

3.1 Motivation

The intended benefits of proxy-based aggregate grid information dissemination are two-fold. First, it helps increase dissemination coverage with less overhead than is incurred with regular information dissemination. This is especially useful for nodes with similar resource characteristics, such as a large cluster of computing nodes. One integral goal of information dissemination—to match resource providers with requesters—is difficult when all the neighbors of a node offer the same resources as it does.

Second, proxy-based dissemination reduces packet overhead by decreasing (if not completely eliminating) the number of intermediate nodes that disseminate in between the source and the destination. If the disseminated information is of no interest to the intermediate nodes, most messages are unnecessary. The number of unnecessary messages increases exponentially as the distance between the provider and the requester becomes larger, even with partial dissemination protocols.

One drawback of using proxies is that information is not necessarily disseminated more aggressively or with more frequency to nearby nodes. Proxies essentially provide an agent for efficient remote advertising. This can increase coverage at reduced overhead costs, but it can also reintroduce problems such as flashcrowding [7] and scheduling applications at distant launch points, thereby increasing

startup costs. Therefore, we advocate the use of proxies for dissemination primarily when localized approaches do not succeed in increasing coverage or query satisfaction rates.

3.2 Proxy-based Grid Information Dissemination

We view dissemination proxies as complementary to the adaptive information dissemination and scoring scheme introduced previously [3, 4]. In particular, we consider the possibility of an autonomous node (or a neighborhood of nodes) adjusting the dissemination behavior by switching dissemination proxies *on* or *off*, and adjusting the number of proxies, under dynamic available resource state. For example, proxies can be used to attract remote nodes when previously matched nearby requesters may not be interested in the providers anymore.

In our view of proxy-based grid information dissemination, an autonomous node or a neighborhood of nodes (via their neighborhood leader) may:

- select a remote node as a proxy,
- request to “attach” to the remote node,
- add the node to its list of active proxy nodes,
- update the proxy nodes with available resource state periodically,
- evaluate the benefit of each proxy node, and
- detach from a proxy node.

In this model, we call the remote node that receives and disseminates information to its vicinity the *proxy*, and the node that uses a proxy the *disseminator*. Disseminators “attach” to proxies and act as their *clients*. Thus, we use the terms disseminator and client interchangeably in this paper.

A candidate node to which a proxy request may be sent is called a *potential proxy*. Our approach selects a potential proxy at random or from a grid directory service, as explained in Section 3.2.3. Nodes that receive proxy requests are autonomous; that is, the decision to accept or refuse an attachment request is at the receiving node’s discretion.

3.2.1 Neighborhood leaders as proxies

In our model, a node can be a provider, a requester, or both. Nodes form neighborhoods that consist of small groups of co-located nodes with similar resource characteristics. Nodes use *local*, *neighborhood* and *grid* scores to determine dissemination aggressiveness adaptively. Each score is a combination of factors that include grid offered load, local utilization, and query satisfaction rate. Scores are normalized between 1 and 100.¹ To minimize the perturbations caused by adaptation, we define *score ranges* and compare the ranges of local, neighborhood and grid scores, as opposed to directly comparing the scores themselves. A local, neighborhood, or grid score may be in the *low*, *normal*, or *high* range. A neighborhood leader tracks its score range and adaptively decides to request a remote node for proxy-based dissemination.

Each neighborhood has a supernode—a *neighborhood leader* that is responsible for functions such as membership

maintenance [16] and neighborhood score calculation [3]. Each neighborhood leader is also a potential proxy node for remote nodes that are interested in disseminating in its neighborhood.

Thus, responsibilities of a neighborhood leaders include:

- recordkeeping—maintaining the neighborhood score and neighborhood score range,
- keeping track of aggregate available resource information in the neighborhood,
- disseminating aggregate available resource information outside the neighborhood,
- disseminating received available resource information to the neighborhood, and
- gatekeeping—receiving reservation requests and forwarding them to actual neighborhood members (see Section 3.2.2 for more information).

Algorithms 1 and 2 detail neighborhood leader tasks.

Algorithm 1 Pseudo-code for neighborhood leader tasks.

```

1: Register itself as a neighborhood leader on the grid.
2: repeat
3:   Poll local scores from neighborhood members,
4:   Calculate the neighborhood score,
5:   Determine the neighborhood score range,
6:   Calculate aggregate available resource state for the
   neighborhood,
7:   Disseminate aggregate available resource state to the
   grid,
8:   Send aggregate available resource state to proxies,
9:   Adjust the number of proxies,
10:  repeat
11:    for all reservation requests received do
12:      Pick an actual provider among neighborhood
      members
13:      Forward the reservation request to that node
14:    end for
15:    for all proxy requests received do
16:      if client limit is not reached then
17:        Accept proxy request
18:        Put the node in clients list
19:      else
20:        Refuse proxy request
21:      end if
22:    end for
23:    for all drop requests received do
24:      Remove sender from client list
25:    end for
26:    for all received aggregate available resource state
    do
27:      Update local client list
28:      Disseminate to neighborhood
29:    end for
30:  until current cycle ends
31: until simulation ends

```

¹See [4] for details on how scores are calculated.

Algorithm 2 Pseudo-code for adjusting number of proxies at a neighborhood leader.

```
1: if neighborhood score range is lower than grid score
   range then
2:   if active proxy limit is not reached then
3:     Pick a potential proxy node
4:     Send a proxy request to potential proxy node
5:     if request is accepted then
6:       Add the potential proxy node to active proxy list
7:     end if
8:   end if
9: else if neighborhood score range is higher than grid
   score range then
10:  Pick a random proxy node from active proxy list
11:  Send a drop proxy message to proxy node
12:  Remove node from active proxy list
13: end if
```

3.2.2 Resource state aggregation

Neighborhood members do not disseminate autonomously as in our previous work [4, 3], instead the neighborhood leader maintains and disseminates the aggregate available resource state on the grid. To determine aggregate available resource state, we add local available resource state information to the periodically exchanged status messages between neighborhood leader and members to maintain the neighborhood score.

Thus the neighborhood leader receives the available resource state from each member periodically, and updates aggregate resource state accordingly. Since the neighborhood leader is the only disseminator for the neighborhood, resource reservation requests for aggregate resources come to the neighborhood leader as well. The neighborhood leader then forwards each request to a neighborhood member, and decreases aggregate available resources. After resource scheduling is complete (i.e. both parties accept the schedule), the provider informs its leader with the next periodic available resource state update, and the neighborhood leader uses this update to verify aggregate resource state.

3.2.3 Selecting a proxy

The neighborhood leader may pick the potential proxy node by one of the following methods:

- *Well-connected, designated* (WCD)—picking a set of nodes that have at least eight connections to other nodes and that are well-distributed across the network, or
- *Random*—sending a random-walk packet that gets forwarded for a random hop count; whichever node the packet stops at acts as the proxy for that dissemination.

Well-connected, designated nodes may not be the “best” possible proxy nodes for increasing coverage footprints. However, a complete analysis would require testing all possible proxy nodes in all different topologies. Therefore, we use well-connected, designated nodes as a heuristic, with

the intuition that they might provide a basis for comparison with random proxies. To exploit the spatial locality of neighborhood query generation, we also ran tests using execution history as another alternative proxy selection method, with the intuition that when a requester is asking for resources, there is a higher possibility that other requesters may be asking for similar resources. Execution history was something that worked well in [2], but in our tests with proxies, it underperforms the random method. Thus we do not report results for this method in this paper.

Algorithm 3 gives proxy selection details for the WCD method, wherein a leader election scheme such as in [16] may pick nodes at optimal locations in each neighborhood in terms of average distance to all other neighborhood members. Thus, there is reason to believe those neighborhood leaders to be attractive candidates as potential proxy nodes to disseminate into that neighborhood. However, making available the locations of all neighborhood leaders on the grid requires a structuring approach with state information kept at special nodes throughout the grid; this incurs the overhead of all structured approaches.

Alternatively, in the random method, nodes may use request forwarding: when a node receives a proxy request, it may either accept the request, or *forward* the request to its neighborhood leader. We have promising results with request forwarding and plan to use random proxies with request forwarding to trade off between the cost of making optimal locations at each neighborhood available to all grid nodes, and proxy-based dissemination overhead when such information is not available. For the results reported in this paper, however, we use well-connected, designated nodes with the intuition that they may be better locations for advertising available resource state.

Algorithm 3 Pseudo-code for picking a potential proxy node in the WCD method.

```
1: if candidateProxyList is not populated then
2:   Create a local copy of well-connected, designated
   potential proxies registry, candidateProxyList
3: end if
4: for all new proxy node requests do
5:   Pick the first entry in candidateProxyList
6:   Send a proxy request to remote node
7:   if request is accepted then
8:     Move the node from candidateProxyList to active-
       ProxyList
9:   else
10:    Move the node to the end of the candidateProxyList
11:   end if
12: end for
```

4 Simulation and Results

We use the Scalable Simulation Network Framework (SSFNet) [1] to simulate the system, and GT-ITM [17] to

generate network topologies. We use a 600-node transit-stub topology at the network level and a power-law topology in the overlay. All simulations execute for 100 cycles; each data point represents the average of 180 runs, 20 separate runs with different random seeds. For each run, we report an average of nine tests with initial dissemination probabilities $P_i = 0.1$ to 0.9.

In our model, a resource is characterized by a resource descriptor tuple (T, U, S) , where T refers to the Type of resource, U refers to available resource Units, and S refers to available time Slots. A request is characterized by the same three parameters contained in the descriptor. Providers disseminate available resource state in this tuple format, which then gets stored in information repositories at remote nodes. A query is satisfied if the following relationship holds between the resource tuple (T_r, U_r, S_r) and the query tuple (T_q, U_q, S_q) :

$$(T_r = T_q) \wedge (U_r \geq U_q) \wedge (S_r \geq S_q).$$

We report dissemination overhead in terms of the total number of hops travelled by information packets. We report query satisfaction rates as the ratio of satisfied queries over the total number of queries generated.

The experiments in this section are organized as follows. Section 4.1 shows how proxy-based dissemination reduces dissemination overhead and increases query satisfaction rates with well-connected, designated (WCD) proxies. In Section 4.2, we study an alternative proxy selection method with random proxies; and compare its performance to well-connected, designated (WCD) proxies. Section 4.3 details each proxy selection method, and Section 4.4 helps verify the results by studying additional cases.

4.1 Well-Connected, Designated Proxies

We report the behavior of a proxy based information dissemination scheme under different resource characteristics. For the proxy-based dissemination results reported in Sections 4.1.1 and 4.1.2, nodes pick well-connected, designated proxies from a central registry as explained in Section 3.2.3.

We run a total of 16 tests in two groups, and report their overhead and query satisfaction rates. In Section 4.1.1, we report 12 basic tests based on relationships between providers and requesters. In Section 4.1.2, we report results of four additional tests that have more participating nodes and neighborhoods in the simulation.

4.1.1 Relationships between providers and requesters

Tests 1 through 12 vary the following factors that affect the performance of partial dissemination protocols: (1) relative locations of provider and requester nodes, (2) query generation style, and (3) system saturation level.

We study the providers and the requesters in two node placement configurations. In the first configuration (c_1), two sets of four providers each form two neighborhoods (called PN1 and PN2 respectively). They are each placed at a corner of the grid such that they are far from each

Table 1. 12 tests that represent basic relationships between providers and requesters in terms of resource characteristics and relative positions.

Test Case	Placement	Resource Provider	Beaming	Saturation Level
1	c_1	Both	None	S_u
2	c_1	Both	None	S
3	c_1	Both	None	S_o
4	c_2	Both	None	S_u
5	c_1	PN1	None	S_u
6	c_1	PN1	R1	S_u
7	c_1	PN1	None	S
8	c_1	PN1	R1	S
9	c_1	Both	Both	S
10	c_1	PN1	Both	S
11	c_2	Both	Both	S
12	c_2	PN1	Both	S

other. Two requesters, R1 and R2, are also added to the two neighborhoods PN1 and PN2 to form close groups with their respective providers. In the second configuration (c_2), we place PN1, PN2, R1, and R2 at four mutually remote corners of the grid. In each specific test case, either both PN1 and PN2, or only PN1 provide resources.

We also vary resource characteristics: R1 and R2 either generate queries during the entire simulation (constant) or they beam queries in alternating periods of dissemination cycles. (ON, OFF, ON, OFF, etc.). Further, we vary the ratio of available resources (AR) versus offered load (OL), and study three resource saturation levels:

- *under-saturated* (S_u), where $AR = OL * 3$,
- *saturated* (S), where $AR = OL$, and
- *over-saturated* (S_o), where $AR = OL/3$.

In all the tests, resource state aggregation helps decrease dissemination overhead by decreasing the number of dissemination sources. The adaptive proxy selection algorithm also affects dissemination overhead by adjusting the number of proxies (see Algorithm 2 for more details).

For each test, we report the proxy-based dissemination performance normalized to the non-proxy baseline for both dissemination overhead and query satisfaction rate. Table 1 summarizes each test case, and Table 2 shows results.

In general, Table 2 shows that proxy-based dissemination reduces overhead in most tests, up to 34%, and it matches query satisfaction rates of regular information dissemination. Moreover, it improves query satisfaction rates by up to 18%, in some cases with higher query satisfaction rates and lower overhead.

Tests 1–6 show that proxy-based dissemination is better when the system is not under-saturated. In particular, proxy-based dissemination reduces overhead by 33% on a saturated system, and also by 29% on an over-saturated system. On the other hand, proxy-based dissemination has higher overhead than regular information dissemination

Table 2. Proxy-based dissemination performance in terms of overhead and query satisfaction rate, for isolated test cases. Proxy-based dissemination has up to 44% higher overhead, due to under-saturated system. On the other hand, proxy-based dissemination reduces overhead up to 34% when the system is saturated. Moreover, it matches query satisfaction rate of regular information dissemination within 18%, and improves it by up to 18%.

Test	Overhead			Query Satisfaction		
	No Proxy	Proxy	Ratio	No Proxy	Proxy	Ratio
1	20,593	29,676	1.44	99.73%	93.99%	0.94
2	18,909	12,597	0.67	91.56%	91.01%	0.99
3	71,586	50,826	0.71	30.62%	27.69%	0.90
4	98,315	83,317	1.18	96.71%	94.36%	0.98
5	27,639	34,825	1.26	90.16%	80.24%	0.89
6	29,298	38,673	1.32	87.81%	71.74%	0.82
7	26,759	17,661	0.66	84.17%	74.59%	0.89
8	30,154	22,012	0.73	81.14%	81.32%	1.00
9	25,366	19,278	0.76	93.32%	95.60%	1.02
10	31,129	24,814	0.80	86.93%	93.75%	1.08
11	91,024	62,839	0.69	87.04%	90.69%	1.04
12	35,677	24,575	0.69	75.94%	89.65%	1.18

when the system is under-saturated. In Tests 1 and 4–6, the adaptive algorithm generates more aggressive dissemination protocols on-the-fly, to attract more requesters. This in turn causes higher dissemination overhead with more dissemination sources in proxy-based dissemination.

Beaming in query generation in Tests 6 and 8 causes the provider neighborhood (PN1) to increase aggressiveness in both proxy-based and regular information dissemination (Test 5 compared to Test 6, and Test 7 compared to Test 8). With beaming, the increase in aggressiveness is higher in proxy-based dissemination, thus higher overhead ratios in Tests 6 and 8, compared to Tests 5 and 7, respectively.

Tests 9–12 show that proxy-based dissemination performance is not affected by the distance between the providers and the requesters. Under different resource characteristics, proxy-based dissemination reduces message overhead up to 31%, while it exceeds query satisfaction rates of regular information dissemination in all cases. With proxy-based dissemination, the increase in overhead is higher in Test 10 compared to Test 9, due to single provider neighborhood satisfying requests of both close and distant requesters. This effect is not visible in Tests 11 and 12, because of the relatively distant placement of requesters and providers.

4.1.2 More providers and requesters

We report performance of the isolated test cases in the previous section. In this section, we distribute the providers and requesters on the grid, and increase the number of neighborhoods. We report the performance of four additional tests in Table 3, where the providers form a neighborhood central to the grid, and the requesters form a distributed neighborhood

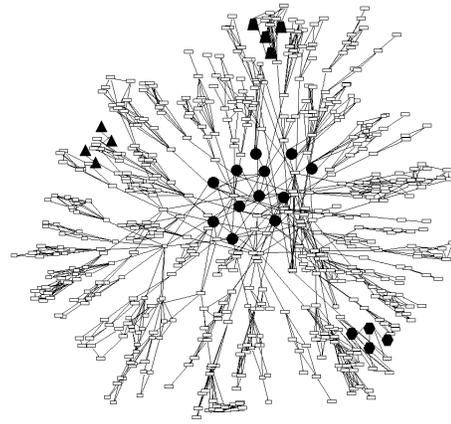


Figure 1. Distributed neighborhood nodes on the topology. Black circles denote providers, black triangles, hexagons and trapezoids denote requester groups. Empty rectangles denote other grid nodes. 12 requester nodes form the distributed requester neighborhood at three parts of the grid and 12 provider nodes form the provider neighborhood.

Table 3. Proxy-based dissemination performance in terms of overhead and query satisfaction rate, for generalized test cases. Proxy-based dissemination reduces dissemination overhead by up to 65% while increasing query satisfaction rates up to 19%.

Test	Overhead			Query Satisfaction		
	No Proxy	Proxy	Ratio	No Proxy	Proxy	Ratio
13	164,087	58,079	0.35	70.34%	76.62%	1.09
14	115,658	58,145	0.50	45.98%	55.71%	1.19
15	149,952	57,756	0.39	83.55%	86.96%	1.04
16	123,686	63,684	0.52	78.65%	82.14%	1.04

at the three edges of the grid, as Figure 1 shows. In each test, we vary available resource and offered load characteristics.

Table 3 shows that the benefits of proxy-based information dissemination are not specific to the tests cases we report in the previous section. Proxy-based dissemination reduces overhead by 56% on average, and as much as 65%. In all the cases we report, proxy-based dissemination results in better query satisfaction rates, up to 19%, compared to cases where no proxies are used.

4.2 Randomly Selected Proxies

In the WCD method, we use the six most connected nodes as proxies. For each of the 20 runs in the WCD method, we pick three of the six proxies and use the same proxies across all tests. To study whether the benefits of proxies are only observable with the WCD method or not, we run more tests with both methods. Figure 2, and Tables 4 and 5 compare the performance of the random method to the WCD method, for the four tests reported in Section 4.1.2.

Figure 2 shows that picking proxy nodes at random performs close to the WCD method. We observe that the WCD method has both higher dissemination overhead and higher

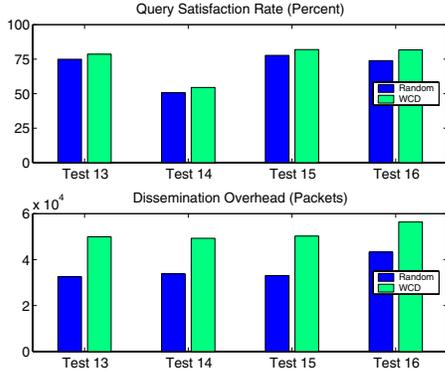


Figure 2. Performance of the random and WCD methods. The random method matches query satisfaction rates of the WCD method with less overhead.

Table 4. Comparison of the random and WCD methods in terms of redundant message ratio and average dissemination probability. Redundant message ratio of the WCD method is lower than the random method. The WCD method has also less aggressive dissemination protocols due to better locations of proxy nodes.

Tests	Redundant message ratio		Average dissemination probability	
	Random	WCD	Random	WCD
13	0.49	0.48	0.64	0.60
14	0.49	0.47	0.28	0.25
15	0.50	0.48	0.62	0.59
16	0.55	0.52	0.37	0.32

Table 5. Comparison of the random and WCD methods in terms of average distance between proxy node and the disseminator, and average edge count of proxies. Lower average edge count contributes to lower overhead with the random method.

Tests	Average distance		Average edge count of proxies	
	Random	WCD	Random	WCD
13	6.75	7.22	3.85	8.83
14	6.90	7.22	3.72	8.83
15	6.85	7.22	3.57	8.83
16	6.78	7.22	3.57	8.83

query satisfaction rates, because we use the most connected nodes, which leads to higher query satisfaction rates. However, the increase in message overhead does not justify the relative increase in query satisfaction rates.

Table 4 shows that redundant message ratio (i.e. the ratio of extra information packets sent per single piece of information) of the WCD method is lower than the random method. The WCD method also have lower average dissemination probabilities, thus they produce less aggressive protocols. This does not result in less overhead however, due to higher number of links in the WCD method, as shown in Table 5.

Table 6. Details of each run in Test 13, sorted by query satisfaction rate for the random and WCD methods. Query satisfaction rate and dissemination overhead.

Query Satisfaction Rate		Dissemination Overhead	
Random	WCD	Random	WCD
0.7777	0.8053	54,293	51,098
0.7725	0.7895	36,154	50,963
0.7698	0.7895	27,984	40,449
0.7640	0.7854	31,853	43,918
0.7597	0.7848	35,512	40,636
0.7595	0.7825	29,283	45,053
0.7569	0.7825	26,395	41,910
0.7567	0.7825	33,470	41,306
0.7553	0.7789	27,200	44,496
0.7537	0.7784	30,577	46,214
0.7522	0.7690	38,963	50,731
0.7520	0.7649	31,059	50,616
0.7415	0.7649	35,054	46,724
0.7398	0.7643	26,592	55,278
0.7395	0.7591	31,474	53,208
0.7316	0.7561	29,086	49,907
0.7310	0.7532	25,812	51,501
0.7287	0.7526	28,115	44,291
0.7275	0.7304	30,492	53,697
0.7222	0.7234	42,293	55,156

4.3 Proxy Selection Alternatives in Detail

To verify our results in the previous section, we now report details of each run for Test 13. Table 6 gives details for the random and WCD methods.

Table 6 reveals that proxy-based dissemination algorithm behavior does not depend on the particular nodes picked. In the long run, picking random proxies allows proxy-based dissemination to achieve performance comparable to picking better proxy nodes.

On the other hand, the best case performance of the random method (i.e. first row in Table 6 for the random method) in terms of query satisfaction rate is better than half of the results with the WCD method, and on average it is within 7% of the WCD method. Moreover, the random method achieves dissemination coverage comparable to the WCD method, with much less overhead in most cases. The dissemination overhead is between 40 to 55 thousands of packets in the WCD method, while it is between 25 to 54 thousands of packets in the random method.

4.4 Random Neighborhoods

In the previous sections, we report performance numbers for one configuration, as explained in Section 4.1.2. To further verify our results, we run the same proxy selection alternatives for a more generalized case, where three providers and three requesters form random provider and requester neighborhoods as follows. For each neighborhood, a random node is picked as the neighborhood leader. Each neighborhood leader disseminates a marker packet, and asks its

neighbors to apply for membership to the neighborhood. The neighborhood leader picks the first five nodes among responding nodes, and form the neighborhood.

Thus, in each test, three neighborhoods (with five nodes in each neighborhood) participate in resource matching. The configuration is comparable to that of Test 13 in Section 4.1.2. The system is *saturated*, and requesters generate queries uniformly in each dissemination cycle. Table 7 shows results. We observe that in general, the random method performance is comparable to the WCD method. Thus, it is not necessary to make available the “best” locations of potential dissemination nodes on the grid.

Table 7. Details of random neighborhoods, for the random and WCD methods. Redundant message ratio, average dissemination probability, average distance between disseminator and proxies, and average edge count of proxies.

	<i>Random</i>	<i>WCD</i>
Query satisfaction rate	61.05%	67.07%
Dissemination overhead (packets)	48,763	74,307
Redundant message ratio	0.56	0.53
Average dissemination probability	0.41	0.40
Average distance	6.78	6.86
Average edge count of proxies	3.59	8.83

5 Summary

Proxy-based information dissemination can help increase coverage footprints and reduce packet overhead by shrinking the scope of dissemination. In particular, we show that dissemination proxies can reduce overhead by up to 65% and improve query satisfaction rates by up to 19%.

Some neighborhood members have shorter average distance to other members, or a larger average edge count, making them desirable as potential proxy locations. Finding these nodes is difficult, however, even if a disseminator had full information about all potential proxy locations in a grid. Such a strategy could also potentially create bottleneck locations, if most disseminators prefer a small subset of nodes as proxies. We show that selecting proxies at random, which avoids both of these problems, performs comparably to picking intuitively “better” nodes as proxies. Thus, in large-scale volatile desktop grids, proxy-based grid information dissemination can increase the efficiency of matching resource requesters to providers.

References

- [1] J. Cowie, H. Liu, J. Liu, D. Nicol, and A. Ogielski. Towards realistic million-node internet simulations. *Proc. of the Int'l Conf. on Parallel and Distributed Processing Techniques and Applications*, 1999.
- [2] 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. *Proc. of CC-Grid2006: IEEE Int'l Symp. on Cluster Computing and the Grid*, 2006.
- [3] D. C. Erdil, M. J. Lewis, and N. Abu-Ghazaleh. An adaptive algorithm for information dissemination in self-organizing grids. *to appear in e-Science 2006: The 2nd IEEE Int'l Conf. on e-Science and Grid Computing*, Amsterdam, the Netherlands, December 4-6, 2006.
- [4] D. C. Erdil, M. J. Lewis, and N. Abu-Ghazaleh. An adaptive approach to information dissemination in self-organizing grids. *Proc. of the Int'l Conf. on Autonomic and Autonomous Systems (ICAS'06)*, Silicon Valley, CA, July 2006.
- [5] I. Gupta, R. van Renesse, and K. Birman. Scalable fault-tolerant aggregation in large process groups. *Conf. on Dependable Systems and Networks*, 2001.
- [6] V. Iyengar, S. Tilak, N. B. Abu-Ghazaleh, and M. J. Lewis. Nonuniform Information Dissemination for Dynamic Grid Resource Discovery. *Proc. of IEEE NCA04: The 3rd IEEE Int'l Symp. on Network Computing and Applications*, 2004.
- [7] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites. *Proc. of the IEEE Int'l World Wide Web Conf.*, 2002.
- [8] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized rumor spreading. *IEEE Symp. on Foundations of Computer Science*, 2000.
- [9] M.-J. Lin and K. Marzullo. Directional gossip: Gossip in a wide area network. *European Dependable Computing Conf.*, 1999.
- [10] A. Nayate, M. Dahlin, and A. Iyengar. Transparent information dissemination. *ACM/IFIP/USENIX 5th Int'l Middleware Conf.*, 2004.
- [11] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed resource management for high throughput computing. *Proc. of 7th IEEE Int'l Symp. on High Perf. Dist'd Computing*, July 1998, Chicago, IL.
- [12] E. C. Shek, S. K. Dao, Y. Zhang, and D. V. Buer. Dynamic multicast information dissemination in hybrid satellite-wireless networks. *ACM Workshop on Data Engineering for Wireless and Mobile Access*, 1999.
- [13] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.
- [14] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. *IFIP Int'l Middleware Conf.*, 1998.
- [15] P. Yalagandula and M. Dahlin. A scalable distributed information management system. *Proc. of ACM SIGCOMM*, Portland, OR, Sept. 2004.
- [16] W. Yang, N. B. Abu-Ghazaleh, and M. J. Lewis. Automatic clustering for self-organizing grids. *Proc. of IEEE Int'l Conf. on Cluster Computing*, 2006.
- [17] E. Zegura and K. Calvert. GT Internetwork Topology Models (GT-ITM). <http://www.cc.gatech.edu/projects/gtitm>.
- [18] W. Zhang, G. Cao, and T. L. Porta. Dynamic proxy tree-based data dissemination schemes for wireless sensor networks. *The 1st IEEE Int'l Conf. on Mobile Ad-hoc and Sensor Networks*, 2004.