Further Observations on the SNL Wireless Sensor Network Leadership Protocol

Thomas C. Henderson

Abstract— **It is sometimes important to have a local leader for a set of wireless sensor nodes. Such a leader may be used as the origin of a coordinate system, as the node responsible for communication, etc. Thus it is important to have a reliable and correct method to assign nodes as leaders. We presented the Sensor Network Leadership (SNL) protocol in previous work. Here we present further properties of this protocol: (1) SNL is an optimal leadership protocol in terms of messages sent, and (2) various statistics concerning the clusters formed by this protocol are given, and in particular, it is noted that in the limit, the cluster placement is similar to circle packing where the radius of the circles is the broadcast range of a sensor device.**

I. INTRODUCTION

We gave an algorithm to solve the *S-cluster* leadership problem [3]. For a good introduction to distributed algorithms, including solutions to variations of the leadership problem and correctness proofs, see [4]. For a leadership election protocol in the context of target tracking, see [9]. Others have introduced leadership protocols (also called cluster formation algorithms); e.g., Chan and Perrig [1] described the ACE algorithm which is an emergent algorithm to form highly uniform clusters, and Shin et al. gave a variation of that [6]. However, both of these algorithms are much more restrictive than SNL in that they require that clusters be disjoint, and thus their methods require an iterative broadcast procedure which consumes much more energy than SNL which requires one broadcast per node to determine the leaders. The leadership problem may be defined as follows:

The Leadership Problem: Each *SEL* has a unique integer ID (UID) and a fixed geographic location; *SEL*s have a restricted broadcast range which defines a connectivity graph. The *SEL*s are to be grouped into subgraphs, called *S-clusters*, such that each *S-cluster* has a leader, and the leader of each *S-cluster* has the lowest ID of all members of the *S-cluster*.

The algorithm is optimal with respect to the number of broadcasts, and has some very nice properties as determined on nodes whose locations are random samples from a uniform distribution in a square area. Given a set of *SEL*s which have determined their neighbors:

- Each *SEL* broadcasts exactly one message during execution of the leadership protocol.
- The number of leaders is bounded by the maximum number of circles (whose radius is the broadcast range) which can be packed into a square area.

T.C. Henderson is with the School of Computing, University of Utah, Salt Lake City, UT USA `tch@cs.utah.edu`

II. LEADERSHIP PROTOCOL

An *S-Net* system will be represented as an undirected graph where each node is a *SEL*. Note that the assumption is that the graph is undirected; however, this is something that must be established by a lower level algorithm (e.g, as part of the communication protocols). It is not the case, in general, that pairs of *SEL*s can receive broadcasts from one another. Each node is a distinct process and each is placed in the environment as a distinct hardware device.

Formal definitions can be given for the nodes, and this involves defining states, including start states, message generating functions, and state transitions. However, only an informal description is given here. Such a description will include *broadcast()* and *receive()* primitive functions with their associated messages. A *broadcast* sends a message to all *SEL*s within range. Proof methods typically involve either invariant assertions and a demonstration that they hold; simulations are used to explore the average case behavior.

The LCR algorithm is a simple example of a leadership algorithm which provides a basic solution to the leadership problem in a synchronous ring network [4]; it involves each process sending its UID in one direction around the ring to its neighbor; when a process receives a UID, it will throw it away if it is less than its own, resend it to its neighbor if it is larger than its own, and declare itself the leader if it is equal to its own. Our solution is related to this idea, although not the same.

The *S-Net* leadership basic algorithm (**SNL**) is executed by each node, and is as follows:

Algorithm **SNL**:

```
Step 1.
    Broadcast own ID for a fixed time, T1.
Step 2.
    Receive from other nodes
    create neighbors
    list for a fixed time, T1
Step 3.
    Create remaining nodes list (initially, neighbors)
    while not done
        if nodes own ID is lower
        than min ID in remaining nodes list,
            then node is leader
```

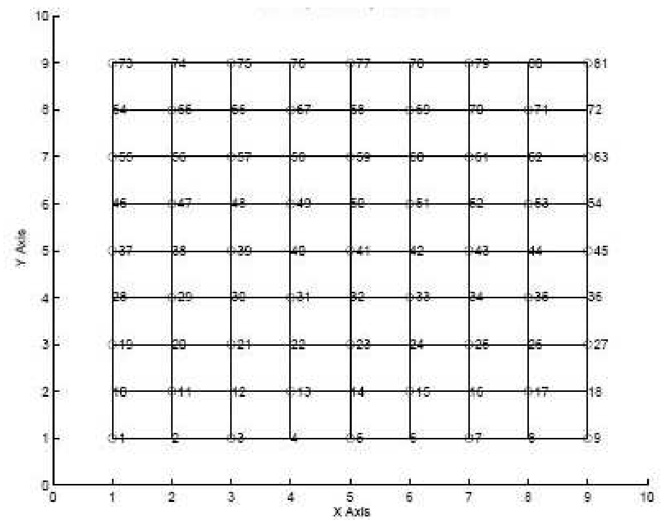

Fig. 1. SNL Protocol Result on a 9x9 Grid with Broadcast Range 1.1 Units.

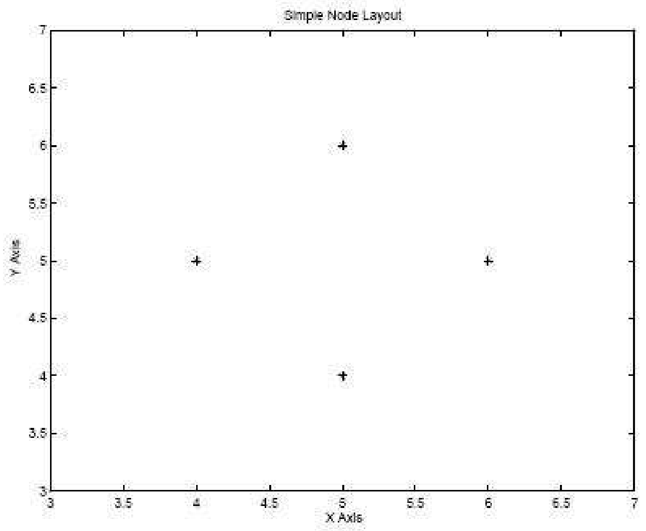

Fig. 2. Simple *SEL* Layout to Demonstrate SNL Protocol.

```
            broadcast cluster (self and neighbors)
            done
        else receive broadcast cluster list
            if in list
                then
                    node is not a leader
                    re-broadcast list
                    done
                else
                    remove list from remaining
```

Note that we assume that enough time is given to Steps 1 and 2 so that each node can complete the step correctly. This will most likely be implemented as a fixed time delay in an embedded system. Also, we assume that there are communications protocols that are reliable enough to transmit the messages without loss of information, and to ensure that communication between nodes is bi-directional.

As can be seen from the algorithm, each node performs exactly one broadcast before it is done. A leader broadcasts its cluster, then quits. A follower broadcasts its cluster, then quits. This is the minimum number of broadcasts possible.

III. SNL SIMULATION

Figure 1 shows the result of running a simulated version of the SNL protocol on 81 *SEL*s which are arranged in a 9x9 grid layout. The broadcast range for each *SEL* is circular with radius 1.1 units; this means each *SEL* can reach its 4-neighbors (distance 1), but not its diagonal neighbors (distance $\sqrt{2}$). This can be verified in the figure as each leader is a circle, and *SEL* n, where n is odd, is a leader.

To better understand the way SNL works, consider the 4-node layout in Figure 2. The node locations, IDs and neighbors are given in Table 3.1. The broadcast range is 1.5 units.

Node ID	x	y	*Neighbors*
1	5	4	2,3
2	4	5	1,3,4
3	6	5	1,2,4
4	5	6	2,3

Table 1: A Simple *SEL* Set.

The nodes proceed asynchronously and at the first iteration of *Step 3*, the following occurs:

Node 1: has a lower ID than its neighbors, and will assert itself as a *leader*.

Node 2: has Node 1 as a neighbor and therefore performs a receive.

Node 3: has Node 1 as a neighbor and therefore performs a receive.

Node 4: has Nodes 2 and 3 as neighbors and therefore performs a receive.

Eventually Node 1 will broadcast its cluster: [1; 2; 3]. The other nodes will loop waiting to receive a broadcast. Nodes 2 and 3 will receive Node 1's broadcast, but Node 4 is out of Node 1's broadcast range and will not receive it.

After Node 1 broadcasts its cluster, it exits and goes to other tasks. Suppose Node 3 receives the broadcast first (this is nondeterministic); then Node 3 finds its ID in the list and asserts itself as a *follower*, rebroadcasts the list, and exits. Node 2 will eventually receive the list and assert itself as a *follower*, rebroadcast the list and exit. Eventually, Node 4 will receive the broadcast from Node 2 or Node 3. Node4 does not find itself in the cluster [1; 2; 3], and it re-assigns its remaining list as [2; 3] - [1; 2; 3] which is the empty list. At this point, Node 4's ID is lower than anything on the list, and so Node 4 asserts itself as a *leader* and exits. Figure 3 shows the resulting leadership structure (Nodes 1 and 4 are *leaders* and Nodes 2 and 3 are *followers*).

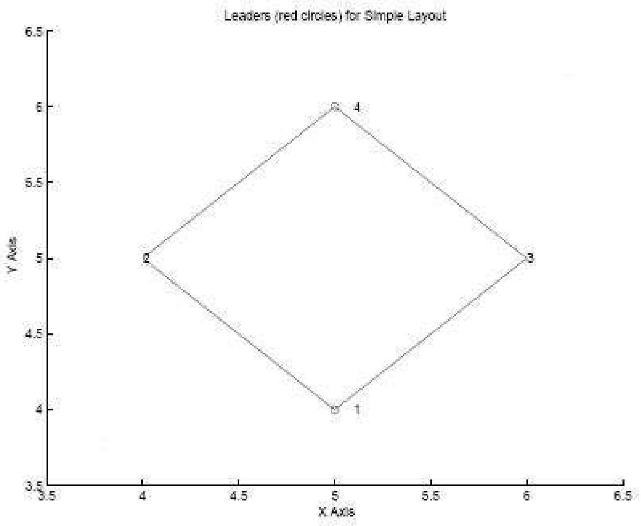

Fig. 3. Result of SNL Protocol on Simple *SEL* Layout.

A. The Simulation Logic

The SNL protocol simulation is organized as follows:

Simulation Protocol:

```
SELs are initialized as described.
Broadcast ID events are scheduled for nodes.
Receive events are scheduled for nodes.
while event-queue $\neq \emptyset$ and $\exists$ unresolved nodes
    Select next event.
    Handle next event.
end.
```

The events are:

Broadcast ID: Broadcast ID and schedule next broadcast ID if still in Phase I (Steps 1 and 2).

Broadcast receive: Receive a broadcast and schedule next receive event if still in Phase I.

Broadcast neighbors: Broadcast neighbors list.

Broadcast cluster: Broadcast cluster list.

Receive ID: Receive ID and schedule next receive ID event if still in Phase I.

Receive cluster: Handle part of Step 3 when node is not a leader; i.e., receives cluster list and either resolves as follower if in list or otherwise subtracts received list from remaining and schedules new receive cluster event.

Phase I timer end: Initializes *SEL*s *neighbors* and *remaining* lists and schedules a first execution of Step 3.1 (i.e., if leader, broadcast cluster; otherwise, schedule a receive list event).

Determine Role: Execute Step 3.1 of SNL algorithm. If *SEL* is not a leader, schedule a receive cluster event.

B. Verification

The algorithm assumes that all neighbor relations are bi-directional. A check is put into the code for this prior to starting Step 3.

Other verification checks include (1) no leader neighbors another leader, (2) every follower neighbors at least one leader, and (3) every *SEL* is resolved (i.e., is either a leader or follower).

Alternatively, this can be formulated as (1) every *SEL* is either a leader or a follower and in a cluster, (2) every follower neighbors at least one leader, and (3) every neighbor of a leader is in its cluster. This is the check performed here; the code has been run on thousands of randomly generated networks, and correctness tested.

C. Validation

There are many sensor networks whose structure can be exploited to test validity. For example, all odd-sided unit grids numbered by row whose *SEL*s have broadcast range 1.1, should have all odd nodes as leaders. Regular polygon nets with *SEL*s on the unit circle and broadcast range $1.1\sqrt{2(1-cos(\theta))}$, where θ is the angle between two adjacent points, should only have the two nearest polygon points as neighbors. Matlab simulations have been run for max nodes up to 200 without error to test validity on such polygon nets (a ring network).

D. SNL Protocol Statistics

The SNL protocol results in a structure of *leaders* and *followers*, and some of the properties of this structure are of interest. Given a set of n node locations sampled from a uniform 2-D distribution, and with randomly assigned *SEL* IDs, we study the following statistics:

- average number of leaders, and
- their spatial distribution.

To obtain these statistics, a suitable framework must be established. We consider *SEL*s distributed randomly in the unit square, and each having the same broadcast range, r, $0 < r \leq 1$. Thus, the leadership protocol structure is a function of the spatial distribution and density, and the broadcast range. Figures 4 and 5 show for various values of r (1, 0.707, 0.5, 0.25, 0.1, 0.05 and 0.01) the average number of clusters per number of *SEL*s (10 to 100).

The simulation protocol for a given number, n, of *SEL*s and broadcast radius r, is as follows: (1) a trial consists of the generation of 200 random layouts for the *SEL*s and the execution of the SNL protocol for each layout; the mean number of leaders is then computed for these 200 results; (2) 20 trials are run and the mean and variance computed for the 20 trials. As a verification check that the data is correct, the average node degree is calculated and shown to grow linearly with the number of *SEL*s. No error bars are shown for the average number of leaders since the 95% confidence interval is about 0.001; thus, confidence is high for a narrow spread about the mean.

As can be seen in Figures 4 and 5, the number of *leaders* (and therefore clusters) approaches a limiting value for the

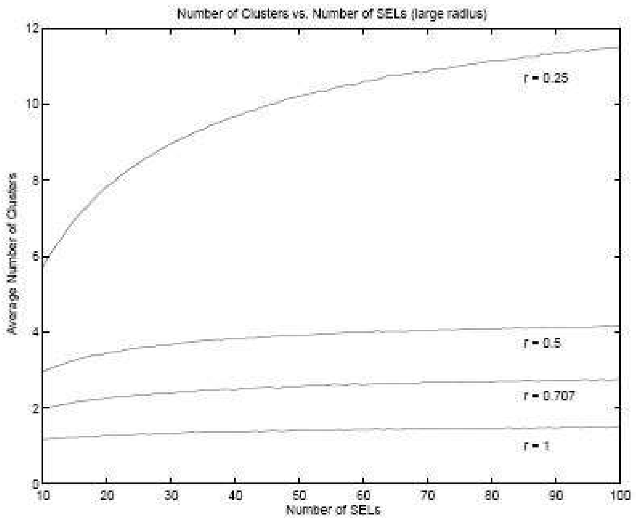

Fig. 4. Average Number of Clusters vs. Number of *SEL*s in Network.

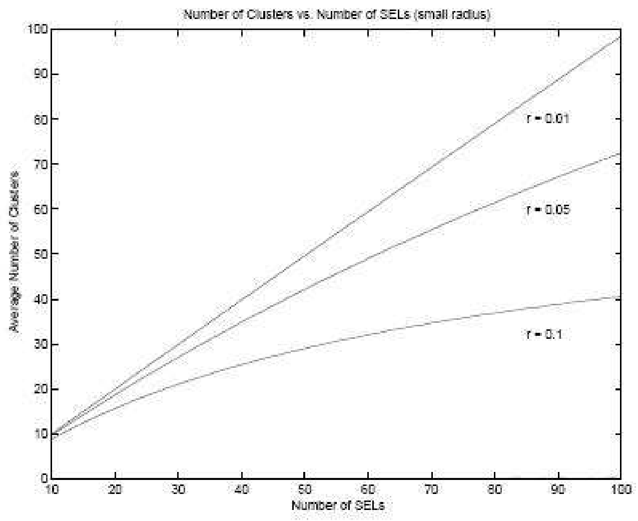

Fig. 5. Average Number of Clusters vs. Number of *SEL*s in Network.

larger radii, but continues to grow through 100 *SEL*s for the smaller radii. Some interesting questions are: (1) What is the maximum number of leaders possible? and (2) Does the average approach the maximum as the number of *SEL*s goes to infinity?

The first question can be posed as a circle packing problem (see [7], [8] for a good introduction to circle packing). The best solutions for packing up to 200 circles into the unit square are given in Table 13.1 in [8]; we give a selected subset in Table 2 here.

N	*Radius*
2	0.292893218813
3	0.254333095030
4	0.250000000000
5	0.207106781187
10	0.148204322565
64	0.063458986813
100	0.051401071774
196	0.036583075322

Table 2. Radius for Packing N Circles in the Unit Square.

Consider the SNL problem with circular broadcast range inside the circle of radius r:

1) The *SEL* location serves as the center of the broadcast circle, and thus all centers of the circles must be in the unit square. However, part of the circle may extend beyond the square.
2) No two leaders may directly communicate, and the minimum distance between leaders is r.

Consider the case of 4 *SEL*s, one at each corner of the square and $r = 1$. For this case, 4 is the maximum number of *SEL*s possible. Note that Figure 4 shows that the average number of clusters for $r = 1$ is about 1.5. The maximal case can only be achieved if *SEL*s are placed on or near the optimal coordinates and if the *SEL* IDs are appropriate.

To convert the SNL problem to a circle packing problem, the following steps are required:

1) In a circle packing problem, the circles are not allowed to overlap; therefore, circles of radius $r = 2$ must be used.
2) For the radius $r = 2$, the square of side $1 + r$ contains all broadcast ranges of possible *SEL*s with centers in the unit square.

These two requirements lead to a scaling from the SNL radius, r_{SNL}, to a standard circle packing radius, r_{pack}:

$$r_{pack} = r/(2(1 + r))$$

This yields the following process to determine the maximal (or upper bound on the) number of leaders (clusters) possible for a given radius, r: Determine upper bound for number of leaders:

```
Compute $r_{pack} = r/(2(1 + r))$.
Find where $r_{pack}$ falls in the
    Best Known Packing Results Table.
```

A Matlab function has been developed which calculates this number.

Table 3 summarizes the results found for the set of radii considered previously:

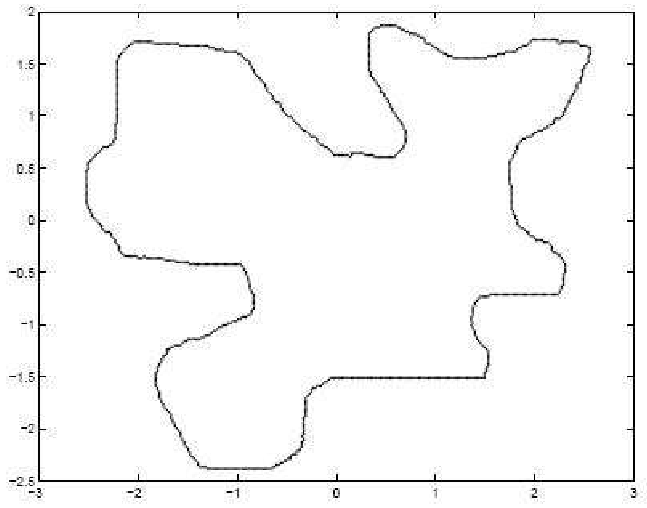
Fig. 6. Approximation of Berkeley Mote Broadcast Shape.

$r_N SL$	r_{pack}	*Upper Bound*	*Average*
1.000	0.2500	4	1.5
0.707	0.2071	6	2.5
0.500	0.1667	10	4.0
0.250	0.1000	25	12.0
0.100	0.0455	129	?
0.050	0.0238	1,849	?
0.010	0.0050	41,209	?

Table 3. Upper Bound and SNL Average Cluster Size for Various Radii.

Of course, it would also be interesting to find a leadership protocol that was equivalent to covering the unit square (see [5]) since this would require the minimum number of leaders, but at the moment, this seems to be a complex computation.

E. Irregular Broadcast Region Shape

The results given previously assume a circular broadcast area, centered at the *SEL*. Ganesan has shown [2] that physical motes do not broadcast this way. Thus, we must examine how irregular broadcast shape influences the statistics determined above.

Using the data given by Ganesan et al. as the basis for a broadcast shape, the statistics for mean number of clusters was recomputed. Figure 6 shows the shape used as an approximation of the Berkeley mote's broadcast shape. A 271x336 array holds the characteristic function of the shape (i.e., 1 where the shape is, and 0 otherwise). These are scaled by 0.0194 in order to obtain a 5.2644x6.5270 unit rectangle so that the shape has area 4 (equivalent area to a circle with radius 2). Two *SEL*s are broadcast neighbors if the broadcast shape of each overlaps the location of the other. The orientations of these broadcast shapes are random across the *SEL*s.

Figure 7 shows the mean number of clusters for various numbers of motes randomly distributed in a 6x6 square. As can be seen, the average number of clusters approaches 8 as N grows larger.

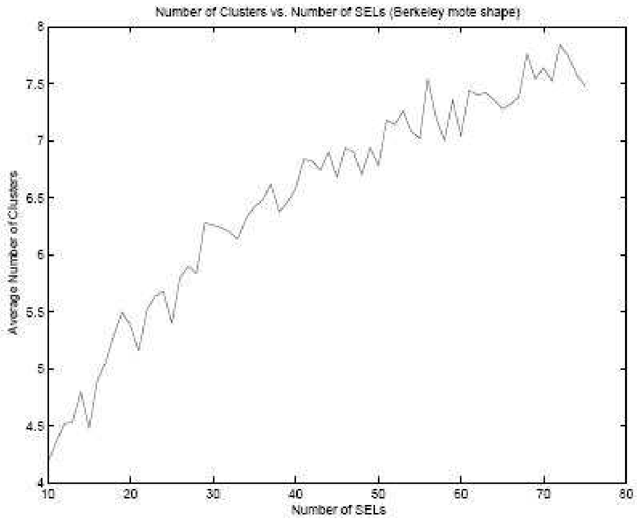

Fig. 7. Average Number of Clusters vs. Number of *SEL*s in Network.

IV. SUMMARY AND CONCLUSIONS

The SNL protocol is optimal in that it requires only one broadcast per node. Moreover, the cluster statistics are interesting and provide an upper bound on the number of leaders in a square area.

Initial results of actual implementations of the *S-Net* algorithms are very encouraging. The leadership protocol algorithm is the basis for most of the other algorithms we are implementing; e.g., coordinate frames, gradient calculation, reaction-diffusion, and level set calculations.

REFERENCES

[1] H. Chan and A. Perrig. Ace: An emergent algorithm for highly uniform cluster formation. In *Proceedings of First European Workshop on Wireless Sensor Networks*, Berlin, Germany, January 2004.

[2] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly resilient, energy efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review*, 1, 2002.

[3] Thomas C. Henderson. Leadership protocol for s-nets. In *Proceedings IEEE Conference on Multisensor Fusion and Integration*, Baden-Baden, Germany, August 2001.

[4] N. Lynch. *Distributed Algorihtms*. Morgan Kaufmann Pub, San Francisco, 1996.

[5] K.J. Nurmela and P.R.J. Ostergard. Covering a square with up to 30 equal circles. Technical Report HUT-TCS-A62 A62, Helsinki University of Technology, 2000.

[6] K. Shin, A. Abraham, and S.Y. Han. Self-organizing sensor networks using intelligent clustering. In *LNCS Proceedings of the Workshp on Ubiquitous Web systems and Intelligence*, Berlin, Germany, 2006. Springer.

[7] K. Stephenson. *Introduction to Circle Packing*. Cambridge University Press, New York, NY, 2005.

[8] P.G. Szabo, M. C. Markot, T. Csendes, E. Specht, L.G. Casado, and I. Garcia. *New Approaches to Circle Packing in a Square*. Springer, New York, NY, 2007.

[9] F. Zhao and L. Guibas. *Wireless Sensor Networks*. Elsevier Press, Amsterdam, The Netherlands, 2004.