A Distributed, Leaderless Algorithm for Logical Location Discovery in Specknets

Ryan McNally and D.K.Arvind

Research Consortium in Speckled Computing, School of Informatics, Edinburgh University, Edinburgh, UK

(rmcnally|dka)@inf.ed.ac.uk

Abstract. A speck is intended to be a miniature (5x5x5mm) device that combines sensing, processing, wireless communication and energy storage capabilities [1]. A specknet is an ad-hoc mobile wireless network of specks. The logical location of specks in the network is useful, for reasons ranging from routing data to giving the data sensed a spatial context. This paper presents a novel algorithm for discovering the logical location of specks and updating that information in the face of movement, without recourse to infrastructure support. The proposed algorithm exploits the location constraints implied by the neighbourhood links in order to compute a likely location: one hop neighbours must lie within radio range, two-hop neighbours probably lie outwith radio range. An iterative approach is used to converge on a location estimate that satisfies all constraints. The performance of the location discovery algorithm is evaluated in the SpeckSim simulator for a number of metrics, including location error. The results demonstrate that the quality of the computed locations is within 90% of optimal when used in routing calculations.

1 Introduction

A speck is designed to combine sensing, processing, wireless networking capabilities and a captive power source, all packaged in a volume no larger than a matchstick head. The specknet is a programmable computational network of a large number of specks; in effect, a fine-grained distributed computation platform built on an ad-hoc wireless network substrate consisting of resource-constrained nodes. In addition, the model of distributed computation takes into account some specific features of specknets, such as the unreliability of wireless communication, and a higher than normal failure rate of specks due to the harsh operating conditions, meagre power supply, and less than well tested nodes thanks to large volume manufacturing. This paper addresses the requirement to map the data being sensed, and the information subsequently extracted, to its location within the specknet. In an earlier work [2], we proposed a distributed relaxation-based algorithm for logical location maintenance for resource-constrained mobile wireless ad-hoc networks such as specknets. The algorithm proposed in this paper for logical location discovery of nodes, combined with the earlier published algorithm for location maintenance in mobile ad hoc networks, results in a unique

set of methods for logical location in specknets without the need for any special infrastructure support. Some of the challenges of developing location algorithms for specknets include: specks with limited resources, both in terms of energy and memory; unreliability of the nodes and the communication between them; and the requirement for the algorithm to be fully distributed in the interests of scalability and robustness.

2 Related Work

Algorithms for logical location discovery in wireless sensor networks can be classified along the following lines: whether any special infrastructure is required for the functioning of the algorithm; whether the algorithm relies on a central computational resource; what information is available to the nodes, e.g. connectivity, distance between nodes; whether the algorithm running on all the nodes is homogeneous, i.e. whether all the nodes are equal or some are more equal than others.

The algorithm due to Bulusu et al [3] could be described as using infrastructural support, or as a non-homogeneous network. The approach uses a number of super-nodes with known location (derived from external means or are fixed with location information preprogrammed). These super-nodes periodically broadcast their location, and all the other nodes will set their location as the average of the received locations. This technique is simple and reliable, and can be easily improved by taking received signal strength information into account, or location history in the case of mobile nodes. The drawback is that the number and placement of the super-nodes is critical: ideally, the super-nodes should be deployed in a regular grid, with interlocking areas of radio coverage, which is difficult to guarantee in an ad-hoc network deployment.

Two algorithms that use a central computational resource are described by Doherty et al [4] and Shang et al [5]. Both approaches are similar in that they function by gathering the connectivity data of the entire network into the central computer, which performs the necessary computation and relays each node's location back into the network. They differ in the method of performing the computation.

Doherty's technique constructs a set of constraints on each node's location by examining network links. A radio link between two nodes implies that these nodes must be within radio range of each other, which is then used to construct a distance constraint. The set of constraints, in conjunction with the locations of super-nodes, can be used to compute the likely locations of all the nodes in the network.

The approach due to Shang, dubbed MDS-MAP, computes the shortest network path between every pair of nodes in the network. The number of hops in these paths is likely to correspond to the euclidean distance between the two nodes, at least in networks with a uniform distribution of nodes. Once the distances between every pair of nodes is known, multidimensional scaling is used to compute the relative positions of every node in the network. This relative map

is useful internally in the network, for tasks such as message routing, and can be combined with knowledge of the absolute locations of a subset of the nodes, in order to reconcile the relative map with the physical locations of the nodes.

The advantages of using a central computing resource for location discovery are considerable. The algorithm can be computationally intensive as this computer need not be resource-constrained, and working on a network-wide view of the data. But the disadvantages are also substantial. The process of routing both incoming connectivity data and outgoing location data will in practice result in communication hotspots in the network around the central computer. As a result the drain on the batteries on the nodes closer to the central computer will be greater, and in due course the central computer may becomes unreachable as the nodes around it are exhausted. Another drawback is the latency between when a node transmits its connectivity data and receives its computed location. The data must make two full traversals of the network, and undergo substantial ($O(n^3)$ for Shang, $O(k^2)$ or $O(k^3)$ for Doherty, for n nodes and k network connections) processing in the central computer. These delays would limit their use in highly mobile networks.

Shang and Ruml have proposed an improvement to their MDS-MAP algorithm [6] that distributes the MDS-MAP process across the network, with every node acting as the central computer for its local neighbourhood, and using the MDS-MAP algorithm to compute a relative location map. These local maps are then stitched together by examining the locations of nodes common to pairs of maps, and computing a transform that reconciles them.

This improvement mitigates the drawbacks of having a central computer, and also improves the performance in cases of non-uniform node distribution. However, the computation and memory requirements are significant, and will scale badly as the density of the network increases. On a resource-constrained platform, as nodes are likely to be, these costs may be prohibitive. Another potential problem is that the relationship between network hops and euclidean distance becomes less accurate as the hop count decreases.

3 Algorithm Description

The algorithm exploits the constraints on possible locations implied by network links but, unlike the algorithm due to Doherty et al [4], this approach is fully decentralised and runs on a homogeneous network.

It is important to note that the algorithm will generate a coordinate system that is internally consistent to the network, and can be converted with an affine transformation to reflect the nodes' physical locations. Computing the corrective transformation is simply a matter of examining the real and computed locations of a small subset of the nodes.

The following subsections describe the intuition behind the algorithm, with implementation details left to section 4.

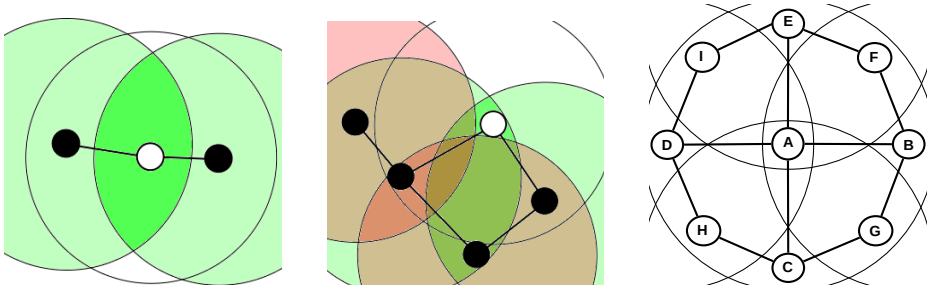

Fig. 1. Loose and tight constraints on node location, and a bridged kernel formation
.

Constraints. As illustrated in Fig. 1, a node that knows its location implies inclusive constraints on the possible locations of its one-hop neighbours, i.e. they must lie within radio range, and exclusive constraints on the locations of its two-hop neighbours, i.e. they probably lie outside the radio range. Once a node detects that its range of possible locations is sufficiently constrained, it commits to a location, and in turn, implies new constraints on its one- and two-hop neighbours. Thus the knowledge of locations will grow from a small kernel of located nodes to encompass the entire network. Even after a node has committed to a location, its estimate is further refined using a numerical relaxation method whenever further constraints are detected.

Kernel Formation. The problem of identifying the initial kernel of located specks is solved by exploiting the idea of mutual ignorance in a neighbourhood. Every node searches for the configuration in which there are four neighbours who do not appear on each others' neighbourhood lists. Once such a configuration is found, it can be asserted that the mutually-ignorant nodes are likely to lie in a 'cross formation', as shown by nodes B, C, D and E in Fig. 1. A further requirement for a valid kernel formation is the presence of bridging nodes F, G, H and I that disambiguate which nodes lie on adjacent arms of the cross formation. Once the formation has been detected by node A, it broadcasts the locations of the mutually-ignorant nodes. The precise values of the computed locations of these nodes are not particularly important, as it will only affect the values of the corrective transform used to reconcile relative and physical locations.

Kernel Incidence. It is probable that more than one kernel will be identified in the network, which will result in a number of competing coordinate systems that must be reconciled. The approach taken here is to have one system take precedence over all the others, based on the identifier (ID) of the original seed node (A in Fig. 1). Normally, nodes will always attempt to locate themselves in the system that has the lowest seed ID, and one system will grow to subsume all the others. However, in practice, it can be the case that the system with the

lower seed ID is a worse candidate for growth than that with the higher seed ID, due to factors such as sparseness of the local network. In these cases the growth will stop, and the network will have two distinct coordinate systems. These can be reconciled using coordinate system stitching techniques described by Shang et al [5], and Moore et al [7].

If the network is deployed in such a manner as to preclude the possibility of a kernel forming, then an artificial kernel can be introduced into the network to start the growth of a coordinate system. All that is required is a small cluster of nodes that have some idea of their locations relative to each other, and the growth of a coordinate system can be triggered.

Guaranteeing Growth. The growth process might stall as the constraints implied on the unlocated nodes are not tight enough for them to commit to a location. This is more likely to occur in sparse networks. This problem is addressed by lowering the threshold for each node to commit to a location. For example, under ideal circumstances the number of located neighbours that a node needs in order to commit to a location might be set to eight. This value is progressively reduced while that node's neighbours report no changes to their location status. The value is reset to the default value as soon as a neighbour's location status changes. This means that the initial estimate of a node's location is more error-prone, but the estimate is later refined as more neighbours commit to a location and imply further constraints.

Seed Death. A potential problem with algorithms in which unreliable nodes can take on a special role lies in the consequences of that node's demise. Fortunately, the special role of the seed node is vanishingly brief. Once the broadcast that sets the locations of the nodes involved in the kernel formation has been made, the seed node has no further responsibilities and can die without notable adverse effects.

Motion. The data shared by each node with its neighbours is a superset of that used in the Distributed Relaxation algorithm for location maintenance [2]. This allows the nodes to maintain their location information in the face of movement with no additional overhead.

Clique Networks. It is important to note that this algorithm is unable to function in degenerate cases such as a clique network, i.e. a fully-connected network in which every node is able to communicate directly to every other node. The only inference that can be made by examining the connection data of such a network is that all nodes are within radio range of each other, which provides a localisation so coarsely-grained as to be essentially useless.

4 Implementation

A fundamental requirement of the algorithm is that every node share information about its neighbourhood with its one-hop neighbours by making repeated broadcasts. The broadcasts' content and the data stored on each node depends on the progress of the coordinate system growth.

Before a kernel is detected, each node maintains a list of its one- and two-hop neighbours' IDs, and periodically broadcasts a list of its one-hop neighbours' IDs. This neighbourhood information is all that is required to detect the kernel formation.

Once a kernel has been detected and system growth has begun, nodes discard the two-hop neighbour information and concentrate on their one-hop neighbours. The nodes maintain a list of each neighbour's ID, location, and the ID of the seed node that each neighbour is located with respect to. The broadcasts will contain the same information, along with the details of the sender. On receipt of a broadcast a node will update its records with the senders details. On examination of the received data, if none of the neighbours have committed to a location, the node will attempt to find a kernel formation. If the search is successful, it sets its own location to be at the origin and broadcasts a message ascribing appropriate locations to the mutually-ignorant nodes.

If the node's neighbourhood contains members that have committed to a location, then the node will scrutinise the constraints implied and determine if it can also safely commit to a location. In general, more located neighbours lead to tighter constraints, and so a convenient way to estimate the tightness of constraints is simply to count them. In all cases, preference will be given to locating with lower-ID seeds.

The procedure for committing to a location uses a simple numerical technique for finding suitable coordinates. Initially, the node will set its location to be the mean of the locations of its one-hop neighbours. This initial estimate is then refined by a sequence of relaxation operations. Each constraint on its location is satisfied in turn by applying the least possible change to the location estimate, i.e. if the location estimate is outside the radio range of a one-hop neighbour, the estimated position is moved towards that neighbour's reported location; conversely, if the estimate is within radio range of a two-hop neighbour, it is moved away. These operations are applied iteratively so that the location estimate will converge on a state in which all constraints are satisfied.

Nodes will only update their neighbourhood records with the details of a sender. Second-hand information (about the sender's neighbourhood) is used to refine the node's location estimate through the relaxation procedure, but is not stored for later use.

5 Metrics and Results

SpeckSim is an event-driven behavioural simulator for wireless networks [8]. The simulation results were based on a typical configuration of 200 nodes distributed

randomly across a unit square. Radio transmissions are assumed to have a circular propagation with a range of 0.2 units. Each node will attempt a transmission once every second. The MAC protocol used on the nodes is trivially simple: when initiating a broadcast, a carrier sense check is made; if another transmission is detected, the node will wait for a random period before attempting a re-broadcast. Simulations were repeated 100 times for each scenario and the mean and standard deviation of the results were calculated.

It should be noted that the circular radio propagation model is convenient and simple to simulate, but does not reflect the behaviour of a radio broadcast in practice. The effective range of a radio broadcast can vary due to multi-path and fading effects, interference with other broadcasts, and presence of conductive materials. This could result in asymmetric links in the network, i.e., node A can receive broadcasts from node B, but not vice versa. This will cause problems with the relaxation procedure used to refine location estimates, i.e., node A will move its location estimate to within radio range of node B upon receiving B's broadcast, and node B will move its location estimate away from node A when it is informed of A's location by a third node, and the cycle begins again. Thus node A will appear to chase node B around in circles.

This problem can be fixed at the cost of some bandwidth by having each node append the neighbourhood lists of its one-hop neighbours to their broadcasts. Therefore a receiving node can detect when it is part of an asymmetric link and terminate the looping behaviour. The additional bandwidth costs can be assuaged by using Bloom filters [9] to compress the neighbourhood lists at the cost of a negligible margin of error.

Performance is primarily judged by location *Error*, i.e. the distance between a node's actual and computed locations, disregarding the effects of any network-wide skew transformation.

Secondary metrics are based on the impact of location data on location-based routing [10]. A simple greedy algorithm is used to route data between two distant nodes; in the first case, perfect knowledge of the locations of the nodes is used, whereas in the second case location data provided by the algorithm is employed. The two routes are then compared and the following statistcics computed:

Success rate The number of successful routes found using computed locations, expressed as a fraction of the number of successful routes found using perfect knowledge.

Efficiency The quotient of the hop-lengths of the two routes. A figure less than one implies that the routes discovered with computed locations were longer than those discovered with perfect knowledge.

The location-based routing metrics highlight that there is much useful information in coordinate systems with large location errors. For example, if the system has a non-uniform scaling factor, i.e., it is stretched in one direction, then the location error will be large, but routing information remains unaffected.

Another metric worth considering is the number of located specks: the proportion of nodes in the network that have been successfully located with respect

to a single seed. For example, if the network was split down the middle, and each half had a separate coordinate system, then the metric would be 0.5.

Kernel Incidence The first graph in Fig. 2 charts the incidence of kernel formations in a random network layout as the network density increases. It shows that there is an abundance of kernel formations for all but the sparsest of networks.

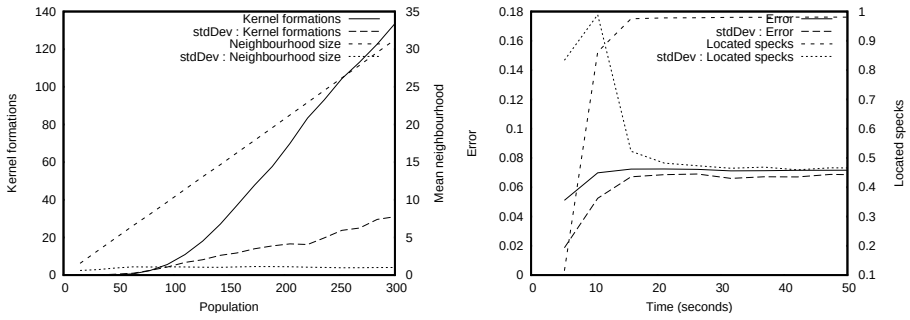

Fig. 2. Kernel incidence and performance with time

Time The graph on the right in Fig. 2 shows the performance of the algorithm over time. It can be seen that the network converges into a single coordinate system in approximately 15 seconds. The stable error level indicates that the nodes have located themselves to the best of their ability, with all constraints having been satisfied. The routing metrics (not shown) show a steady route success rate of around 0.9, with a route efficiency of around 1.0.

Sparse Networks Figure 3 investigates the effects of sparseness, which shows that a stable error rate is achieved even in sparse networks, with around 80% of nodes being located with respect to one seed. The location-based routing statistics indicate that when a successful route is found, its efficiency is very close to that of a route found using perfect location information.

Bandwidth Conservation The previous graphs have been generated without taking the communications bandwidth into account. This is acceptable for static networks as the message frequency can be reduced for low-bandwidth communications. However, communication is one of the most expensive operations for a speck from the point of view of power consumption, and so algorithms must strive to keep messages as terse as possible. The default behaviour of the algorithm is to simply broadcast a node's entire neighbour list. Fortunately, we can place a cap on the number of neighbours for which full details are broadcast without significantly degrading performance. This dramatically reduces the size of each message, and further savings can be made by limiting the precision of location information. Taking these two savings together, Fig. 4 shows that limiting the broadcast neighbourhood details

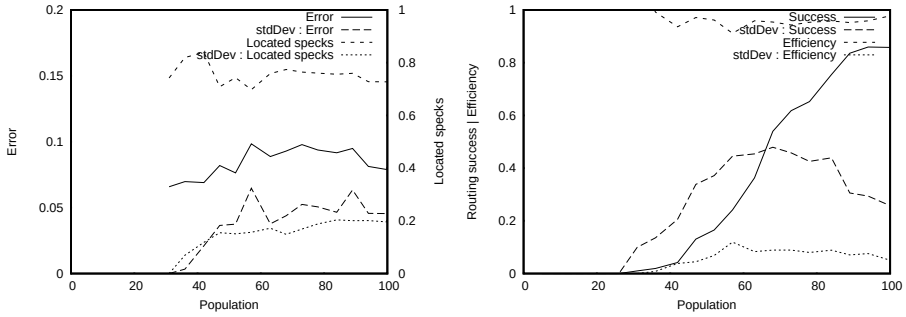

Fig. 3. Sparse networks

and their precision, will lead to an increase in performance over the default behaviour in low-bandwidth situations, in addition to bandwidth savings of around 60%. This is due to lower channel utilisation and hence a higher message delivery rate. The first graph charts performance and message size as the neighbour limit is increased, while the second graph shows the performance of the algorithm with no limits, and with the number of neighbours limited to 8 and a precision limit of 8 bits.

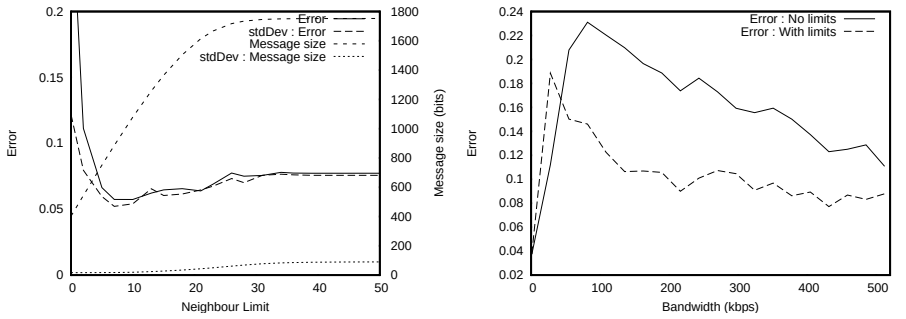

Fig. 4. Bandwidth effects

6 Conclusion and Future Work

We have described a novel method for logical location estimation for mobile networks of resource-constrained devices. This method uses only binary connection information, as opposed to received signal strength or other methods of range estimation. The method does not use any special infrastructure nodes such as

beacons with known positions. In addition it is entirely distributed, running on a homogeneous network of nodes. Simulation results have been presented to demonstrate the performance of the algorithm with respect to different parameters. For future work we intend to investigate methods by which disparate coordinate systems may be reconciled, and also the imapct of mobility, unreliable and non-uniform transmissions and obstructions will have on the network formation.

References

1. Arvind, D.: Speckled computing. In: Technical Proceedings of the 2005 NSTI Nanotechnology Conference. (May 2005)
2. McNally, R., Wong, K., Arvind, D.: A distributed algorithm for logical location estimation in speckled computing. In: Proceedings of IEEE Wireless Communications and Networking Conference. (3 2005)
3. Bulusu, N., Heidemann, J., Estrin, D.: Gps-less low cost outdoor localization for very small devices. IEEE Personal Communications Magazine **7**(5) (October 2000) 28–34
4. Doherty, L., Pister, K.S.J., Ghaoui, L.E.: Convex position estimation in wireless sensor networks. In: Proc. of the IEEE Infocom. (2001) 1655–1663
5. Shang, Y., Ruml, W., Zhang, Y., Fromherz, M.: Localization from mere connectivity (2003)
6. Shang, Y., Ruml, W.: Improved mds-based localization (2004)
7. Moore, D., Leonard, J., D.Rus, Teller, S.: Robust distributed network localization with noisy range measurements. In: Proceedings of Second ACM Conference on Embedded Networked Sensor Systems (SenSys '04), Baltimore, MD, USA (11 2004) 50–61
8. McNally, R.: Specksim at www.specknet.org/dev/specksim
9. MITZENMACHER, M.: Compressed bloom filters. In: Proc. of the 20th Annual ACM Symposium on Principles of Distributed Computing. IEEE/ACM Trans. on Networking (2001) 144–150
10. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless networks. In: Mobile Computing and Networking. (2000) 243–254