Message-Efficient Self-Organization of Wireless Sensor Networks

Rajesh Krishnan
BBN Technologies
10 Moulton Street
Cambridge, MA 02138, USA
Phone: +1 617 873 4684
Email: krash@bbn.com

David Starobinski[†]
Boston University
8 Saint Mary's Street
Boston, MA 02215, USA
Phone: +1 617 353 0202
Email: staro@bu.edu

***Abstract*— Distributed self-organization algorithms for wireless sensor (and actuator) networks must have low message complexity from energy and bandwidth considerations. In this paper, we present a novel approach for message-efficient clustering, in which nodes allocate local *growth budgets* to neighbors. We introduce two algorithms that make use of this approach. Unlike the expanding ring approach [10], our algorithms do not involve the initiator in each round, and do not violate the specified upper bound on the cluster size at any time. We derive analytical performance bounds of our algorithms and also provide performance results from simulations. The algorithms produce clusters of bounded size and low diameter, using significantly fewer messages than the expanding ring approach.**

I. Introduction

Wireless sensor (and actuator) networks have critical applications in the scientific, medical, commercial, and military domains. Examples of these applications include environmental monitoring, smart homes and offices, surveillance, intelligent transportation systems, and many others. As societal reliance on wireless sensor networks increases, we can expect the size and complexity of individual networks as well as the number of networks to increase dramatically.

For many applications, we envision large static ad hoc networks consisting of hundreds, thousands, or even millions of inexpensive wireless sensor nodes that may be placed either regularly or irregularly. Wireless sensor networks can be exposed to highly dynamic and hostile environments, and therefore, they must be tolerant to the failure of individual nodes. Algorithms for wireless sensor networks must be *distributed* to prevent single points of failure, and *self-organizing* for scalable deployment.

Wireless sensor networks are *bandwidth-* and *energy-constrained*. Self-organization algorithms that minimize the number of message transmissions (and receptions) can help conserve energy and bandwidth. Protocols with low message overhead are preferable in tactical wireless sensor networks, since for example, statistical methods of cryptanalysis require a large number of samples to be successful. Therefore, *low message complexity* is a highly desirable property of self-organization algorithms for wireless sensor networks.

[†]This work was supported in part by the National Science Foundation under NSF CAREER grant ANI-0132802.

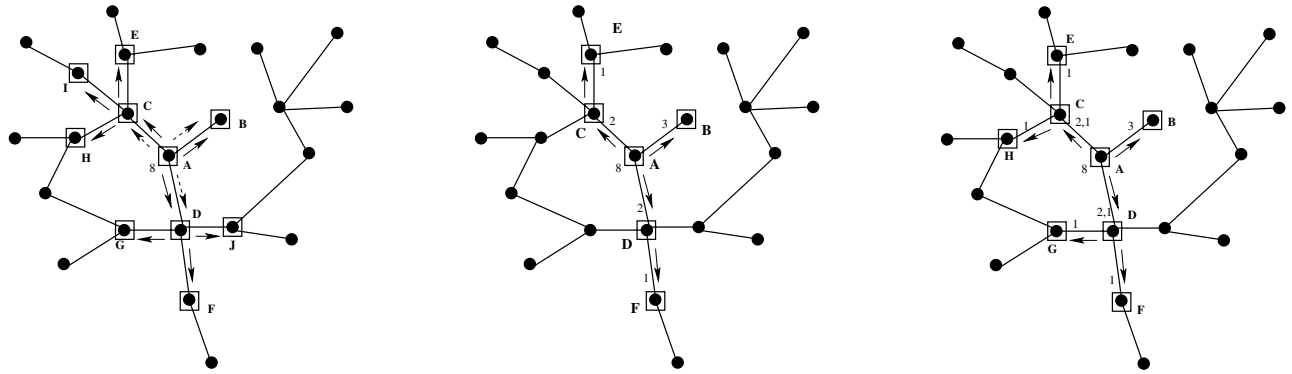
The autonomous set up of addressing and routing in a large network requires the decomposition of the network into connected *clusters*. Bounding the cluster size[1] is a requirement imposed by routing protocol complexity [5], [6], [8], [10]. Furthermore, resource constraints or specific network architectures can impose limits on the cluster size. Thus, it is desirable to develop distributed algorithms that can form bounded size clusters. Moreover, the cluster sizes produced should be as close as possible to the specified bound in order to limit the total number of clusters. This can help the construction of a flatter hierarchy for efficient routing.

Despite the importance of the bound on cluster size, clustering algorithms so far have treated size as a secondary constraint, to be optimized in a post-processing phase [2], [10]. Ramamoorthy *et al.* apply an expanding ring approach [10], in which the cluster depth is progressively relaxed until the desired cluster size is exceeded. Additional messages are used to selectively drop excess nodes picked up in the outermost layer. As shown in the sequel, this approach allows temporary violations of the bound and often leads to the transmission of an excessively large number of messages. More recently, Banerjee and Khuller [2] proposed new algorithms for constructing overlapping clusters of bounded size. Their algorithms use a post-processing phase to enforce the size bound.

Other work in this area has focused on algorithms to achieve sparse, low-diameter network decompositions [1], [7] based on the rationale that a low diameter leads to low end-to-end delay. However, these algorithms can allow the cluster size to be violated arbitrarily. This is especially true in densely connected networks. For example, consider the extreme case of a clique network in which the desired cluster size is small compared to the network size. Every cluster in this case will have unit diameter, but the size bound can be violated arbitrarily.

In this paper, we propose a novel approach for message-efficient clustering based on the concept of budget allocation. Our novel approach relies on allocating *growth budgets* to neighbors. This approach significantly reduces the number of messages exchanged and allows the cluster to grow based on local decisions rather than involving the initiator at each round.

[1]The *size* of a cluster is the number of nodes that it includes.



(a) Expanding Ring (b) Rapid (c) Persistent

Fig. 1. Distributed algorithms for bounded size clustering.

We introduce two algorithms making use of this approach. The first algorithm called *Rapid* produces clusters of bounded size. This algorithm uses few messages, but has a poor worst-case analytical performance. The second algorithm called *Persistent* persistently tries to produce a cluster of the specified bound if possible. Simulations show that this algorithm performs well on average when building a single cluster.

Through analysis and simulation, we demonstrate that our algorithms are significantly more efficient in terms of message complexity than the expanding ring search algorithm of [10]. In our analysis scenarios that were restricted (for tractability) to the construction of a single cluster within a network, the proposed algorithms have a worst-case message complexity that is polynomial in the specified cluster size, while the worst-case complexity of expanding ring search is exponential.

The proposed algorithms do not violate the cluster size bound at any time. They generally produce *flat* rather than *deep* clusters [2]. This is advantageous because flat clusters lead to smaller end-to-end delays.

This paper is organized as follows. In Section II, we describe the proposed Rapid and Persistent algorithms and apply them to network decomposition of wireless sensor networks. We present a performance analysis of the algorithms in Section III, followed by simulation results in Section IV. The last section of this paper is devoted to concluding remarks, followed by bibliographic references at the end.

II. Algorithms

In this section, we describe two novel algorithms to produce clusters of bounded size – Algorithm Rapid and Algorithm Persistent. We also describe the Expanding Ring algorithm [10], which we use as a baseline for performance evaluation. We note here that an arbitrary network cannot always be decomposed into non-overlapping clusters of a particular size. For example, with a star topology, once the hub node is assigned to a cluster, then all unclustered nodes will end up in single node clusters.

[2]It is possible to build clusters of a bounded size by using a depth-first-search (DFS) algorithm, but in large networks, this can result in deep clusters.

We provide the definitions of some terms used in this paper. We consider a network $G = (V, E)$ with $|V| = n$ nodes and $|E| = m$ edges. Nodes in G can send (and receive) messages to (from) its neighbors. Every node $v \in V$ has a unique identity $\mathrm{ID}(v)$. Each node initially knows only its own identity and the identities of its immediate neighbors in G.

A *cluster* is a connected component of the original graph G together a node designated as the initiator in the cluster and a spanning tree (a by-product of clustering) [1]. An *initiator* $v_0 \in V$ is a node that is permitted to initiate the process of clustering by sending messages. The *size* of a cluster is the number of nodes belonging to it. The bound on the maximum cluster size is B, where B is smaller than n. The *depth* of the cluster is the height of the associated spanning tree.

A. Algorithm Expanding Ring

Ramamoorthy *et al.* present an algorithm for bounded size clustering based on an expanding ring search [10]. The Expanding Ring algorithm proceeds in rounds starting with a maximum hop limit of 1. In each round of the algorithm, the initiator sends messages with a scope set to one hop more than the previous round. At the end of each round, the initiator knows the total number and IDs of nodes that are added in the last layer, that is, nodes at a distance equal to the maximum hops for that round. Eventually, after a few rounds, the cluster size bound will be exceeded. For example, in Figure 1(a), a bound of 8 is exceeded at 2 hops resulting in a cluster containing 10 nodes.

The initiator then sends another message containing a list of (arbitrary chosen) nodes in the last layer to be dropped from the cluster in order to achieve the bound. This message is then forwarded successively to all children.

B. Algorithm Rapid

In this algorithm, the initiator is assigned a budget of B, of which it accounts for itself and *evenly* distributes $B-1$ among its neighbors by sending a message to each one of them. An arbitrary subset is chosen if there are more neighbors than the budget. The neighbors that receive the message, account for themselves and distribute the remaining among all their

neighbors except the parent. The messages propagate until they reach a stage where the budget is exhausted. This algorithm is illustrated in Figure 1(b) in which a bound of 8 results in forming an actual cluster of size 6. Note that the initiator (node A) allocates a budget of 3 to a neighbor (node B) which is a leaf and can add only 1 node to the cluster.

Each node that receives a message sends an acknowledgment to its parent when either the budget is exhausted or it has received acknowledgments from all its children. The algorithm terminates when the initiator receives acknowledgments from all the neighbors it sent a message to. The acknowledgments can be used to convey the size and depth of the subtree and the maximum hop count reached. The initiator can use this to determine the size and depth of the cluster. Furthermore, by including hop count information in the messages and acknowledgments, a modified algorithm can limit the depth of the cluster, but this can affect the cluster size achieved.

C. Algorithm Persistent

The Rapid algorithm presented in the previous section has a low message complexity, at most $2(B-1)$ messages (see Section III). However, in the worst-case it can produce clusters that are very small when compared to the desired bound. Decomposing a network into a large number of small clusters can lead to inefficient routing.

Algorithm Persistent that we present next, uses more messages than Algorithm Rapid to significantly improve the worst-case behavior, that is, the smallest cluster size produced.

Using the Persistent algorithm, the initiator is provided a budget of B, of which it accounts for itself and evenly distributes $B-1$ among its neighbors via messages. An arbitrary subset is chosen if there are more neighbors than the budget. The neighbors which receive the message, account for themselves and distribute the remaining among all their neighbors except the parent. The messages propagate until they reach a stage where the budget is exhausted.

In this algorithm, each node that receives a message does not send an acknowledgment to its parent immediately on receiving acknowledgments from all its children. It computes the size of the subtree and compares it to the budget allocated to it. It distributes the shortfall among its neighbors that either were not explored previously or met all previously allocated budgets. When either the budget is met or when further growth is not possible, it returns an acknowledgment to its parent. The algorithm terminates when the initiator determines that the bound has been met or when no further growth is possible (for example, there are not enough nodes adjacent to the cluster that are free to join this cluster.) This algorithm is illustrated in Figure 1(c). In this example, node A first allocates a budget of 3 to node B. Node B returns an acknowledgment to node A, with a budget consumption of 1. Node A then equally allocates the surplus budget of 2 among its other children (C and D).

In the presence of a single initiator, Algorithm Persistent produces a cluster of the specified bound. When a cluster of the specified size cannot be constructed (because not enough unclustered nodes are available), it attempts to build the largest cluster possible.

D. Network decomposition

The three algorithms described above, can each produce a single cluster of bounded size. We can apply any one of them to perform network decomposition into clusters of bounded size using the simple procedure described below.

Each node that comes up waits for clustering messages from its neighbors. When a timeout (randomly chosen within a configured interval) occurs, the node becomes an initiator itself and invokes one of the three clustering algorithms. The process continues until all the nodes in the network are clustered.

III. Analysis

In the context of wireless sensor networks with energy-limited and bandwidth-limited devices, message complexity is the metric of choice to evaluate the efficiency of an algorithm. In this section, we analyze the message complexity of the three algorithms when applied to both the construction of a single cluster and network decomposition.

In Section III-A, we show that in the presence of a single initiator the Rapid and Persistent algorithms have a worst-case message complexity that is polynomial in the specified bound. The Expanding Ring algorithm has a worst-case message complexity that is proportional to the size of the entire network, and this can be very large.

In Section III-B, we show that the Rapid algorithm can produce very small clusters in the worst case, even with a single initiator. We show that in such a case, the other two algorithms produce a cluster of size equal to the specified upper bound.

In Section III-C, we analyze the message complexity of the algorithms when applied to network decomposition. For tractability of analysis, we restrict ourselves to clique and ring topologies, which represent dense and sparse topologies respectively. On a clique topology, we show that the average-case message complexity of the Persistent algorithm is significantly lower than that of the Expanding Ring algorithm, although both produce the same number of clusters. On a ring topology, the two algorithms have similar asymptotic message complexity, and produce the same number of clusters. From this analysis, we can expect that the Persistent algorithm will require fewer messages on average than the Expanding Ring algorithm for the decomposition of dense networks. Simulations presented later validate this conclusion.

For the purpose of analysis, we assume that during the clustering process: (i) the graph is connected, (ii) there are no changes in the network topology, and (ii) no messages are lost during clustering.

A. Message Complexity for Creating a Single Cluster

Lemma 1: Algorithm Rapid uses $\mathcal{O}(B)$ messages.

Proof: Each message sent by a node to one of its neighbors implies that the total budget is reduced by at least 1. Since the initial budget is B, there will be at most $B-1$ messages (or $2(B-1)$ messages, if one includes the acknowledgments). ■

Lemma 2: Algorithm Persistent has a worst-case message complexity that is polynomial in B.

Proof: The budget can be reallocated at most B times by each node on the spanning tree. So the total number of messages on each link of spanning tree is $\mathcal{O}(B)$ in the worst-case. Each node in the spanning tree could have sent messages to at most $\mathcal{O}(B)$ neighbors that have not been included in the spanning tree as a child of the given node. Again the number of messages wasted is $\mathcal{O}(B^2)$. Thus the worst-case message complexity of Algorithm Persistent is polynomial in B. ■

Lemma 3: Algorithm Expanding Ring uses $\mathcal{O}(n)$ messages.

Proof: In any round in which the cluster size is less than B, the total number of messages exchanged is polynomial in B. In the worst case, all nodes in the network may be sent messages in the last round. Therefore, the worst-case message complexity of expanding ring search is $\mathcal{O}(n)$. ■

From the lemmas above, we see that the proposed Rapid and Persistent algorithms both have message complexity that is polynomial in B. The Expanding Ring algorithm has a complexity that is not related to B and can be very large (e.g. $n = 2^B$).

B. Cluster Size with Unique Initiator

Lemma 4: Algorithm Rapid produces a cluster of size $b \leq B$.

Proof: It is easy to see that Algorithm Rapid produces a cluster of size $b \leq B$. At each stage, the total budget allocated to the node and its neighbors by the algorithm is no more than the budget available at that stage. Since the initiator has a budget of B, the size of the cluster can not exceed B. ■

Theorem 1: With a unique initiator, the lower bound on the worst-case cluster size produced by Algorithm Rapid is $e^{W(\log B)}$ where W is the Lambert-W function[3].

Proof sketch: Suppose the worst-case cluster size produced is $T(B) \leq B$. In order to achieve the worst case, the algorithm must achieve the minimum growth possible at each stage (namely 1), and waste the maximum possible fraction of the total budget available at that stage.

Clearly, each node in the worst-case cluster has no more than $T(B)$ neighbors and at least one neighbor which will be able to use the budget, since G is connected and $n > B$.

In the worst case, the initial budget B is reduced by a factor of $T(B)$ at each stage. With a growth by 1 node at each stage and an initial budget of B, the total number of steps does not exceed $T(B)$. Since the budget must be exhausted after $T(B)$ stages, this leads to the inequality: $\frac{B}{T(B)^{T(B)}} \leq 1$. The solution for $f(x)^{f(x)} = x$ is $f(x) = e^{W(\log x)}$ where $W(x)$ is the Lambert W-function. Thus $T(B) \geq e^{W(\log B)}$. ■

Lemma 5: Algorithm Persistent produces a cluster of size B if the initiator is unique.

Proof: Each node (including the initiator) recursively reallocates the unutilized budget until either the entire budget is exhausted or no more growth is possible. Since the graph is connected, is larger in size than the cluster size bound, and no nodes are unavailable to this initiator since it is unique, Algorithm Persistent produces a cluster of size B. ■

[3]The Lambert-W function is the inverse of the function $f(W) = We^W$, and has the series expansion $W(x) = \sum_{n=1}^{\infty} \frac{(-n)^{n-1}}{n!} x^n$.

Lemma 6: Algorithm Expanding Ring produces a cluster of size B if the initiator is unique.

Proof: It is easy to see that Algorithm Expanding Ring produces a cluster of size B, since it recursively expands the cluster by one hop until either the cluster size equals or exceeds the bound, and then sheds the excess nodes from the last layer if necessary. ■

From the above discussion, we see that the Rapid algorithm may produce very small clusters in the worst case, even with a unique initiator. In such a case, the other two algorithms produce a cluster of size equal to the specified upper bound.

C. Message Complexity of Network Decomposition

We analyze the message complexity of network decomposition of two regular topologies – clique and ring – that represent extremes of sparse and dense topologies respectively. The analysis provides insight into the message complexity of the algorithms both in sparsely and densely-connected graphs.

The analysis in the case of a general graph is difficult. The actual set of initiators and the specific sequence in which they become active will affect the number of clusters, the size of individual clusters, and the total number of messages used. Therefore, in order to simplify the message complexity analysis (and to isolate the effects of implementation details), we assume that only one initiator is active at any given time. In the case of the ring, we also assume that an unclustered neighbor of the last cluster is chosen as the next initiator. Furthermore, the regularity of the ring and clique topologies makes the choice of initiators less critical to the performance analysis. Our simulation-based comparisons (in Section IV) extend to other general topologies and cases in which multiple initiators are concurrently active.

In these two topologies, both the Persistent and Expanding Ring Algorithm produce the same number of clusters. We have computed the worst-case and average-case message complexity, which are summarized in Table I. Our analysis shows that on a clique topology, the Persistent algorithm requires significantly fewer messages on average than the Expanding Ring algorithm. On a ring topology, the two algorithms have the same message complexity.

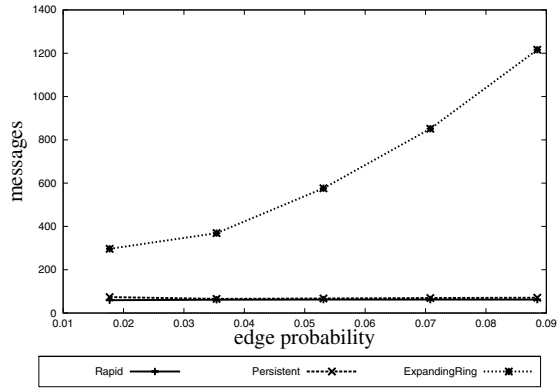
IV. Simulation results

In this section, we compare the performance of the Rapid, Persistent, and the Expanding Ring algorithms using the ns-2 simulator [9]. We consider two cases: (i) when only one initiator is present and a single cluster is produced, and (ii) network decomposition.

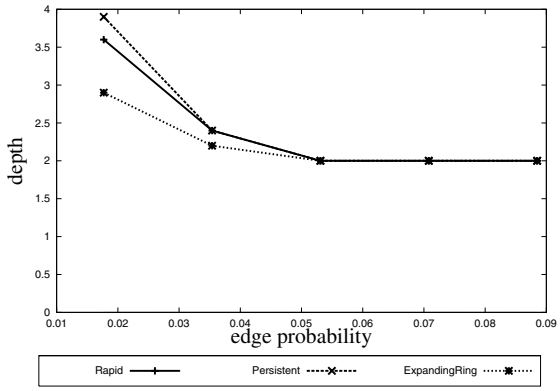
TABLE I

Message complexity of network decomposition in clique and ring topologies of size n.

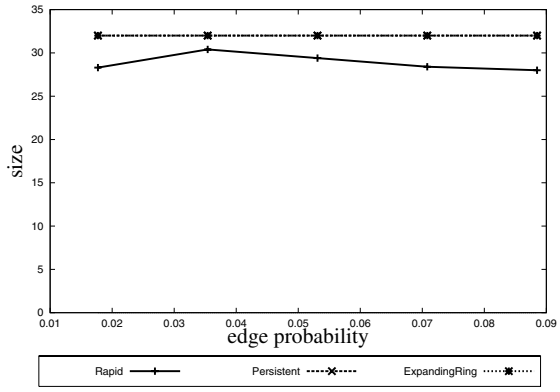
	Clique		Ring
	Average case	Worst case	Average and Worst case
Persistent	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
Exp. Ring	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$



(a) Number of messages



(b) Cluster depth



(c) Cluster size

Fig. 2. Performance of the algorithms with bound = 32 for different α. Each point is averaged over 10 different 200-node random graphs.

We used the GT-ITM tool [3] to generate random graphs based on the Erdos-Renyi model with edge probability α chosen from the set $\{0.0177, 0.0354, 0.0531, 0.0708, 0.0885\}$. The capacity and delay for all links were set to 1Mb/s and 10 ms respectively.

A. Single cluster performance

First, we consider the case of one initiator producing a single cluster. The node numbered 0 was arbitrarily chosen as the initiator and was given a size bound chosen from the range $\{2, 4, 8, 16, 32, 48, 64, 80, 96, 112, 128\}$. We ran the three algorithms – Rapid, Persistent, and Expanding Ring – on the graphs, and compared the size and depth of the clusters produced, and the number of messages exchanged. Each data point is computed by averaging the results from 10 different 200-node graphs for each given α.

Figure 2 shows the effect of graph density on the performance of our algorithms. We have plotted the results for a cluster size bound of 32, and we have noted a similar trend with other cluster size bounds. The proposed algorithms use far fewer messages than the Expanding Ring algorithm (a difference of almost an order of magnitude in some cases). We can make two observations as the graph density increases: (i) The number of messages required by our algorithms increases only marginally; in contrast, the number of messages required by the baseline Expanding Ring algorithm increases rapidly. (ii) The depth of the clusters produced by our algorithms approaches the best possible, matching those produced by the Expanding Ring algorithm.

We note that the clusters produced by the Rapid algorithm are larger than the analytical worst-case size, and are quite close to the specified bound.

B. Network decomposition performance

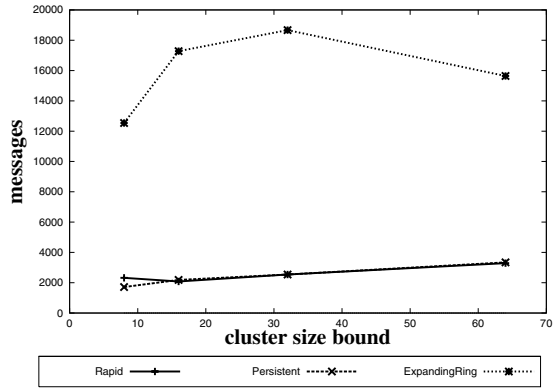
In this section, we consider the case of network decomposition using the three algorithms. The size bound was chosen from the range $\{8, 16, 32, 64\}$. We compare the three algorithms using the number of clusters produced, the average cluster size and the number of messages exchanged. Each data point is computed by averaging the results from 100 different 400-node graphs for each given α.

Figure 3 shows the performance of our algorithms when applied to network decomposition for different size bounds in networks with edge probability $\alpha = 0.0354$. We see that the Rapid and Persistent algorithms continue to use far fewer messages in comparison to the expanding ring algorithm. The Persistent algorithm is similar to the Expanding Ring in terms of the number of clusters (and the average cluster size). Even though the Rapid algorithm retains its message efficiency, it produces a large number of clusters.

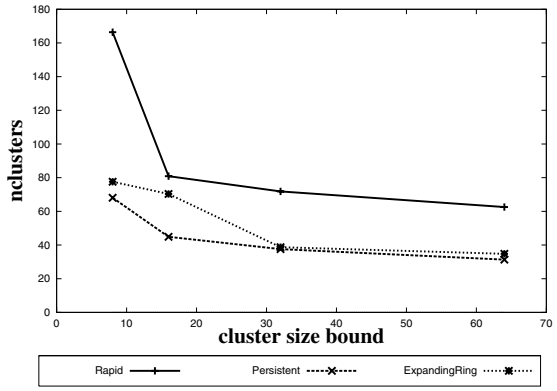
Figure 4 shows network decomposition performance with a size bound of 32 for different graph densities. As expected, the simple Rapid algorithm produces a large number of clusters when compared to the other two algorithms. As the density increases, the number of messages required for network decomposition by both the Rapid and Persistent algorithms increases marginally; with the Expanding Ring algorithm the number of messages required increases rapidly as the density increases. At the same time, the Persistent algorithm produces roughly the same number of clusters as the Expanding Ring algorithm.

V. Conclusions and Future Work

In this paper we presented novel algorithms that produce clusters of bounded size in which nodes allocate local *growth budgets* to neighbors to avoid involving the initiator in each round; they do not violate the specified upper bound on the cluster size at any time. The proposed algorithms also produce

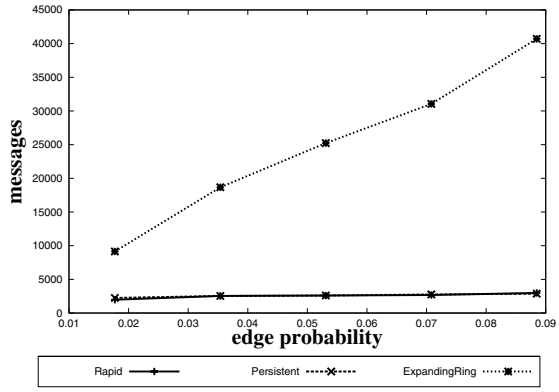


(a) Number of messages

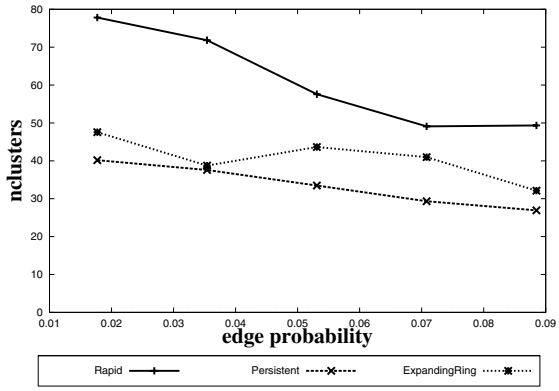


(b) Number of clusters

Fig. 3. Network decomposition performance for different size bounds. Each point is averaged over 100 different 400-node random graphs.



(a) Number of messages



(b) Number of clusters

Fig. 4. Network decomposition performance with bound = 32 and different α. Each point is averaged over 100 different 400-node random graphs.

clusters of low diameter, using significantly fewer messages than the expanding ring approach. The benefits increase with increasing network density.

With a unique initiator, Algorithm Rapid uses the least possible number of messages and guarantees clusters that do not exceed the specified bound, but it can produce very small clusters in the worst-case. Algorithm Persistent uses more messages, but produces a cluster of specified size when possible (i.e. the number of unclustered nodes in the connected component of the initiator equals or exceeds the bound).

When applied to network decomposition, Algorithm Persistent uses significantly fewer messages than the expanding ring approach, while producing a comparable number of clusters (and average cluster size) as the expanding ring approach. Although Algorithm Rapid continues to be message-efficient, it produces a large number of clusters.

We envision that the proposed algorithms can achieve efficient bootstrap of the network organization into clusters of bounded size and low diameter. Overlays to optimize for various criteria can be used in conjunction with our algorithms. Our approach is consistent with, and complementary to, approaches such as LEACH [4], in which different nodes become initiators in different rounds for sharing the energy load. Initiator rotation could also allow for recovery in the event of failed nodes. In future work, we will consider the effects of timeouts and initiator rotation strategies on performance.

REFERENCES

[1] B. Awerbuch, A. V. Goldberg, M. Luby, and S.A. Plotkin, "Network Decomposition and Locality in Distributed Computation," *Proc. IEEE 30th Annual Symposium on Foundations of Computer Science*, Research Triangle Park, NC, USA, pp. 364–369, 30 Oct – 1 Nov 1989.

[2] S. Banerjee and S. Khuller, "A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks," *Proc. IEEE INFOCOM 2001*, Anchorage, AK, USA, Apr 2001.

[3] Georgia Tech Internet Topology Models, Available from *http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz*

[4] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks," *Proc. Hawaiian Int'l Conf. on Systems Science* Jan 2000.

[5] L. Kleinrock and F. Kamoun, "Hierarchical Routing for Large Networks: Performance Evaluation and Optimization," *Computer Networks*, Volume 1, Number 1, pp. 155–174, 1977.

[6] R. Krishnan, R. Ramanathan, and M. Steenstrup, "Optimization Algorithms for Large Self-Structuring Networks," *Proc. IEEE INFOCOM '99*, New York, NY, USA, Mar 1999.

[7] N. Linial and M. Saks, "Low Diameter Graph Decompositions," *Combinatorica,* Volume 13, Number 4, pp. 441–454, 1993.

[8] J. McQuillan, "Adaptive Routing Algorithms for Distributed Computer Networks," *BBN Report Number 2831*, May 1974.

[9] ns-2 simulator, *http://www.isi.edu/nsnam/ns/*

[10] C. V. Ramamoorthy, A. Bhide, and J. Srivastava, "Reliable Clustering Techniques for Large, Mobile Packet Radio Networks," *Proc. IEEE INFOCOM '87*, pp. 218–226, 1987.