Department of Electrical
and
Computer Systems Engineering

Technical Report
MECSE-2-2006

Virtual Localization for Routing in Sensor Mesh Networks

N. Moore, Y. A. Sekercioglu and G. K. Egan

MONASH
UNIVERSITY

Virtual Localization for Routing in Sensor Mesh Networks*

Nick Moore Y. Ahmet Şekercioğlu Gregory K. Egan

Centre for Telecommunications and Information Engineering,
Monash University, Melbourne, Australia

Abstract

We present a novel distributed 'virtual localization' method for non-mobile mesh networks, which does not depend on radio ranging or the existence of anchor nodes. We show in simulation that it can be used to rapidly establish efficient and reliable location-based routing information without excessive overheads.

INDEX TERMS: Wireless sensor networks, distributed networks, routing protocols.

1 Introduction

1.1 Sensor Networks

Miniaturized sensors which measure temperature, pressure, humidity and various chemical concentrations have become cost-effective to mass-produce in recent years. The data processing power required to compress and analyse the measurements has also become very cheap. These developments have enabled the widespread use of large numbers of sensors to collect environmental data.

Data however must still be collected and brought back for analysis. Where there are only a small number of sensor nodes, it is practical to collect this data by hand, or fit each node out with a powerful radio transmitter but where hundreds of nodes are required such as in large scale environmental sensing this rapidly becomes impractical.

Sensor networks avoid this scalability issue by putting nodes in communication with each other. Rather than gathering data from each node, data is relayed from node to node and gathered at a single point. The difficulty is in configuring the network of nodes: not every node can communicate directly with every other node, and the topology of the network may change as nodes move or fail, new nodes are added, or just as the environmental conditions change.

1.2 Mesh Sensor Networks

Mesh sensor networks get around the problem of network configuration by allowing the network to self-configure. Small, cheap sensor nodes (often called 'motes') collect environmental data and communicate with nearby nodes using low-power, short range wireless interfaces.

*Early results of this study were presented at the IASTED International Conference on Networks and Communication Systems (NCS 2005) [1]

When the density of nodes is sufficient, they form a mesh of network links, across which data can be transferred from node to node or back to a central repository.

Battery power is limited, and so computational power and network usage are precious resources. A mesh sensor node must preprocess and transmit its data as efficiently as possible, and this requires an efficient way to route communications from node to node.

Mesh networks have some general properties:

- All nodes are equal.
- All routing computation is distributed.
- Battery power is limited, and processing power and network usage are therefore expensive.

Unlike other kinds of ad-hoc / mesh networks, the nodes of a sensor mesh network are generally assumed not to move during the life of the network, so we have not considered mobility issues in this paper.

1.3 Routing Within a Mesh

The classic Internet routing strategy of hierarchical partitioning into sub-networks [2] is not appropriate for mesh networks. Mesh network topologies are generally not planned and may even be entirely ad-hoc. Imposing a hierarchical structure or a spanning tree over the mesh would be difficult and inefficient.

There are many possible methods for routing within meshes [3]. Most either require centralized control or a large amount of data to be 'flooded' across the network to let every node know about every other node. The extra traffic on the mesh network requires extra resources so more resource efficient strategies are desirable.

1.4 Location-based Routing

If the location of each node can be determined, 'Location-based Routing' algorithms can be used. The simplest of these methods, known as 'greedy forwarding', is for a node to forward packets to whichever of its neighbours is closest to the destination.

1.4.1 Greedy Forwarding

Figure 1 shows the operation of greedy forwarding. A packet starting at H is forwarded to whichever neighbour of H is closest to the destination G – in this case, E. This process continues as the packet is forwarded to F, M, D and finally G. Greedy forwarding does not always find the optimal path[1] but it generally produces a reasonably efficient route to the destination.

However, it is possible for packets forwarded by this method to end up blocked by 'voids', where an intermediate node is closer to the destination than any of its neighbours, and thus cannot determine where to forward the packet.

For example, in Figure 2, a packet travelling from N to R would be stuck at N, since neither P nor S are closer to R than N is. Sometimes, forwarding the packet to the closest

[1] path $\overrightarrow{HEFCG}$ would be shorter in both distance and number of hops, but F will forward the packet to M as M is closer to G than C is.

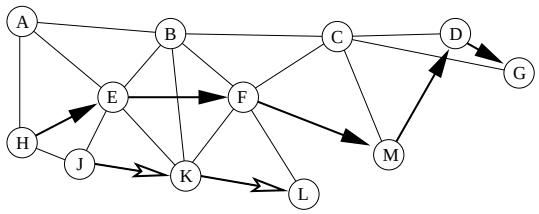


Figure 1: Greedy forwarding

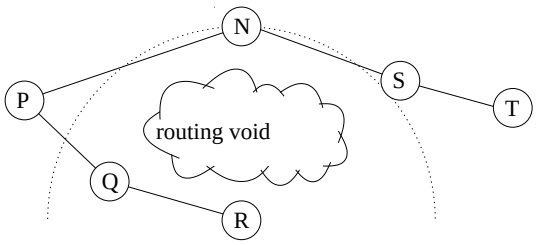


Figure 2: Greedy forwarding voids

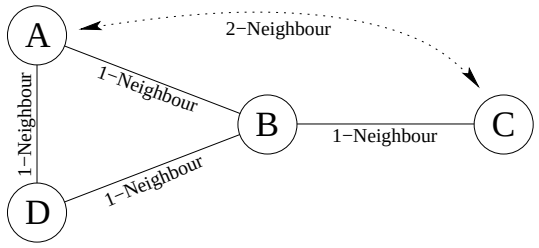


Figure 3: 1-neighbours and 2-neighbours

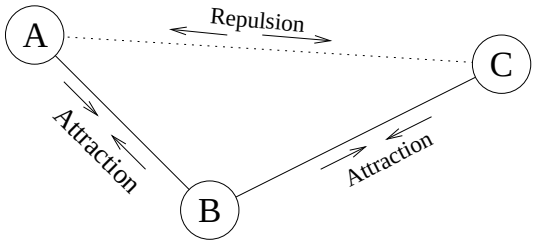


Figure 4: Simple Attraction/Repulsion Model

neighbour will be sufficient but in this case it is not since S will simply pass the packet back to N, S's closest neighbour to R, thus creating a routing loop. Unless there is some way to prevent routing loops, N must drop the packet.

Schemes such as GFG/GPSR [4, 5] and INR [6] have been developed to mitigate this problem by using alternative strategies when greedy forwarding fails.

1.4.2 n-Neighbours and n-Neighbourhoods

The '1-neighbours' of a node are the nodes it can communicate with directly. For example, in Figure 3, node A has 1-neighbours B and D.

A can also communicate with its '2-neighbour', C, via B. Because there are a minimum of two hops between A and C, C is called A's '2-neighbour'. The '2-neighbourhood' of a node consists of all nodes which are two or less hops from it.

More generally, node Y is an n-neighbour of node X if the smallest number of hops from X to Y is n, and the n-neighbourhood of X consists of all nodes which are n or less hops from X.

1.4.3 Supergreedy Forwarding

This paper concentrates on localization algorithms rather than routing algorithms, so rather than implementing sophisticated protocols, we have implemented a simple 'supergreedy' al-

gorithm, a trivial enhancement of Greedy Forwarding [7].

If a node has information about its 2-neighbours (or higher), it makes sense to use this information. Supergreedy Forwarding determines which of the node's known n-neighbours is closest to the packet's destination location, and tries to forward the packet via this intermediate neighbour.

Using this extra information allows Supergreedy Forwarding to find a way around many small voids, increasing performance substantially. For example, in Figure 2, if N knows of node Q from it's immediate neighbour P, packets travelling to R will be forwarded from N to P to Q to R.

Supergreedy Forwarding can prevent packets being blocked by small voids. As the extent of nodes' knowledge increases, larger and larger voids can be avoided. However, there are costs associated with the gathering of this information. Since the Virtual Localization algorithm requires nodes to have knowledge of their 2-neighbourhood, we can reuse this information for the purposes of Supergreedy Forwarding.

1.5 Determining Location

The routing algorithms discussed in Section 1.4 require all participating nodes to obtain location information. Where does this information come from? The naïve solution is to equip every node with a Global Positioning System (GPS) receiver or a similar out-of-band location determination method. In this way, each node knows its own position, and can communicate this to its neighbours for routing purposes. However, GPS receivers are still too large and expensive for use in tiny, ubiquitous nodes, and are inaccurate or unusable indoors.

Several papers suggest 'anchoring' networks with some percentage of nodes which can determine their location, and then determining the position of the other nodes by geometric constraints [8, 9], by iterative methods [10] or using Kalman Filters [11]. However, these methods require up to 20% of the population to be 'anchor' nodes, which establishes an undesirable hierarchy in our formerly egalitarian mesh.

1.6 Radio Ranging

Some methods of localization assume that nodes can measure their distance from their neighbours based on radio propagation – signal levels are assumed to attenuate as a function of distance. These measured distances are then used to triangulate a location. However, radio propagation in the real world is rarely so simple. Due to diffraction, scattering, reflection, refraction and attenuation by intervening materials [12], signal strength can not be considered a function of distance alone. Localization algorithms can assume a limited correlation with signal strength [13] at best.

1.7 Virtual Location

Location-based routing requires only relative location information, and if geographic location is not needed for other purposes, location can be decoupled from reality and a 'virtual location' determined instead. Anchor nodes can be dispensed with and geometric [14], constraint [15], iterative [16] or energy-minimization [17] methods can be used to find a situation which is internally consistent and thus usable for location-based routing algorithms.

Virtual locations are generally only useful for routing purposes as they do not correspond to geographic locations.

1.8 Mathematical Publications

Decades before engineers began considering ad-hoc and sensor mesh networks, mathematicians were seeking a way to present graph topologies neatly on paper. The language used by mathematicians is different to that used by engineers: for 'network', 'node' and 'link', substitute 'graph', 'vertex' and 'edge'. The process of assigning each vertex a location for plotting is referred to as 'graph embedding'.

Eades [18] presents a system to lay out graphs with "less than 30 vertices", using a simple physical attraction/repulsion model. Attraction is springlike, but logarithmic rather than linear, and repulsion from all nonadjacent vertices is inversely proportional with the square of the distance. At each iteration, forces on each node are calculated, and the position of each node updated proportional to this force.

Fructerman and Reingold [19] use a similar method with a less directly physical model and a number of optimizations to speed convergence.

Kamada and Kawai [20] delve more deeply into the mathematics, considering the differential equations of a linear spring model. Only one vertex moves at each iteration.

Davidson and Harel [21] use a quite different approach, which seeks to minimize a cost function. By stating the problem in this form, general optimization strategies such as 'simulated annealing' may be applied.

2 The Algorithm

The Virtual Localization algorithm presented here has been inspired by the location and graph embedding algorithms described in Sections 1.7 and 1.8 respectively.

2.1 Attraction/Repulsion Models

We will start describing the operation of the algorithm through a simple attraction/repulsion model shown in Figure 4. Masses A and C are attached to mass B by springs, and A and C are repelled from each other by an electrostatic-like force. A, B and C will converge along a straight line, with A and C equidistant from B, since this is the configuration with the minimum energy.

This emergent behaviour of the simple spring model is central to the Virtual Localization algorithm presented here. By minimizing the potential energy of each node, the overall potential energy of the network is minimized. The minimal energy for a node has its neighbours pulled close and non-neighbouring nodes pushed away, and this results in a network whose virtual geometry approximates its real geometry, enabling virtual location based routing.

2.2 Encoding Virtual Location

Our implementation of the Virtual Localization algorithm uses N 32-bit signed integers to represent a position in a bounded N-dimensional Euclidean space.

We locate the nodes of 2- and 3- dimensional networks in a 4-dimensional space. Using an N-dimensional space to solve problems with less than N dimensions may seem excessive, but it allows an extra degree of freedom to prevent nodes being trapped in local minima. This is discussed further in Section 4.1.

Only a small part of the range of these coordinates is ever used, and so it would be possible to pack this information into 64 bits or less and thus encode the virtual location information in an IPv6 prefix or suffix.

2.3 Beacons

Nodes must communicate their position estimates, and they do this by sending broadcast 'beacon' messages over their network interfaces. They do so periodically, with a random offset to avoid network congestion. Each beacon indicates the identity and current virtual location of the node, and the identities, hop distances and virtual locations of its 1-neighbours (and optionally 2-neighbours).

Communication between nodes is not instantaneous: they awake sporadically, recalculate their location and then transmit a beacon. Beacons received while sleeping are stored but not acted upon immediately. Thus, there may be discrepancies between the virtual location a node has recorded for a neighbour and the virtual location the neighbour would have if it recalculated at that instant.

Sending beacons frequently once the network has converged is wasteful of network resources: a more practical implementation would reduce the rate as the neighbourhood becomes less dynamic but this is not considered here.

2.4 Forces and Potentials

The Virtual Localization algorithm seeks to minimize the overall potential energy of the network by allowing each node to minimize its own potential energy. In this way, a global solution can be converged upon using only distributed processing and localized signalling.

1-neighbours have a 'spring-like' attraction:

$$A_{ij} = k_{att} \cdot d_{ij}^2 \tag{1}$$

where A_{ij} is the potential energy felt by the node i due to the node j, d_{ij} is the distance between them and k_{att} is a constant.

2-neighbours (and higher) have an 'electrostatic-like' repulsion, with a small offset to prevent infinities:

$$R_{ij} = k_{rep} \cdot \frac{1}{d_{ij} + 1} \tag{2}$$

where R_{ij} is the potential energy felt by the node i due to the node j, d_{ij} is the distance between them and k_{rep} is a constant.

The total potential energy for a node i, written U_i, is found by summing these potentials:

$$U_i = \sum_{j \in M_1} A_{ij} + \sum_{k \in M_2} R_{ik} \tag{3}$$

where M_1 and M_2 are the sets of 1- and 2-neighbours of which node i is aware.

The values of the constants k_{att} and k_{rep} have been chosen as 1 and 8×10^6 respectively, so that a network of three nodes in a line settles to an even spacing of 100. This is for convenience only: it keeps the coordinates human-readable, and allows the implementation to use mostly integer arithmetic.

2.5 Perturbation

A very simple iterative descent method is used: at each iteration, the node's virtual location is randomly perturbed, its new energy calculated, and if this energy is greater, the old location is restored. While this is an inefficient method compared to force-direction models, it simplifies the code enormously which leads to good simulation performance.

It is important to realize that nodes do not send a beacon at each step of the iterative descent, but only once the process has completed.

2.6 Root Node

While it is desirable for all nodes to have the same properties, the simulation used to test this algorithm has one node with a special property – the root node. This node never recalculates its position, and is always at virtual location (0,0,0). This is not an essential property of this algorithm, but is a useful one for a network which will be linked to the outside world, as the root node can act as a gateway to the Internet or similar large network.

The root node does not provide centralized processing to the mesh. In fact, the root node does *less* processing than the other nodes, as it never recalculates its position. Once converged, communication within the mesh does not depend on the root node, and so it is not a single point of failure.

Convergence occurs much more readily when the mesh network calculation is allowed to expand from a single node to its neighbours and their neighbours and so forth.

For these experiments, nodes other than the root node do not send their own beacons until 300 milliseconds after they have received their first beacon from a neighbour. This allows a node to become aware of all of its neigbours before deciding on an initial position. The root node is arbitrarily chosen as the first node to send a beacon, and the localization process propagates outwards from the root node.

2.7 Neighbourhood Size

The explanation above discusses the use of 2-neighbourhood information for virtual localization. In some cases, it is useful to expand this definition and exchange information on the nodes' 3-neighbourhoods. This information can be used by the virtual localization algorithm to obtain a better convergence, and by supergreedy routing to avoid larger voids.

3 Walkthrough

In this section we will show the operation of the algorithm through a simple network of three nodes: A, B and C.

A is the root node, it begins at (0,0,0,0) and broadcasts the first beacon:

ID	*Distance*	*Coordinate*
A	0	(0,0,0,0)

This beacon informs any receiving node that A believes it is at (0,0,0,0) and does not know of any neighbours. The distance of 0 indicates that this is the identity and virtual location of the node sending the beacon. Nodes receiving this beacon will become aware of A as a 1-neighbour.

At some time after receiving this beacon, B will wake up and recalculate its own virtual location. At this point B only knows of one neighbour – A. B performs an iterative descent to find the point of lowest total potential energy as per Equation 3, and as it is attracted to A and knows of no other neighbours, it will converge on coordinate (0,0,0,0)[2].

B sends the following beacon:

ID	*Distance*	*Coordinate*
A	1	(0,0,0,0)
B	0	(0,0,0,0)

This beacon will be received by A and by C. The distance of 1 indicates that A is a 1-neighbour of B. A will learn of B's existence, but take no action. C will learn of both A's and B's existence, and it will be attracted to B, a 1-neighbour, and repelled from A, a 2-neighbour.

When C recalculates its position it will combine potential energies from attraction (Equation 1) and repulsion (Equation 2), as in Figure 5. C will take its initial location from B, and iteratively descend from this maximum into the circular minimum surrounding it. The exact path it takes is unimportant, but it will end up with $d_{CB} = d_{CA} \approx 158$, which is the distance at which the potential energy is minimized.

For the sake of clarity, suppose that C changes its virtual location only in the positive X direction[3]. The minimum energy will be at the coordinate (158,0,0,0), and C will take this as its new virtual location.

C sends:

ID	*Distance*	*Coordinate*
B	1	(0,0,0,0)
C	0	(158,0,0,0)

When B next recalculates its position, it will find itself attracted to both A and C, as in Figure 6. Its potential energy minima will be half-way between A and C, at (79,0,0,0).

B sends:

ID	*Distance*	*Coordinate*
A	1	(0,0,0,0)
B	0	(79,0,0,0)
C	1	(158,0,0,0)

C is still attracted to B and repelled by A, and as the nodes continue to communicate, they will drift apart until they reach an equilibrium when $d_{AB} = d_{BC} = \frac{1}{2}d_{AC} = 100$. Figures 6 and 7 show the energy minima in this situation.

In the case described here, A is the root node, and does not change its position. This simplifies the explanation. In most cases, many nodes will be asynchronously recalculating and beaconing their virtual locations. The same principles apply even in complicated situations.

[2]It may seem odd to have two nodes with the same location. However, this is an intentional feature of the algorithm. Two nodes A and B will converge on the same location if A is a neighbour of B, and there is no node X which is a neighbour of A and not B, or of B and not A. Routes to and from A and B are identical, so there is no reason to regard them as separately located. When C is discovered, which is a neighbour of B but not A, A and B assume separate locations.

[3]whereas in fact, it will move in an entirely random direction at each perturbation.

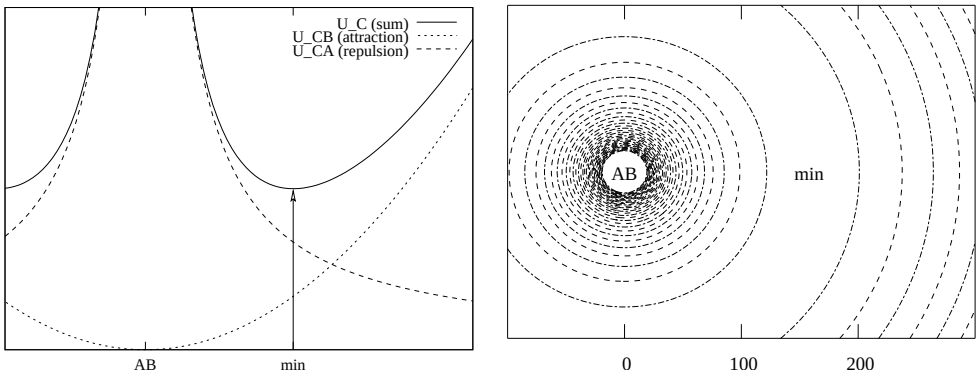


Figure 5: U_C with $d_{AB} = 0$

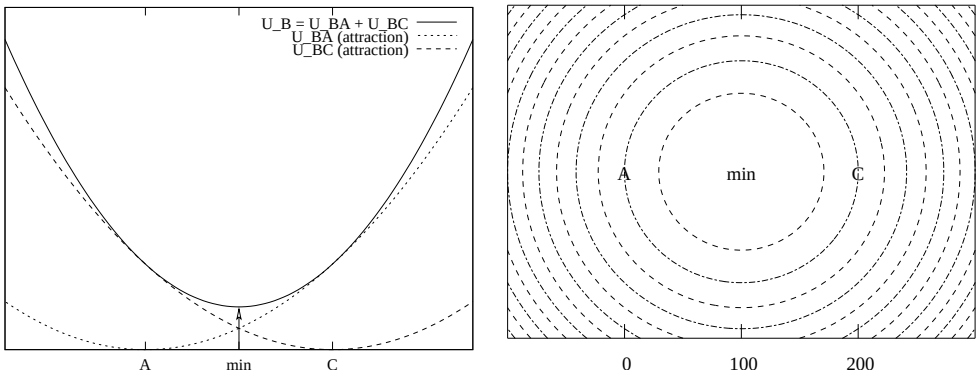


Figure 6: U_B with $d_{AC} = 200$

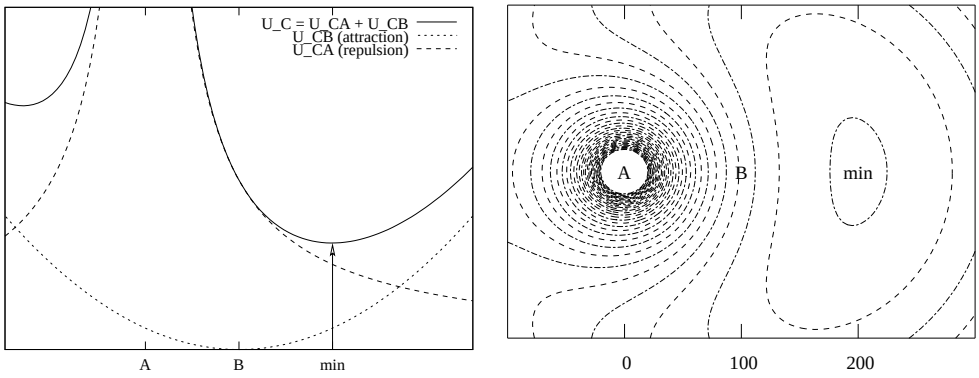


Figure 7: U_C with $d_{AB} = 100$

4 Simulation

We wrote a discrete event simulation program in C in order to develop and test the algorithm. We also wrote utilities in C, Perl, 'bash' and 'awk' to generate network topologies, distribute processing, gather statistics and to transform the simulation output into graphs and animations.

4.1 Generating Topologies

A large number of random topologies were created in order to test the performance of the algorithm on 2- and 3- dimensional problems of varying size and mean degree.

The 'size' of the network is simply the number of nodes in the network. Our tests use networks with 100, 200, 400, 800, 1600 and 3200 nodes. The 'degree' of a node is just its number of 1-neighbours. The 'mean degree' of the network is the average number of neighbours of its nodes which, since every link connects two nodes, is given by

$$mean\ degree = 2\ \frac{N_{links}}{N_{nodes}}.$$

A commonly used method for generating N-node random meshes is to pick N random locations within a square area. This method does nothing to prevent disjoint networks being generated, but for a high density of nodes this rarely occurs. This random scattering of nodes is not very realistic, although it might be similar to the distribution of small, rugged sensor motes scattered from an aeroplane. Čapkun, et al. [14] use this model, randomly distributing 400 nodes in a 1km × 1km square, and varying the range to 90, 100 or 110m.

We use a similar method to generate 2-dimensional problems of 400 nodes, placing the root node at the center of a square kilometer area and distributing nodes within this area until a connected subgraph of at least 400 nodes has been generated. Disjoint parts of the graph are then removed, topological information is extracted and the original location information is ignored.

4.2 Mean degree

The 'degree' of a node is just its number of 1-neighbours. The 'mean degree' of the network is the average number of neighbours of its nodes which, since every link connects two nodes, is given by:

$$mean\ degree = 2\ \frac{N_{links}}{N_{nodes}}$$

The mean degree of the networks produced in this manner depends on the range of communications. Figure 8 shows how the mean degree of the network increases with an increased communications range.

Mean degree is indicative of the overall behaviour of the network, but not all nodes have the same degree. Figure 9 shows the Poisson distribution of node degree in 1km × 1km networks of 400 and 800 nodes with range 100m.

Once the nodes are placed, topological information is extracted and the original location information is discarded. The topological information is then fed to the simulation for testing.

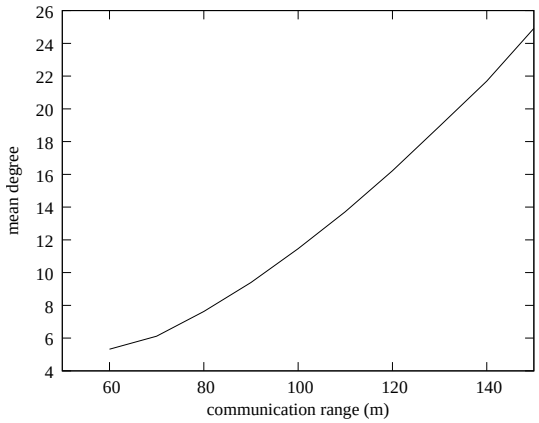


Figure 8: Communication range vs. mean degree

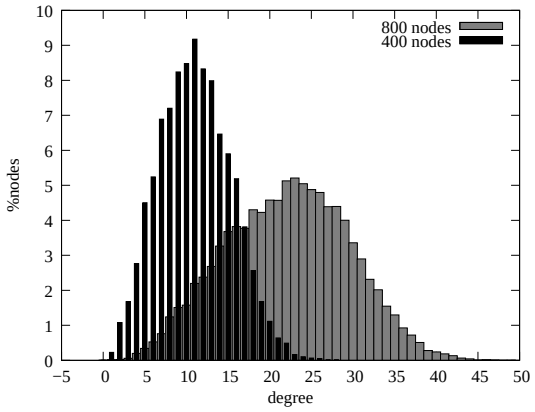


Figure 9: Degree distribution of 400 and 800 nodes with range 100m in a square 1km × 1km area.

4.3 Node Behaviour

In these experiments, virtual locations are 4-dimensional, and every node recalculates its virtual location every 50 to 150 ms. At each recalculation, 100 random perturbations are tried, with the maximum size of perturbation decreasing from 100 by 1 with each try. The process is considered complete when 100 further perturbations of size 1 fail to decrease the energy further. This allows the location to change dramatically when a large change is needed, but concentrates its efforts on fine optimization towards the end. This convergence process is inefficient but it is sufficient to demonstrate the behaviour of the algorithm.

Once the new virtual location is calculated, a beacon is sent. Each beacon contains an identifier for the node, and its newly calculated virtual location. It also contains a list of identities, hop distances and virtual locations of its 1-neighbours (and optionally 2-neighbours) as obtained from beacons from its neighbours.

4.4 Routing

We implemented Greedy Forwarding and Supergreedy Forwarding algorithms (see Section 1.4) and used them to test routability in the mesh network. A node i is considered routeable if a packet can be routed from node i to the root node at (0,0,0), and a packet can be routed from the root node back to the location of node i. This measures the routing performance of a mesh network with a single gateway at the root node.

5 Testing

5.1 Virtual Maps

Figure 10 shows the process of virtual localization in a 3-dimensional space. The topology information is generated by placing 400 nodes at random in a square kilometre, with a link

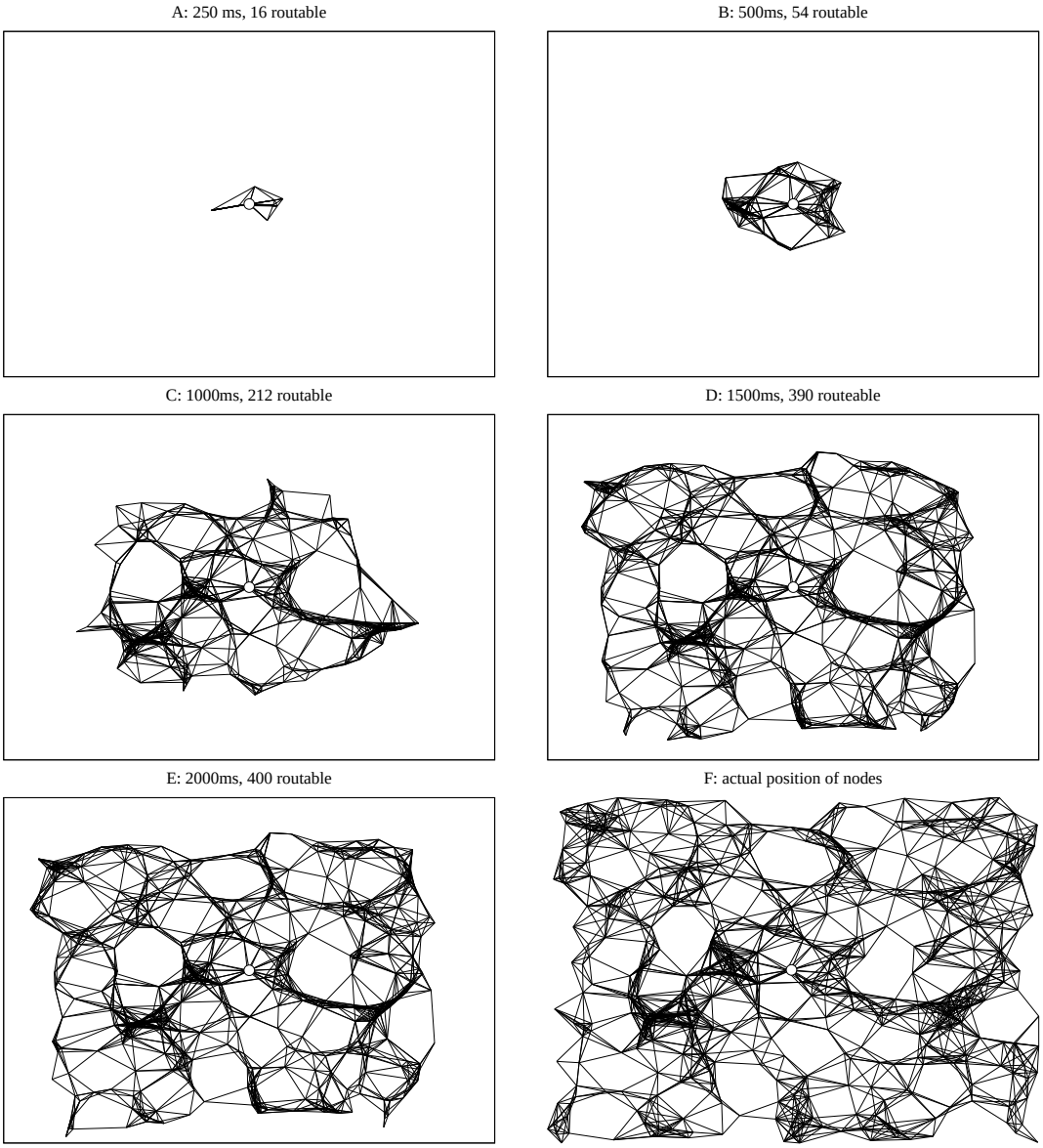


Figure 10: Nodes progressively discover their virtual locations in a 3-dimensional space (seen here at `A`: 250, `B`: 500, `C`: 1000, `D`: 1500 and `E`: 2000 ms) until the entire network has been virtually mapped. Lines join nodes which are 1-neighbours. The root node is marked with a small circle. All nodes are part of a connected network, and all nodes can successfully route packets to and from the root node using supergreedy routing. Note that these are 2-dimensional projections of the 3-dimensional virtual location, and have been rotated manually for the sake of clarity. The original 2-dimensional topology is included as `F` for comparison purposes.

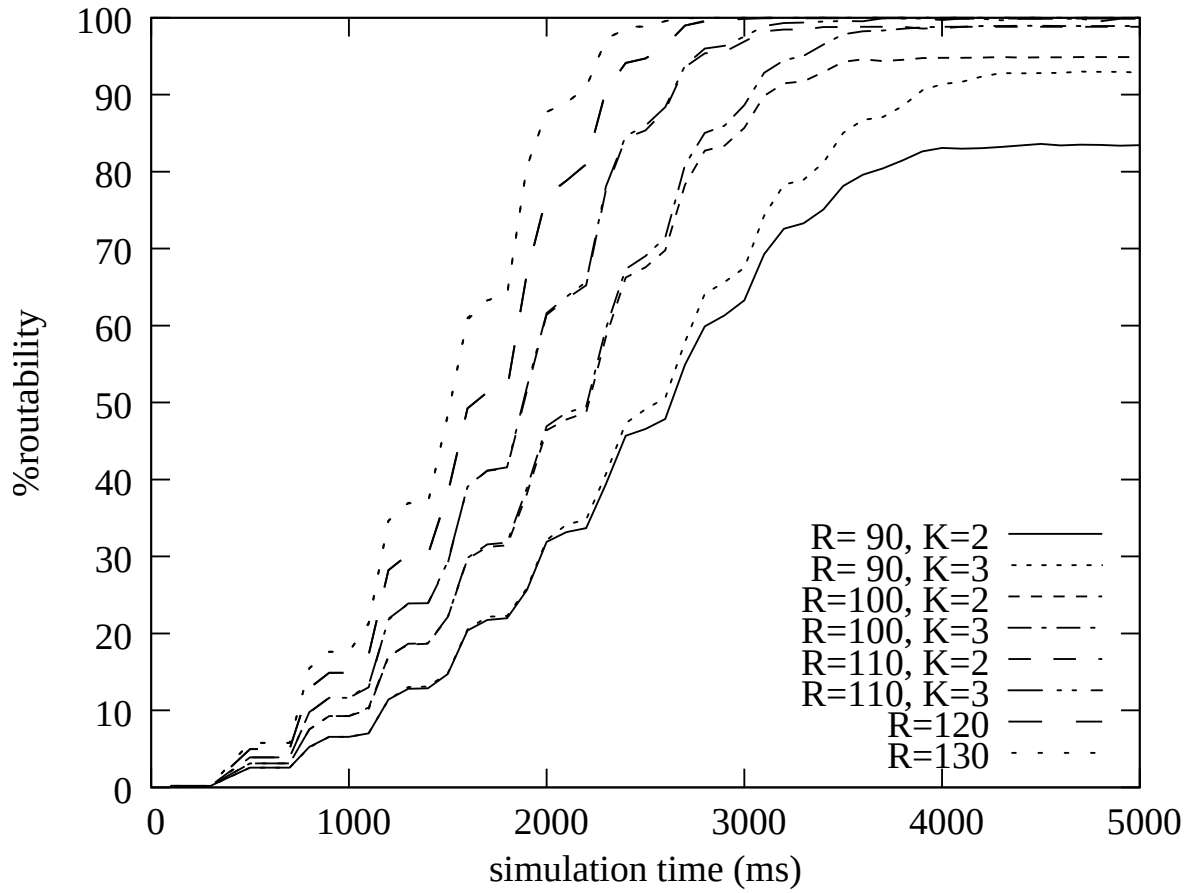


Figure 11: Convergence for networks with varying range (R=90, 100, 110, 120 and 130m) and varying neighbourhood size (K=2,3)

range of 100m. Snapshots of the virtual location of active nodes are shown for 250, 500, 1000, 1500 and 2000 elapsed simulated milliseconds, and at bottom right is a representation of the actual layout of the network which was used to create the topology.

It is important to remember that these 'maps' of the network are a composite, with each node knowing only its own virtual location and that of its 1- and 2-neighbours. The complete map is never broadcast through the network.

Some features of the original network are visible in the virtual layout, but the graphs are not easy to compare visually. A more important metric of the performance of the algorithm is the number of routable nodes. Once the network has converged, 100% of nodes can route packets to and from the root node using Supergreedy Forwarding.

Animated GIF files showing the progress of this convergence and the 3D structure of the virtual location space can be found at:

`http://www.ctie.monash.edu.au/mesh/virt_dim/`

5.2 Convergence

Figure 11 shows the effect of network mean degree on the speed and success of convergence for networks of 400 nodes in a 1km × 1km area. Mean degree was varied from 9.4 to 18.9 by increasing the range R from 90m to 130m. Averages were taken over 50 runs with different random topologies.

Networks with higher mean degree converge more quickly and more completely. The algorithm as presented performs best on dense networks, with mean degree greater than 11.

Figure 11 also compares the performance of the algorithm with varying neighbourhood size K. For networks of smaller mean degree (R=90, 100, 110m), exchanging data about the 3-neighbourhood and using this data for supergreedy routing improves performance markedly. For networks of larger mean degree (R=120, 130m) this improvement is insignificant.

Figure 12 compares the convergence of 2-dimensional networks of varying size but roughly constant mean degree. Range is set at 120m, and mean degree is kept roughly constant by increasing the bounds of the network as $\sqrt{N}$ to keep a constant area per node. Averages were taken over 50 different random topologies per size.

These graphs show that convergence becomes slower as the network size grows. The algorithm as presented performs well on networks of up to 6400 nodes.

Figure 12 also shows the short time and small number of beacons per node required to achieve 95% convergence.

6 Conclusions

In this paper we presented our Virtual Localization algorithm which provides a distributed method for constructing a 'virtual map' of a mesh network, without GPS-equipped anchors, radio ranging, broadcast flooding or centralized processing.

Our simulation based experiments show that the generated virtual map can be effectively used for location-based routing traffic through a mesh network, providing consistent internal connectivity for sensor mesh networks.

In addition, the algorithm presented converges very quickly and with less than 100 beacons sent per node, making it a practical method for routing in sensor networks.

7 Acknowledgements

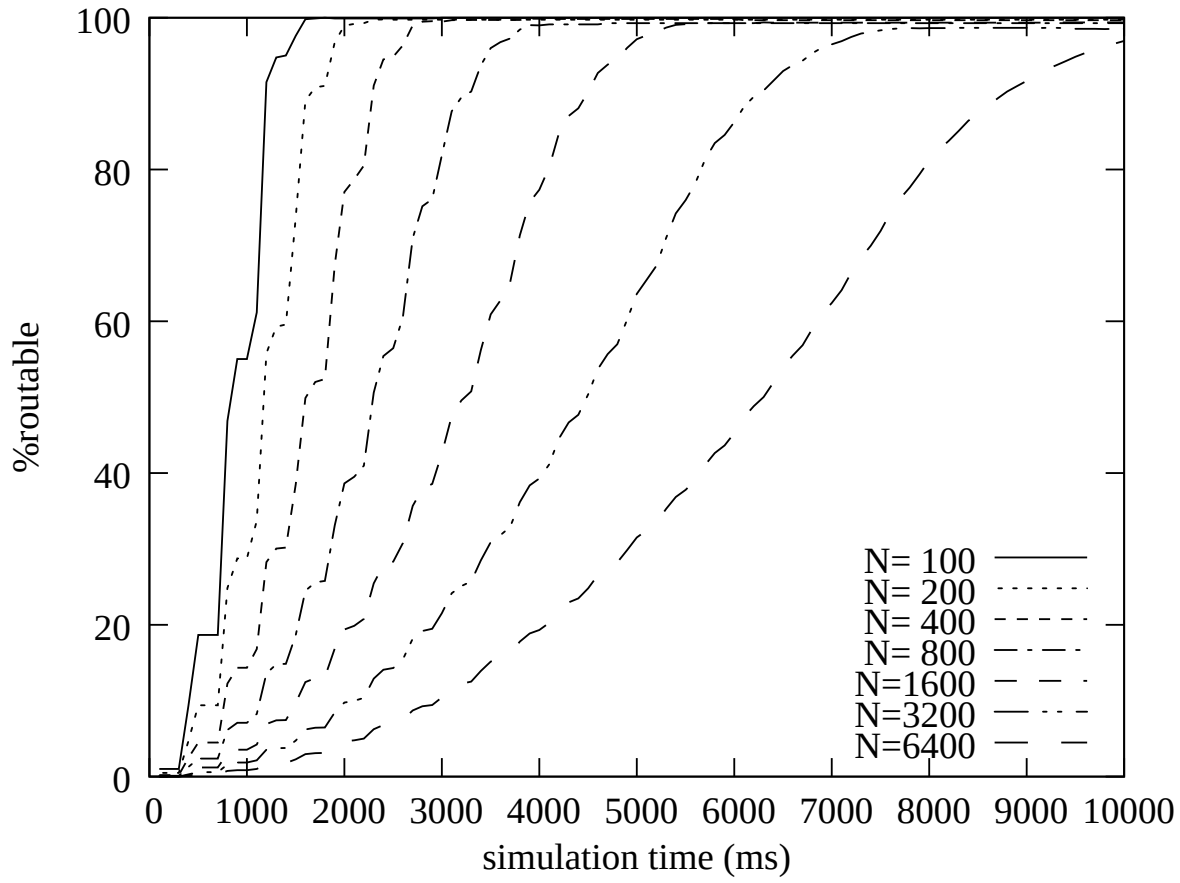
This work was supported by the Australian Telecommunications CRC (ATCRC) `http://www.telecommunications.crc.org.au/`

References

[1] N. Moore, Y. A. Şekercioğlu, and G. K. Egan. Virtual localization for mesh network routing. *Proceedings of IASTED International Conference on Networks and Communication Systems (NCS2005)*, April 2005.

[2] R. Hinden, M. O'Dell, and S. Deering. RFC2374: An IPv6 Aggregatable Global Unicast Address Format (historic). URL: `http://www.ietf.org/rfc/rfc2374.txt`, July 1998.

[3] M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad-hoc networks. *IEEE Network Magazine*, 15(6):30 – 39, November 2001.



Nodes	Mean degree	%routable			to > 95%	
		t=2.5s	t=5.0s	t=10.0s	time (s)	beac./node
100	14.4	99.9	100	100	1.4	5.7
200	15.5	99.7	99.7	99.7	1.9	7.7
400	16.4	94.9	99.8	99.7	2.6	10.4
800	16.8	56.4	99.3	99.3	3.5	13.2
1600	17.1	28.3	97.2	99.4	4.9	18.7
3200	17.4	14.3	63.6	98.5	6.8	25.8
6400	17.6	7.0	31.5	97.0	9.6	36.6

Figure 12: Routability vs. Time for various network sizes

[4] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[5] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. *Proc. ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom 2000)*, 2000.

[6] D. S. J. De Couto and R. Morris. Location proxies and intermediate node forwarding for practical geographic forwarding. Technical Report MIT-LCS-TR824, MIT Laboratory for Computer Science, June 2001.

[7] N. Moore and Y. A. Şekercioğlu. Finite greed: Limits to greedy routing in located meshes. *in submission to IEEE TENCON 2005*, 2005.

[8] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications, Special Issue on Smart Spaces and Environments*, October 2000.

[9] L. Doherty, K. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. *IEEE Infocom 2001*, pages 1655–1663, April 2001.

[10] C. Savarese, J. Rabsey, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. *USENIX Technical Annual Conference*, June 2002.

[11] A. Savvides, H. Park, and M. B. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. *WSNA '02*, pages 112–121, September 2002.

[12] A. Neskovic, N. Neskovic, and G. Paunovic. Modern approachs in modeling of mobile radio systems propagation environment. *IEEE Communications Surveys*, 2000.

[13] T. He, C. Huang, B.M. Blum, J.A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. *Proceedings of ACM MobiCom '03*, pages 81–95, September 2003.

[14] S. Čapkun, M. Hamdi, and J.-P. Hubaux. GPS-free positioning in mobile ad-hoc networks. *Hawaii Int. Conf. on System Sciences (HICSS-34)*, pages 3481–3490, January 2001.

[15] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer. Virtual coordinates for ad hoc and sensor networks. *Proceedings of 2nd Joint Workshop on Foundations of Mobile Computing (DIALM-POMC 2004)*, 2004.

[16] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. *Proceedings of ACM MobiCom '03*, pages 96–108, September 2003.

[17] A. Howard, M. J. Mataric, and G. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2001)*, October 2001.

[18] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[19] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software – Practice and Experience*, 21:1129–1164, November 1991.

[20] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–13, April 1989.

[21] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.