VIRTUAL LOCALIZATION FOR MESH NETWORK ROUTING

Nick Moore Ahmet Şekercioğlu Gregory K Egan

Center for Telecommunications and Information Engineering
Monash University, Melbourne, Australia
email: nick.moore@eng.monash.edu.au

ABSTRACT

We present a novel distributed 'virtual localization' method for mesh networks, such as sensor meshes, which does not depend on radio ranging or the existence of anchor nodes. We show in simulation that it can be used for establishing efficient and reliable location-based routing information.

KEY WORDS
Mesh Networks, Localization, Routing, Distributed Algorithms

1 Introduction

1.1 Sensor Networks

Miniaturized sensors which measure temperature, pressure, humidity and various chemical concentrations have become simple to mass-produce in recent years. The data processing power required to compress and analyse the measurements has become very cheap.

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 can be relayed from node to node and gathered or transmitted from 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 weather changes.

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. 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.

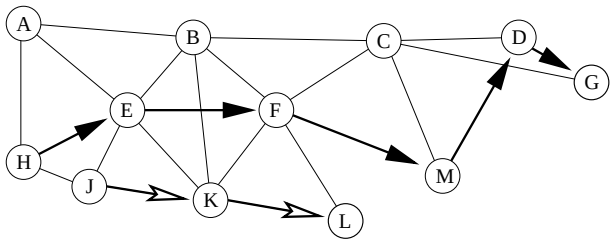

Figure 1. Greedy Forwarding

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.

1.3 Routing Within a Mesh

The classic Internet routing strategy of hierarchical partitioning into sub-networks [1] is not appropriate for mesh networks. Mesh networks 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 [2]. 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.

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, a packet travelling from J to G would be forwarded along the path $\overrightarrow{JKL}$, and be stuck at L as L has no neighbours closer to G than itself. Schemes such as GPSR [3] and INR [4] have been developed to mitigate this problem by using alternative strategies when greedy forwarding fails.

1.5 Determining Location

The routing algorithms discussed above require that all participating nodes are able 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. 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 [5, 6], by iterative methods [7, 8] or using Kalman Filters [9]. However, the 'anchor' nodes may be up to 20% of the population, which establishes an undesirable hierarchy in our formerly egalitarian mesh.

A naïve approach to localization would be to 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. However, radio propagation in the real world is rarely so simple. Due to diffraction, scattering, reflection, refraction and attenuation by intervening materials [10], signal strength is not a one-to-one function of distance. Localization algorithms can assume a limited correlation with signal strength [11] at best.

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 [12], iterative[8] or energy-minimization [13] 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] path $\overrightarrow{HEFCG}$ would be shorter in both distance and hops, but F will forward the packet to M as M is closer to G than C is.

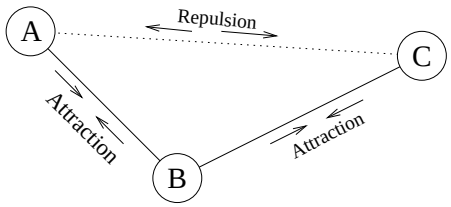

Figure 2. Simple Spring Model

1.6 Mathematical Publications

Decades before engineers began considering 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 [14] presents a system to lay out graphs with "less than 30 vertices", using a simple physical attraction/repulsion model.

Fructerman and Reingold [15] use a similar method with a less directly physical model and a number of optimizations to speed convergence.

Kamada and Kawai [16] delve more deeply into the mathematics, considering the differential equations of a linear spring model.

Davidson and Harel [17] 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

2.1 Spring Models

Figure 2 shows a simple spring model: 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.

Nothing specifies that A, B and C should end up in a straight line, with A and C equidistant from B. But since this is the configuration with the minimum energy, this is the configuration that the springs and masses will converge upon.

This emergent behaviour of the simple spring model is central to the Virtual Localization algorithm presented here.

2.2 Encoding Virtual Location

Our implementation of the Virtual Localization algorithm uses three 32-bit signed integers to represent a position in a three-dimensional Euclidean space. Using an $N+1$ dimensional virtual location to solve an N dimensional problem

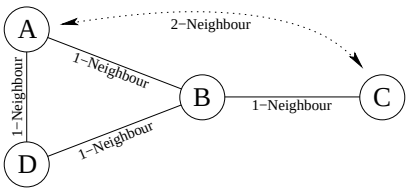

Figure 3. 1-neighbours and 2-neighbours

may seem excessive, but it allows an extra degree of freedom to prevent nodes being trapped in local minima.

2.3 n-Neighbours

When a node can communicate directly with another node, it records that node as a *neighbour of distance 1 hop*, or a '1-neighbour' for short.

A 1-neighbours' 1-neighbour which is not also your 1-neighbour is referred to as your '2-neighbour', because it is a minimum of 2 hops away.

In Figure 3, A has 1-neighbours B and D, and via B it knows of a 2-neighbour C.

2.4 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. In our simulation experiments (section 4), beacons are sent every 50 to 150 ms.

Each beacon indicates the identity and current virtual location of the node, and the identities, hop distances and virtual locations of its 1-neighbours. Each beacon should take up no more than a kilobyte, as there is generally no need to consider more than a dozen 1-neighbours.

2.5 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:

$$U_{ij} = k_{att} \cdot d_{ij}^2$$

where U_{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 have a 'electrostatic-like' repulsion, with a small offset to prevent infinities:

$$U_{ik} = k_{rep} \cdot \frac{1}{d_{ik} + 1}$$

where U_{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:

$$U_i = \sum_{j \in N} U_{ij}$$

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.

2.6 Perturbation

A very simple iterative descent method is used: at each iteration, the node's location is perturbed slightly, its new energy calculated, and if this energy is greater, the old location is restored. At each recalculation, 50 perturbations are tried. While this is an inefficient convergence compared to force-direction models, it simplifies the code enormously which leads to good performance.

2.7 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 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 other 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 they have received 5 beacons from other nodes. Beacons are sent frequently, so this does not introduce a long delay, but it does allow a node to become aware of each of its neigbours before deciding on an initial position. The root node is arbitrarily chosen as the first node to send a beacon.

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) and broadcasts the first beacon:

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

This beacon informs any receiving node that A believes it is at (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 potential energy, and as it is attracted to A and knows of no other neighbours, it will converge on coordinate (0,0,0).

B sends the following beacon:

ID	*Distance*	*Coordinate*
A	1	(0,0,0)
B	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 and repulsion, as in Figure 4. 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. The minimum energy will be at the coordinate (158,0,0), and C will take this as its new virtual location.

C sends:

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

When B next recalculates its position, it will find itself attracted to both A and C, as in Figure 5. Its potential energy minima will be half-way between A and C, at (79,0,0).

B sends:

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

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 5 and 6 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.

4 Test Scenarios

4.1 Simulation

A discrete event simulation program was written in C in order to develop and test the algorithm. Other programs were written to generate network topologies and to transform the simulation output into graphs and animations.

A simple 'greedy forwarding' algorithm has been included to periodically test nodes for routeability. A node N is considered routeable if a packet can be routed from node N to the root node at (0,0,0), and that a packet can be routed from the root node back to the location of node N. All nodes are routeable in the maps presented, but occasionally nodes are temporarily unrouteable until the network layout converges, and sometimes nodes remain unroutable if they are hidden behind routing 'voids'. This effect would be mitigated if a more sophisticated routing algorithm was used.

4.2 Comparing Topologies

The virtual location generated for each node does not directly correspond to the real location of that node, making it difficult to visually compare the results. In addition, the virtual location space is three dimensional, and is projected orthographically for presentation in this paper.

The network maps in Figure 7 have been manually rotated about X and Y and Z to make visual comparison easier.

4.3 200 Node Mesh

Typically, mesh routing algorithms are tested on a network with nodes placed completely at random. Randomly assigning the locations of nodes is inefficient, with large numbers of nodes required to provide complete coverage. A high density of nodes is needed if the network is not to be disjoint.

Instead, for this paper we have placed nodes randomly within the restriction that the minimum distance between any two nodes is half the range, and that every node is placed within range of at least one neighbour. This ensures a non-disjoint network with a limit on the density of nodes and a limit on the number of neighbours of any given node. This kind of placement would be typical where it is possible to select the approximate position of a node but the exact position depends on other factors such as the terrain.

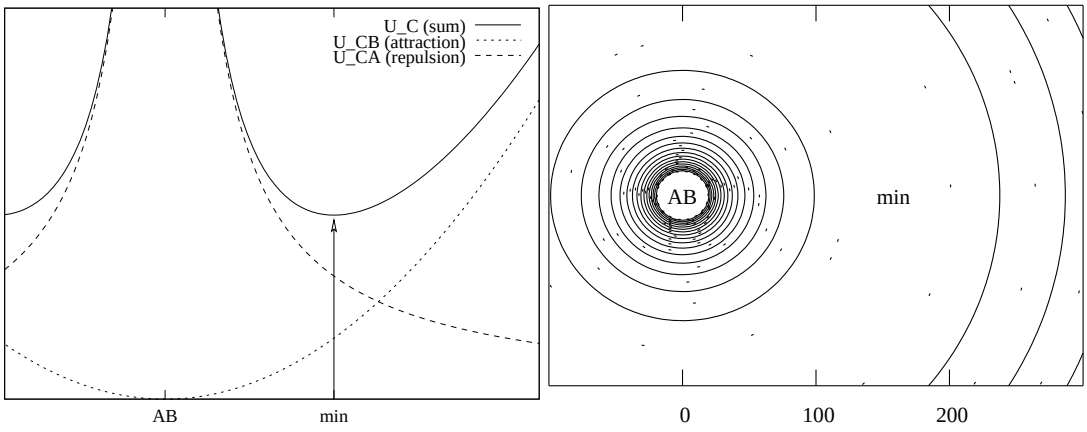

Figure 4. U_C with $d_{AB} = 0$

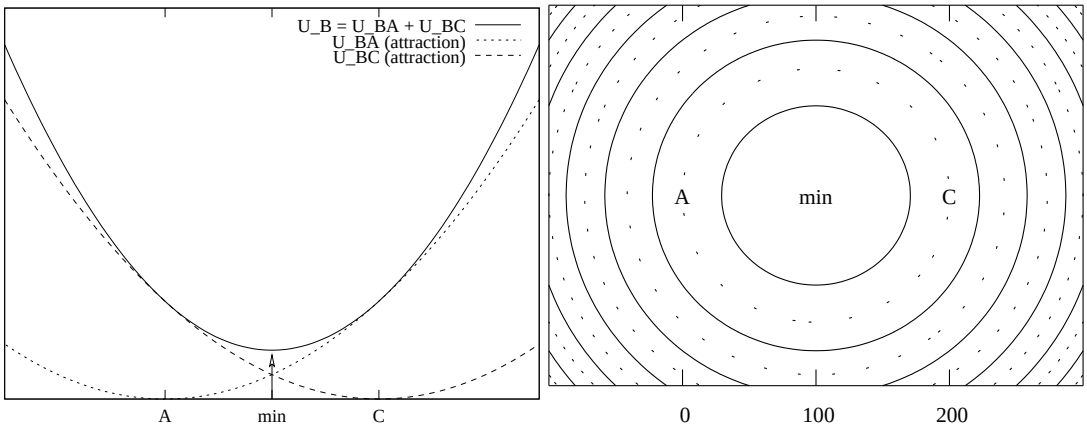

Figure 5. U_B with $d_{AC} = 200$

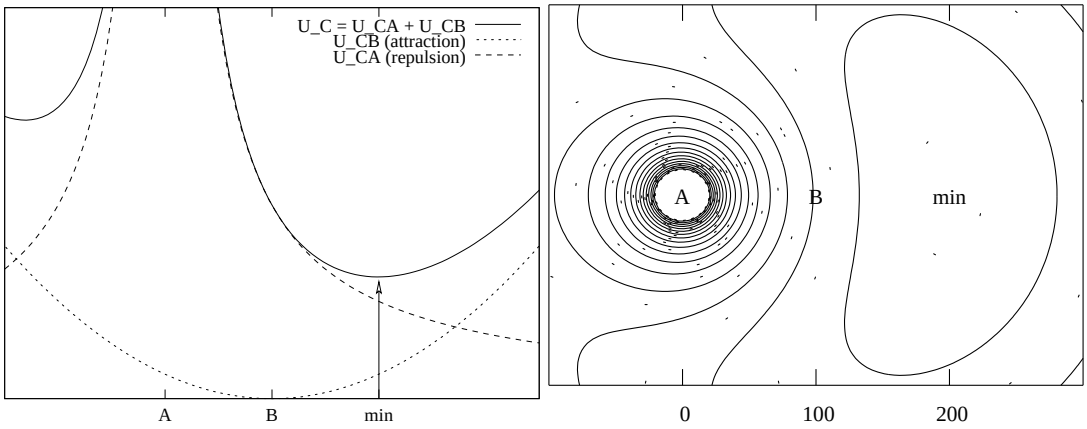

Figure 6. U_C with $d_{AB} = 100$

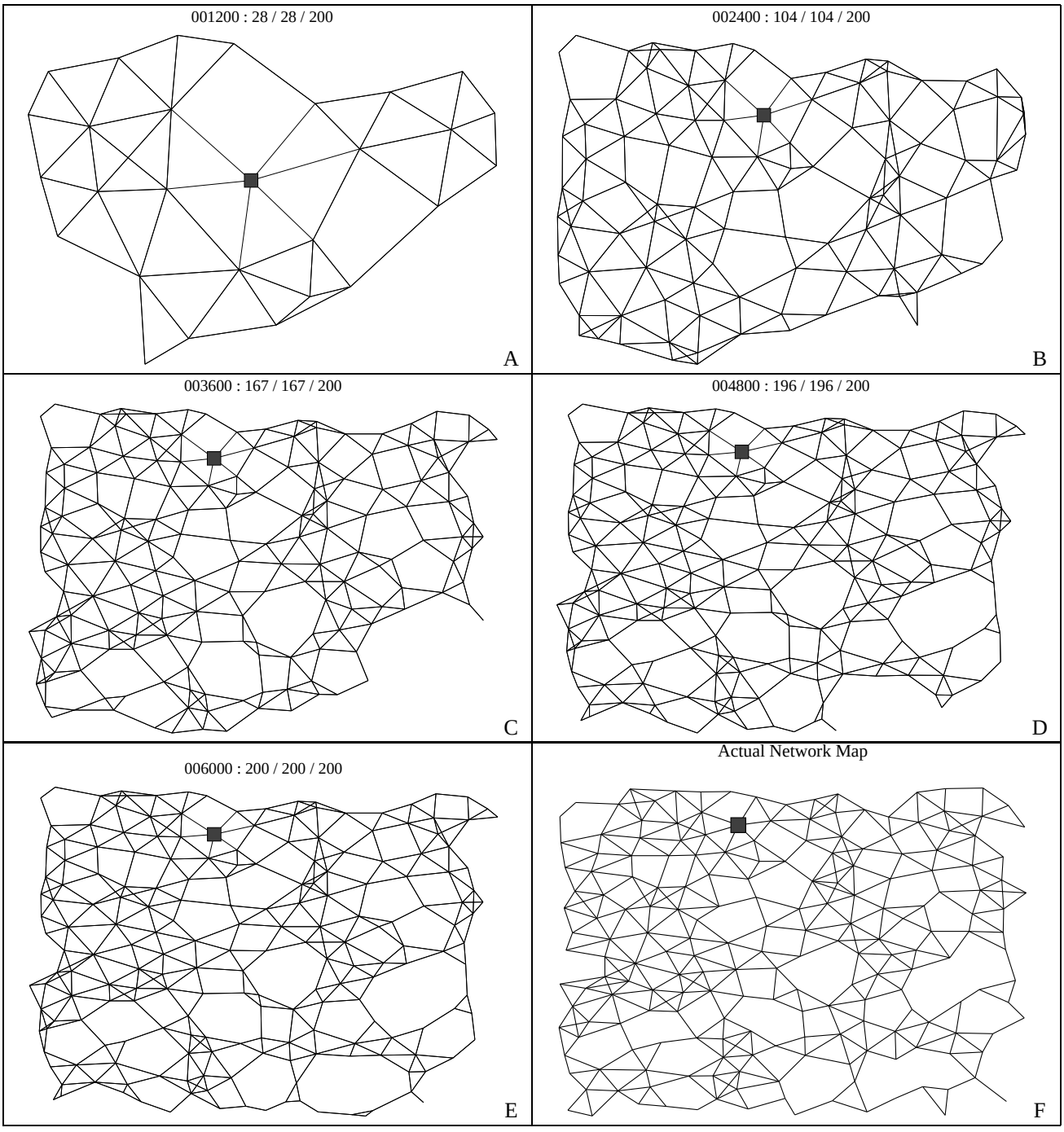

Figure 7. This diagram shows a series of snapshots at A: 1200, B: 2400, C: 3600, D: 4800 and E: 6000 milliseconds, as the nodes progressively discover their virtual locations; F: a map of the original network layout for comparison purposes. The root node is marked.

The Virtual Localization algorithm can then be used to find an efficient routing map of the network.

Figure 7 shows a network constructed in this manner, and illustrates the process of determining virtual locations for each node in the network. The area of interest is one square kilometre, and the communication range is 100 metres.

An animated GIF file showing the progress of this convergence can be found at `http://www.ctie.monash.edu.au/mesh/virt_loc/`

4.4 400 Node Random Mesh

The results of testing a 400 node mesh with randomly placed nodes are not presented here due to space considerations, but are available at `http://www.ctie.monash.edu.au/mesh/virt_loc/`

5 Conclusions

Virtual Localization provides a distributed method for constructing a 'virtual map' of a mesh network, without GPS-equipped anchors, broadcast flooding or centralized processing.

This virtual map can be used for location-based routing traffic through a mesh network, providing efficient internal connectivity for sensor mesh networks.

Future works will investigate the use of more sophisticated routing algorihtms with Virtual Localization, the use of Virtual Localization with mobile and short-lived nodes and the incorporation of relative and absolute direction information into the Virtual Localization algorithm.

References

[1] R. Hinden, M. O'Dell, and S. Deering. RFC2374: an IPv6 aggregatable global unicast address format. URL: `http://www.ietf.org/rfc/rfc2374.txt`, July 1998.

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

[3] Brad 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.

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

[5] 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.

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

[7] Chris Savarese, Jan Rabsey, and Koen Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. *USENIX Technical Annual Conference*, June 2002.

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

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

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

[11] 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.

[12] Srdjan Čapkun, Maher Hamdi, and Jean-Pierre Hubaux. GPS-free positioning in mobile ad-hoc networks. *Hawaii Int. Conf. on System Sciences (HICSS-34)*, pages 3481–3490, January 2001.

[13] Andrew Howard, Maja J Mataric, and Gaurav 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.

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

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

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

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