A Robust Amorphous Hierarchy from Persistent Nodes

Jacob Beal

April, 2003

Abstract

For a very large network deployed in space with only nearby nodes able to talk to each other, we want to do tasks like robust routing and data storage. One way to organize the network is via a hierarchy, but hierarchies often have a few critical nodes whose death can disrupt organization over long distances. I address this with a system of distributed aggregates called Persistent Nodes, such that spatially local failures disrupt the hierarchy in an area proportional to the diameter of the failure. I describe and analyze this system, which has been implemented in simulation.

1 Introduction

There are many ways to build hierarchies; why does the world need another one, and what is special about the one which I am building? In a word: stability.

In a world of pervasive computing, we need systems that will organize themselves without human interference, and maintain a reasonably stable organization despite failures — possibly even in very large networks and during very large failures. Hierarchies are often a useful tool for organizing, but generally have critical nodes whose disruption can produce long-range effects.

In this paper, I present an algorithm that contains the effects of failures to an area proportional to the size of the failure. This algorithm expects to run in a *geographically local* network — a network embedded in space, with only short-distance links (e.g. an ad-hoc wireless network with limited transmission range). First I lay out my networking model and a motivation in the form of a real-world application example. Next I describe the PNHierarchy algorithm. Finally, I show the response of the algorithm to localized stopping failures, and present brief experimental results demonstrating that PNHierarchy works in simulation.

2 Amorphous Computing Model

One useful model for studying geographically local networks is two-dimensional amorphous computing.[1] An *amorphous network* is generated by distributing n particles uniformly randomly over a unit square (or other 2D region) and placing edges between all pairs of particles less than distance d apart. The resulting graph has spatially local connectivity but no long-distance links. The *amorphous matrix* for a given network refers to the spatial embedding of the network graph.

Each particle has a clock which drifts at a rate $|r| \leq \epsilon$ per tick, yielding timing uncertainty $L = \frac{1+\epsilon}{1-\epsilon}$. On each particle, we run an identical partially synchronous algorithm, subject to the following restrictions:

1. Particles receive no geometric information or coordinates
2. Particles receive no time synchronization information
3. Each particle's clock is initialized to a random time

Any of a wide variety of communication models may be applied. For purposes of this paper, I assume send/receive communication over links with reliable message delivery in maximum time t_m.

2.1 Application: Gigascale Wireless Networks

A real life example of an amorphous network is dense urban peer-to-peer wireless systems. For example, consider the city of Boston a few years from now, when every building has several wireless base-stations. If the base-stations communicate directly with nearby base-stations, then the city forms a geographically local network with approximately 10^7 nodes and a diameter of around 10^3 hops. If, in addition, we assume a pervasive computing scenario where most electrical devices communicate via wireless as well, then the density of the network skyrockets to on the order of 10^{10} nodes. Unlike most ad-hoc networking scenarios, there are so many more nodes than people in this scenario that it is safe to assume almost all of the nodes are stationary almost all of the time.

Controlling this sort of gigascale wireless network presents serious challenges. A robust decentralized hierarchy generation system such as the one described in this paper can be used to address problems of naming and routing without making strong assumptions of global coordinate or time synchronization systems. Another application is to use the tree created by the hierarchy as a storage data structure, in which case the bounded response to failure can be used to address atomicity problems.

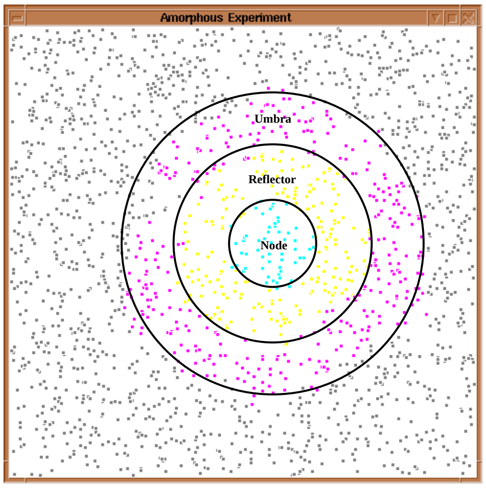


Figure 1: Anatomy of a persistent node. The innermost circle is the node proper (blue), where data can be read or written. Every particle within middle circle is in the reflector (yellow), which holds data and calculates which direction the node will move. The outermost circle (red) is the umbra (k=3 in this example), which knows the identity of the node, but nothing else.

3 The PNHierarchy Algorithm

The PNHierarchy algorithm is a fairly straightforward application of Persistent Nodes.[4] The key idea is to use Persistent Nodes with a variant congestion control heuristic to partition space evenly. We run an independent partition for each level of the hierarchy, with coarser granularity for higher levels, and wire the partitions into a hierarchy induced by the node centers.

From this simple procedure, we produce a surprisingly stable and robust hierarchical partition of space. The robustness properties of Persistent Nodes guarantee that the space partitions always rapidly converges toward a consistent state, and hysteresis in the hierarchy creation process enables the higher levels of the hierarchy to change extremely infrequently. Furthermore, failures cause noticeable effects only in a region proportional to the diameter of the failed region, so local failures do not cause global re-organization.

In the following sections, I will describe in detail the PNHierarchy algorithm. First, I review some properties and terminology of Persistent Nodes. Next, I describe how to produce a space partition using Persistent Nodes. Finally, I describe how a hierarchy is produced from a set of space partitions, and analyze properties of the hierarchy produced.

3.1 A Brief Review of Persistent Nodes

Most of the functionality of Persistent Nodes is actually not used by PNHierarchy: this section will sketch only their anatomy and movement. For an in-depth treatment of the PersistentNode algorithm, see [4].

Persistent Nodes, as used by the PNHierarchy algorithm, are named spherical regions of a geographically local network. A Persistent Node N begins life centered at a single particle and expands to incorporate all the particles within r hops. Thereafter, it slowly shifts its center according to a potential function Φ, moving N through the amorphous matrix.

The particles near a Persistent Node are grouped into three sets based on the estimated distance to the center of the node $h_N(p)$. Particles with $h_N(p) < r$ are in the *node* proper. Particles less than $2r$ are in the *reflector* — the region which controls movement direction. Particles within kr are in the *umbra*, the informational "shadow" of particles that know about node N.[1] (See Figure 1)

[1] In [4] $k = 3$. In this paper, however, we adjust k up to 8. Since particles in the umbra play no computational role in the node, this does not change any results from [4].

3.2 Partitioning Space

The space partitioning process, $PNPartition(r, \rho, D)$ is a subprocess of three parameters invoked by the PNHIERARCHY algorithm. The three parameters specify the node radius r to be used, and the estimated density ρ and dimensionality D of the amorphous matrix on which the $PNPartition$ process is running. The latter two parameters are assumed: they can be calculated once for a given matrix and used for all $PNPartition$ processes on that matrix.

The goal of the $PNPartition$ process is assign every particle to a group G_i such that the groups have diameter $O(r)$ and the diameter of the biggest group is only a constant factor bigger than the smallest group (We obtain bounds $7r > diam(G_i) \geq r$). The $PNPartition$ process achieves this by generating a population of Persistent Nodes and spreading them evenly through the matrix such that no two nodes are too close together and no particle is too far from any node.

Nodes are spread evenly by a potential function Φ which causes nodes in the same partition process to repel each other. For a node N, let l be the number of reflectors for $PNPartition(r)$ nodes[2] other than N to which a particle belongs, and let $K(p) = \sum_{i \in children(p)} v(i)$, the sum of heuristic values at the children of the particle. Then we define the potential function piecewise as:

$$\Phi_N(p) = \begin{cases} (K(p)-1)/(l(K(P)+1)) & (l \geq 0) \\ 1 & (l=0) \end{cases}$$

This function produces a rough approximation of repulsion between charged particles, tending to spread the nodes evenly through space. It only remains, then, to see that there are not too many or too few nodes.

Overpopulation of nodes is controlled by killing off a node whenever two nodes collide. When a particle is a member of two nodes, it invokes a $KILL$ operation on the node with the lesser ID. This gives us a simple constraint: no two node centers can be closer than $2r$ hops.

Under-population of nodes is controlled by randomly seeding new nodes in areas distant from any node center. Every particle running process $PNPartition(r, \rho, D)$ which is at least $2.5r$ from the nearest node center has probability P_c at every round of creating a new node with a random name. This probability is set so that an unpopulated matrix volume of radius $1.5r$ is expected to create one new node during $1.5r$ rounds. The radius is set to $1.5r$ so that the initial population will be spread rather than packed tightly, and the time-stretch allows time for the construction of a new node to inhibit creation of other new nodes. Approximating the volume as an n-cube, we thus set the creation rate to:

[2]This specifically excludes nodes used by all other processes, including partition processes with different values of r, in order to preserve independence between partitionings.

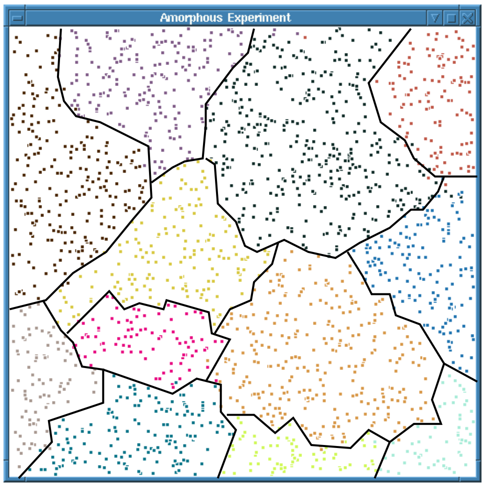


Figure 2: $PNPartition$ run with $r = 4$ on a matrix of 2000 nodes and communication radius 0.04. Thick black lines show the approximate boundary between groups.

$$P_c = \frac{2}{\rho(3r)^{D+1}}$$

From this population of nodes, every particle can is assigned to a group G_i affiliated with a particular node, or the unassigned group G_0. Let R_p be the set of nodes N for which $h_N(p) < 3.5r$. At each step, then, let P_p be the current assignment of p and N_p be the nearest node in R_p or G_0 if R_p is empty. We then set P_p to be N_p if N_p is G_0 or if $h_{N_p}(p) + r < h_{P_p}(r)$. This produces an assignment to closest node, with hysteresis, forming a partition of the particles into groups. Note that the subnet for a group is not necessarily connected, though its embedding in the matrix is (except for G_0).

3.2.1 Properties of $PNPartition$

Here I will examine some of the key properties of the $PNPartition$ method. In general, I will assume that density and dimensionality are consistent throughout the matrix, and consider only interior regions (boundary cases complicate the analysis greatly and only effect the constants).

First, recall that the PERSISTENTNODE algorithm is expected to move a node's center at most every $4r/L$ rounds in the failure-free case. This extremely slow rate of motion means that the velocity distortion of a node may be disregarded.

As a consequence, particles go a quadratic period between reassignments. For a particle p which has just joined $G_i \neq G_0$, it must be the case that no node N_j is closer than N_i. If N_i moves away and some other N_j moves toward p at maximum velocity, then $r/2$ moves of their centers are required to change p's group to G_j. Each move takes $4r/L$ rounds, so at least $2r^2/L$ rounds elapse between reassignments of a given particle p. A collision between nodes can reassign a particle faster, but this is an exception case which happens even more infrequently due to the load-balancing effect of the potential function.

The case of a node joining G_0 must be dealt with separately. That event is unlikely once the matrix has reached steady state, and cannot persist for long. We will start by analyzing how long a particle can remain in G_0. Consider a particle p which joins G_0: this particle has $h_N(p) \geq 3.5r$ for every node N, which means that every particle within r hops is eligible to create a new node. Assuming a two-dimensional matrix, this means that there are $\frac{2}{\rho(3r)^3}\rho\pi r^2 = \frac{2\pi}{27r}$ nodes expected to be created in the region each round that p remains unlabelled. In the time for a node to move one step, then, there is an expectation of at least $\frac{8\pi}{27L} \approx 0.93/L$ nodes created — so some node will be created in $O(rL)$ rounds with high probability. Note that this analysis covers the startup transient as well.

Once a node is created in this region, p will be within r of the new node, and therefor unable to join G_0 again for at least $10r^2/L$ rounds. Moreover, the likelihood of p joining G_0 again at all is extremely small for large r: for the last r^2/L rounds before p joins G_0, there must be a region of radius $r/2$ particles eligible for node creation around p, which are expected to produce $\frac{2}{\rho(3r)^3}\rho\pi\frac{r^2}{4} = \frac{\pi}{54r}$ nodes per round, or an expected $\frac{\pi}{54L}r$ nodes. The probability that no node is generated during this time, by the Poisson distribution, is $e^{-\frac{\pi}{54L}r}$, which quickly drops toward zero with rising r.

In practice performance will generally be much better than these time bounds, as the movement of nodes more closely approximates the random walk of particles in a Brownian gas than the worst-case linear motions used in this analysis. I will not, however, analyze this further in this paper.

Finally, note that the number of nodes is proportional to the area, and that the diameter of the set of particles in a node is bounded above and below: for any node N, its diameter $diam(N) < 7r$ because only particles within $3.5r$ hops of the center can be labelled N. Conversely there is a lower bound of $diam(N) \geq r$ for an interior node,[3] because no two node centers can be closer than $2r$, so with a hysteresis of magnitude r every particle within $r/2$ hops of the center of N_i must be belong to G_i.

3.3 Generating a Hierarchy

Using the *PNPartition* process, we can generate a space partitioning for any given characteristic radius. We use this to generate a set of $\log(diam)$ space partitions and form logical connection between layers such that we have a hierarchy that covers the entire area. The partitions are run with exponentially increasing node sizes: level zero is simply the individual particles; the first level uses nodes of radius $r_1 = 2$, and each level thereafter doubles in radius (e.g. level 10 would have radius $r_{10} = 1024$ nodes[4]).

There are two issues in building a hierarchy from these exponentially increasing partitions: how do we determine how many partitions we need ($\log(diam)$) and how do we connect successive partitions in order to form a hierarchy?

3.3.1 Finding $\log(diam)$

The simplest way to find $\log(diam)$ would be to simply poll for the diameter. However, failures and additions in the matrix could radically change this during run-time, so we need a cheap dynamic calculation instead. For this, we use the fact that the top of a hierarchy is a single root — so therefor the top level should have a single node large enough that no other node can be created at that level. Thus, we have a constraint that precisely one partition, the top level, should have one node.

To test this constraint, the center of every node polls for neighbors at an expected rate of once every kr rounds.[5] If any particle in the node is in the umbra of another node, then there are neighbors. The result of the poll is broadcast throughout the umbra, allowing particles to adjust their estimate of $\log(diam)$: if every existing level has neighbors, a new level is added; if some level has no neighbors, then that level is set to be the top level. In practice, this means that

[3] A node near the edge of the matrix may, of course, be constricted further by lack of space.

[4] I will use r_i as shorthand for the radius of ith level nodes, 2^i.

[5] Experiments used the arbitrary value k=10.

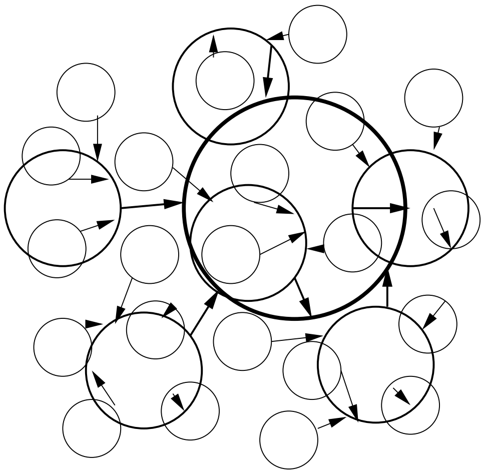


Figure 3: A hierarchy of three levels of nodes. Larger circles are higher-level nodes, and arrows show parentage relations between a nodes.

we start with a single partition, which then builds upwards as successive partitions become populated.

3.3.2 From Partitions to Hierarchy

This is actually a very simple process: at the base of the hierarchy, every particle is a leaf. The parent of each leaf (particle) is its label in the first partition. To recurse upward, the particle at the center of a level i node N_i chooses its $i+1$ parent to be its group G_{i+1} in the level $i+1$ partition, with r_{i+1} hops of hysteresis.[6] It then broadcasts its parent and every particle with ith parent G_i takes that for its $i+1$ parent.

This procedure clearly generates a hierarchy: all particles with ith parent G_i will adopt the same $i+1$st parent G_{i+1}, every particle is a unique leaf at the base level (by assumption), and every particle has the same root at some level $O(\log(diam))$ (when there is a single node whose group covers all space). Finally, there can never be a naming collision, because if two nodes with the same name adopt the same parent, then they must be close enough for their umbras to touch, which will cause the underlying PERSISTENTNODE algorithm to delete one of the conflicting nodes.

3.3.3 Properties of PNHIERARCHY

Distance/Address Relationship Particle p_a is a neighbor of p_b at level i if p_a is in the umbra of p_b's ith parent node, N_i. We can bound neighbor relations with the distance between the particles $d(p_a, p_b)$, and vice versa.

If p_a is a level i neighbor of p_b, that means it is within $8r_i$ hops of the center of N_i. The center of N_i is at most $3.5r_i$ hops from the center of N_{i-1}, which is at most $3.5r_{i-1}$ hops from the center of N_{i-2} and so on. Summing all the distances, we obtain a maximum distance of $15r_i = 15{*}2^i$ between two nodes which are neighbors at level i. Conversely, $d(p_a, p_b)$ bounds the minimum level at which the two particles can become neighbors: no earlier than level $i = \lceil \lg \frac{d(p_a,p_b)}{15} \rceil$.

We can also obtain tidy minimum distance and maximum level bounds. For two nodes which are neighbors at level i, but not at any level $j < i$, they cannot be any closer than $\frac{r_i}{2} = 2^{i-1}$ hops distant from each other, or else they would be neighbors at $i-1$. Conversely, for two nodes at distance $d(p_a, p_b)$, they must be neighbors at every level $i \geq \lceil \lg d(p_a, p_b) \rceil$.

Construction Time Starting from scratch, PNHIERARCHY constructs a hierarchy in $O(diam \cdot L)$ time: the level partitions are constructed serially, with level i taking $O(r_i L)$ rounds to construct. This is dominated by the last level, which has radius $O(diam)$.

Communication Cost For communication cost, we will consider *communication density* rather than message cost. Communication density measures the amount of bandwidth consumed per particle: a particle in a network with n particles which multicasts a k-word message to its immediate neighbors costs k/n.

The communication cost for PNHIERARCHY is $O(\log(diam))$ density: there are three components to the cost — cost of running the *PNPartition* processes, cost of polling for neighbors, and cost of maintaining the hierarchy tree. A *PNPartition* process costs based on the number of node umbras that a particle is participating in, which is limited to $O(1)$ by sphere packing.[7] The converge-cast used to poll for neighbors can be extremely costly, up to $O(r^3)$, but since the poll is only expected to occur once per node in every kr rounds, the final density is only $O(1)$ per level. Finally, transmitting parent information throughout a node costs by number of nodes, just as *PNPartition*, so maintaining the hierarchy costs $O(1)$ per level. So the cost for each level is $O(1)$ and there are $O(\log(diam))$ levels, yielding a final communication density of $O(\log(diam))$.

[6] i.e. if G_{i+1} is not its old parent, it keeps its old parent unless G_{i+1} is 2^{i+1} hops closer.

[7] This does not hold for an arbitrary graph, but does for spatially embedded networks such as in an amorphous matrix.

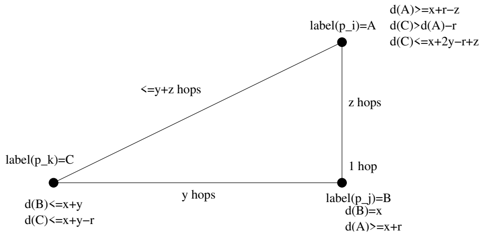


Figure 4: Displacement between three successive changes in parentage (A,B, and C) is at least $r_{i+1}/2$ for a static set of nodes. Each transition involves a differential of r_{i+1} hops distance, due to hysteresis in the hierarchy assignment process, but the distance between the first and third transitions must be high enough that that the first transition point can be labelled A.

Stability of Hierarchy Particles change parents slowly, and large-scale changes happen extremely infrequently. The parents of an individual particle p change every time they adopt a different level 1 parent: this class of change, however, is cheap and tightly localized, no matter how many parents of p are different after the change. When a level i node adopts a new $i+1$ parent, however, the region which must be updated is much larger, affecting $O(r_i^2)$ particles. Accordingly, we want changes at the bottom levels of the hierarchy to dominate.

To see that this is the case, consider the parentage of a level i node in the hierarchy tree. The parent is determined by the level i and level $i+1$ partitions: labels from level $i+1$ and the locus of decision at the (shifting) center of the level i node. The label of a particle in $PNPartition$ keeps the same label for at least $2r^2/L$ rounds, but when the center of the level i node moves, it may be to a particle with a different label. The hysteresis in choosing labels, however, throttles the rate at which changing labels can change the hierarchy, because the distance must be instantaneously better as well.

For a static arrangement of level $i+1$ nodes and a level i node moving through the matrix, the node's center is displaced an average of at least $\frac{r_{i+1}}{4}$ hops between changes of parentage. This is because each change in label requires a differential of r_{i+1} hops. Consider three consecutive changes of parentage for a node N_i: to A to B and to C, occurring at particles p_i, p_j, and p_k. At p_j, the distance to B is $d_j(B) = x$, and the distances to p_k and p_i are y and z respectively. Hysteresis then implies $d_j(A) \geq x + r_{i+1}$, and $d_k(C) \leq x + y - r_{i+1}$. Translating these distances to p_i gives $d_i(A) \geq x + r_{i+1} - z$ and $d_i(C) \leq x + 2y - r_{i+1} + z$. Finally, since p_i has label A, we know that $d_i(C) > d_i(A) - r_{i+1}$ assuming $A \neq C$[8]. Combining these three constraints and solving for $(y+z)$ yields $(y+z) > r_{i+1}/2$: thus every pair of transitions is separated by movement of at least $r_{i+1}/2$ hops, yielding an average bound of $r_{i+1}/4$.

Translating this to the dynamic case, where the $i+1$ nodes are moving translates the distance bound into a relative distance bound, leaving us still with an expected time of $\Omega(r_{i+1}^2/L) = \Omega(2^{2i}/L)$ rounds between changes of the hierarchy at level i. As desired, the low-level changes dominate.

4 Recovery From Stopping Failures

The real desirability of the PNHIERARCHY algorithm is that stopping failures only disrupt an region linear in their diameter. In effect, the natural fluidity of the hierarchy provides a "noise floor", and a stopping failure causes disruption above this noise floor only in a local area.

To investigate this, we will consider a stopping failure F of diameter f (i.e. a set of failing nodes where the distance between any two is at most f hops on the amorphous network). How much effect will F have on the hierarchy tree?

First, note that the hierarchy is, in fact, guaranteed to converge: the hierarchy is derived from the partition, which is based on Persistent Node constructs, which will either be completely destroyed or reconstructed within $3Lr$ rounds of a failure. So within $O(diam \cdot L)$ rounds some hierarchy will have been reconstructed. This is a very loose bound, however, and fails to address the question of how different the new hierarchy will be from the old hierarchy.

A stopping failure can cause a change in the parentage of a node at level i if some level i node ends up with its center having a different partition label which overcomes hysteresis. This can be effected by some combination of shifting the center of a node at level i, and shifting or destroying level $i+1$ nodes such that the labels of particles are changed. Note that either way, the critical operation is moving the center of a level i node.

Now here is a problem: even a single particle failure might cause an arbitrarily large change, if the particle which fails is the center of a level i node and its death shifts the center across a hysteresis boundary. Salvation comes from the fact that this happens very

[8]If $A = C$, then $z \geq r_{i+1}$, or hysteresis would prevent the transition back to A.

rarely — rarely enough, in fact, that we will be able to shift the effect under the noise floor for high levels.

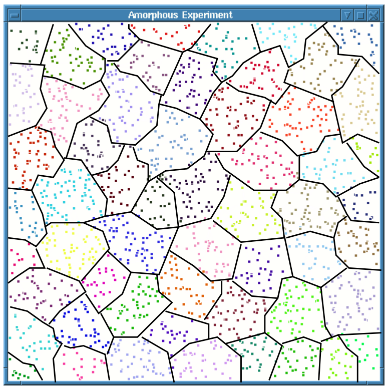
Recall that in the non-failing case, a node's center must have a relative motion of at least 2^{i-1} hops between three successive transitions. A stopping failure of diameter f, then, simply introduces an instantaneous additional displacement of up to πf hops, which can only cause a level i change in hierarchy if the relative motion since the next-to-last transition is at least $2^{i-1} - \pi f$ — in other words, in a small window of vulnerability when a change could happen shortly anyway.[9] When 2^i is large, then, the relative size of the vulnerability can be bounded to arbitrarily small fraction ϵ, below which the changes caused by the failure are considered indistinguishable from failure-free hierarchy changes.

Thus a stopping failure of diameter f causes distinguishable changes in only the bottom $O(\log f)$ levels of the hierarchy. Additionally, since these changes involve only nodes of radius $O(f)$, the hierarchy will converge again within $O(f \cdot L)$ rounds, having affected only particles within $O(f)$ hops of the region of failure.

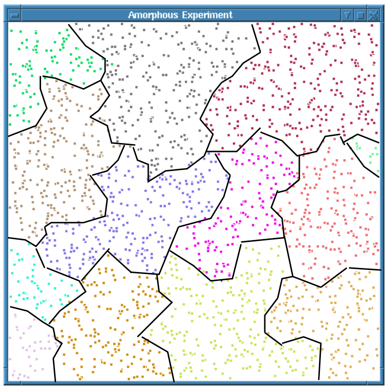
5 Experiments

I have implemented the PNHierarchy algorithm and tested that it behaves as expected in an amorphous matrix with 2000 particles and a communication radius of 0.04 (i.e. approximately 40 hops in diameter). As expected, the nodes converge rapidly to a hierarchy which changes very slowly, and shows disruption linear in the diameter of stopping failures. The results of a typical run of PNHierarchy are show in Figure 5. The hierarchy tree structure for this run is shown in 6.

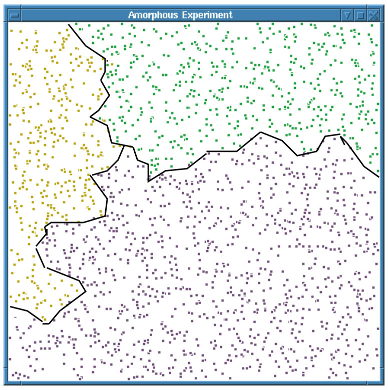
My greatest frustration while running experiments to confirm the results in this paper was that the implemented version of PNHierarchy could not be persuaded to misbehave badly enough to approach the bounds showed in this paper. Case in point is the frequency of transitions in the hierarchy. The following table shows the results of running PNHierarchy for 300,000 rounds:



(a) Level 1



(b) Level 2



(c) Level 3

Figure 5: A typical run of PNHierarchy for 2000 particles with communication radius 0.04. There are five levels in the resulting hierarchy. The top and bottom levels of the hierarchy are uninteresting, as the top has every particle in the same group and the bottom has every particle in a different group; these images show the middle three levels of the hierarchy, with each ith level node a different color, and thick black lines showing the approximate boundaries. The logical tree is shown in Figure 6.

[9]It could also cause a collision in a window of vulnerability similarly bounded, involving immediate level i neighbors as well and increasing disruption by no more than a constant factor.

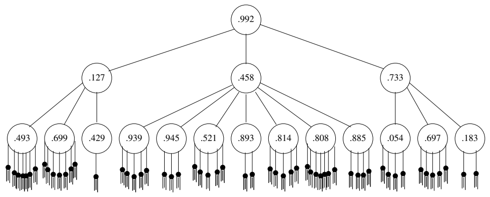


Figure 6: The hierarchy tree produced by the run of PNHierarchy shown in Figure 5. The random numbers are the names of nodes in the top three levels; the level 1 nodes are shown as black dots and the 2000 leaf nodes are not explicitly enumerated.

Level	Transitions	$\frac{300000}{\lceil 2^{2i} \rceil}$
0	276222	300000
1	3140	75000
2	570	18750
3	0	4688
4	0	1172

The hierarchy produced has four levels, though no changes occurred in the top two levels after creation — they were too stable. Unfortunately, this also means that there is not enough data to satisfactorily verify the 2^{2i} bound predicted in Section 3.3.3, though the results are, in fact, well within the predicted range. As expected, however, the relabellings of the bottom nodes vastly outweighed the relabellings of the higher level nodes.

Determining the range of disruption also turned out to be difficult. Many failures produce very little disruption, and only by targeting failures at the centers of nodes was it possible to obtain long-range effects. The following table shows the effects of sample targeted disruptions of different diameters:

$diam(f)$	Disruption (hops)	Time (rounds)
1	0	N/A
2	1	15
3	0	N/A
4	6	45
5	4	30
6	7	60
7	7	45
9	14	100
10	18	140

As expected, the data is roughly linear, but the constant is smaller than might be expected, evidence of the difficulty in producing maximal disruptions, likely due to effects of hysteresis and collision-avoidance.

6 Contributions

The PNHierarchy algorithm uses Persistent Nodes to build a rapidly converging hierarchical partition of space. The hierarchy produced changes exponentially slowly with height, producing a "noise floor" of expected changes. Stopping failures produce disruption above the "noise floor" only within a region linear in the diameter of the failure, and reconvergence takes place within linear time. Experiments in simulation confirm that the algorithm performs as expected.

This hierarchical partitioning can be applied as a basis for routing, naming, or data storage in gigascale localized networks, such as metropolitan ad-hoc peer-to-peer wireless networks.

6.1 Related Work

Previous work on hierarchy building in the Amorphous Computing model ([6] [11] [12]) has focused on construction of a hierarchy, without addressing maintenance or recovery from failure of critical nodes.

Much work has been done on hierarchy systems for geographically local networks in the context of packet radio, such as Tsuchiya's Landmark routing system[13], which can produce an unbounded radius change due to the death of a particle being used as a high level landmark, or Baker and Ephremides[2] LCA algorithm, which shows similar vulnerabilities when invoked recursively. More recently, ad-hoc networking algorithms have used clustering to produce hierarchical structures (as, for example, in [8] [7] [3] [9]) which similarly either assume networks small enough to use only a constant number of clustering levels or fail to address bounding of changes produced by small changes in topology.

7 Acknowledgements

Thank you to Tim Shepard for the idea of applying amorphous computing to gigascale wireless and the Boston scenario. Thanks also to Hal Abelson, Radhika Nagpal, and Gerry Sussman for editing and for challenging questions.

References

[1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Suss-

man and R. Weiss. Amorphous Computing. AI Memo 1665, August 1999.

[2] D.J. Baker and A. Ephremides, "The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm," IEEE Trans. Comm., vol. 29, no. 11, pp. 1694-1701, Nov. 1981.

[3] S. Basagni, "Distributed Clustering for Ad Hoc Networks," Proceedings of the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN), Perth, Western Australia, June 1999, pp. 310-315.

[4] Jacob Beal. Persistent Nodes for Reliable Memory in Geographically Local Networks. MIT AI Memo 2003-011.

[5] Daniel Coore. Establishing a Coordinate System on an Amorphous Computer. MIT Student Workshop on High Performance Computing, 1998.

[6] Daniel Coore, Radhika Nagpal and Ron Weiss. Paradigms for structure in an amorphous computer. MIT AI Memo 1614.

[7] B. Das, E. Sivakumar, and V. Bhargavan, "Routing in ad-hoc networks using a spine," in IEEE International Conference on Computer Communications and Networks '97, 1997.

[8] M. Gerla and T.-C. Tsai. Multicluster, mobile, multimedia radio network. Wireless Networks, pages 255–265, 1995.

[9] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. ACM SIGCOMM Computer Communication Review, pages 49–65, April 1997.

[10] Nancy Lynch. Distributed Algorithms. Morgan Kaufmann Publishers, Inc, San Francisco, California, 1996. Chapter 8, pages 199-234.

[11] Radhika Nagpal. Organizing a Global Coordinate System from Local Information on an Amorphous Computer. MIT AI Memo 1999

[12] Radhika Nagpal and Daniel Coore. An Algorithm for Group Formation in an Amorphous Computer. Intl Conf on Parallel and Distributed Computing Systems (PDCS'98), 1998

[13] Paul F. Tsuchiya. The landmark hierarchy: A new hierarchy for routing in very large networks. In *Proc. of the SIGCOMM '88 Symposium on Communications Architectures and Protocols*, Stanford, CA, August 1988.