

A Distributed Algorithm for Logical Location Estimation in Speckled Computing

R. McNally, K.J Wong, D.K Arvind

Institute for Computing Systems Architecture
School of Informatics, University of Edinburgh
Mayfield Road, Edinburgh EH9 3JZ, Scotland.

r.mcnelly@sms.ed.ac.uk | k.j.wong@sms.ed.ac.uk | dka@inf.ed.ac.uk

Abstract — *Speckled Computing [1] is an emerging technology in which data will be sensed in minute (eventually one cubic millimetre) semiconductor grains called Specks. Information will be extracted in situ from each Speck and will be exchanged and processed in a collaborative fashion in a wireless network of thousands of Specks, called a Specknet. Specks are not assumed to be static and therefore estimating and maintaining the logical location of the mobile Specks in a network would be essential for a number of Speckled Computing and sensor network applications. A novel lightweight distributed algorithm is introduced in this paper for this purpose and simulation results are presented to determine the goodness of the algorithm for different parameters. The algorithm was also successfully ported to a hardware prototype of the Speck called the ProSpeckz. The problems and issues of porting the algorithm onto such a resource-constrained hardware platform will also be discussed. Finally, the paper concludes with plans for future work to be carried out to improve the algorithm.*

Index Terms — Logical Location Estimation Algorithm, Speckled Computing, Intelligent Sensor Networks

I. INTRODUCTION

A Speck is intended to integrate sensing, processing and wireless networking capabilities in a minute semiconductor grain, which will ultimately be the size of a pinhead. Specks will be autonomous with their own energy source and will be assumed to be mobile. Thousands of Specks are designed to collaborate as a programmable computational network called a Specknet. Specknets are intended to be a generic platform for ubiquitous computing, wherein data is sensed processed and information is extracted *in situ* in a collaborative fashion. Some Specknet applications require the estimation and maintenance of the location of the location of each Speck node. Sensor data may have little or no value if it is not informed by the location where it was sensed; indeed, the location of each node in the network might very well be the data to be sensed.

Computing the location of nodes in such a dynamic wireless network is not without difficulties. Specks are typically feeble in terms of computation and storage, communication between nodes will be unreliable and relatively expensive, and the network can be of any size and density. A good solution to this

problem should require minimal computation and storage, be robust against unreliable communication, and be fully distributed across the network. This paper presents such an algorithm which meets these criteria for estimating the location of each Speck locally, which is based on two-hop neighbour information, in the face of movement and importantly does not use received signal strength as an indication of range between Specks.

In the rest of this paper, Section II outlines existing algorithms for location discovery and their unsuitability for Specknets; Section III describes the proposed algorithm; Section IV describes the simulator and the metrics chosen to evaluate the goodness of the algorithm; Section V details the simulation scenarios and discusses the results; Section VI describes the implementation of the algorithm on a hardware prototype of a Speck called the ProSpeckz and the problems involved in porting to such a platform with plans for future work and conclusions outlined in Sections VII and VIII, respectively.

II. RELATED WORK ON LOCATION DISCOVERY

Existing work in this area has understandably focused on location discovery, and one of the ways of maintaining location data is to repeatedly run a location discovery algorithm. However, most methods for location discovery have drawbacks that would preclude their use for Speckled Computing.

For instance, the Lighthouse location System [2], proposed for use with Berkeley's Smart Dust, requires a base station equipped with rotating beacon lights and a clear line of sight to every node in the network. This approach is clearly unsuitable for applications in which the Specknet must be unobtrusive.

Doherty et al [3] describes another approach that also requires a base station. This approach coalesces the connectivity data of the entire network into the base station, where it is processed and the resultant location data is then dispersed back into the network. Such an approach also suffers from the lack of scalability. As the network size increases, the network latency and energy requirement of collecting and disseminating such a large data set, along with the computation time, would prove inefficient.

The system proposed by Bulusu et al [4] does not require a central base station but does require an infrastructure of beacon nodes to be placed with known and fixed positions. A node's position can then be estimated based on which of the beacon nodes can be contacted by radio. However, the ad-hoc nature of SpeckNet disallows the use of such an infrastructure.

Other algorithms [5], [6] rely on Received Signal Strength Information (RSSI) in order to estimate the location. However, RSSI is notoriously unreliable indicator of physical distances, as they are affected by multi-path radio propagations and interferences. Specknets aspires to be a truly generic technology, with the ability to be deployed and to adapt in any environment without the need for additional calibration.

III. THE PROPOSED ALGORITHM

The proposed algorithm presented here is a simple distributed algorithm that enables each node to estimate its own logical position (or logical location). A logical position is defined as a coordinate in two/three-dimensions that is relative to the logical positions of the other nodes. Thus, a set of logical positions would give the layout of the nodes in a network without the physical effects of rotation and translation. To simplify the notations, all locations or positions referred from this point onwards in this paper would be logical coordinates instead of physical coordinates.

The basic premise of the proposed algorithm is the use of inclusive and exclusive distance constraints on a node's position. By using the estimated location of the nodes within a two-hop distance, a node is able to estimate its location by including and excluding areas in which it could possibly be. If node *A* can communicate with nodes *B* and *C*, then *A* must be located within a distance *r* of their positions, for a given radio range *r*. This forms the inclusive distance constraint. On the other hand, exclusive distance constraints on a node's position can be inferred from the nodes that cannot be contacted. In Figure 1, if *A* cannot communicate directly with *D*, then it must be located at a distance further than *r* from its position. Thus, by applying the inclusive distance constraints with *B* and *C* and the exclusive distance constraint with *D*, the set of possible positions for *A* is restricted to the shaded segment.

The algorithm can thus pithily be summarised as "move towards your neighbours, move away from your neighbour's neighbours". A more comprehensive description is as follows:

- a. Each node maintains a neighbour list of its one hop neighbours' identifiers (id) and positions. The list degrades over time and is updated when the node receives a location information packet.
- b. Each node will periodically broadcast a location information packet that consists of its id and position, along with the contents of its neighbour list.

- c. Upon receiving such a packet, the receiving node will:
 - Update its neighbour list with the sender's details
 - For every entry in the neighbour list, satisfy the inclusive distance constraint on the node's position.
 - For every node detailed in the received message that is not in the neighbour list, satisfy the exclusive distance constraint.
- d. Satisfying an inclusive constraint implies moving the node's computed position closer to the position of the other node involved in the constraint. For instance, if node *A* receives a broadcast from node *B* that has a maximum range of 10 units, but *A*'s and *B*'s computed positions are such that the distance between them is 16 units, then *A* will estimate its position closer to *B*'s. To fully satisfy the constraint would require *A* to be moved 6 units closer to *B*, but since *A* and *B* are equal partners in the constraint, *A* should only move 3 units closer. The remaining distance should be moved by *B* the next time *B* receives a broadcast from *A*.
- e. Satisfying an exclusive constraint is almost identical to the inclusive case, where instead of pulling nodes closer, they are pushed apart.

Computing an exact solution to satisfy each node's set of constraints requires complex calculations. However, the proposed algorithm here uses a disarmingly simple iterative approach. Each node will satisfy each of its constraints in turn by moving its computed estimated position, and over time the network as a whole will converge to a steady and valid state after several iterations even though each node begins with an erroneous estimation of its own location.

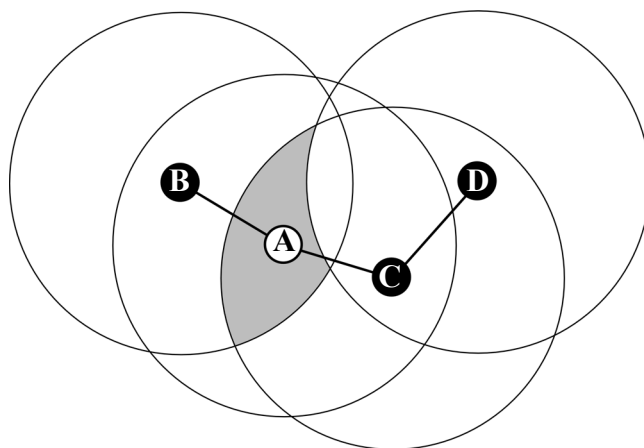


Figure 1. The use of inclusive and exclusive constraints to estimate the possible position of node A

IV. THE SIMULATOR

The performance of the algorithm was simulated on a Java-based simulator. The assumption was made that radio propagation is spherical, with a known and uniform range. The movement model used is one of random waypoints. Initially, the nodes are scattered over a unit square. Each node chooses a random destination point and moves towards it at a defined maximum speed. Upon reaching the destination, a new waypoint is chosen, and the operation continues. The challenge is to maintain each node's location data using minimal resources.

The performance metrics measured on the simulator are:

- a. **Raw error** - The distance between where a node actually is and where it thinks it is. This is a measure of the error in the actual physical location.
- b. **Aligned error** - The distance between where a node actually is and where it thinks it is, but disregarding the effects of any network-wide rotation and translation. This is used to measure the internal accuracy of the network layout. This is the measure of the error in the logical location.
- c. **Trajectory-based routing (TBR) statistics** - One possible use of the location data is to aid in packet routing decisions. For example, if node *A* wishes to send a packet to some distant node *B*, with a known location, then it can use the location data to decide which of its immediate neighbours the packet should be routed through. There are three different variations of TBR implemented on the simulator:
 - **Maximum Distance** - The packet is routed to the node that makes the most progress towards node *B*.
 - **Minimum Distance** - The packet is routed to the node that is closest to node *A*, while still making some progress towards node *B*.
 - **Closest Path** - The packet is routed to the node that lies closest to the line between node *A* and node *B*.

The performance of TBR using the estimated logical location can then be quantified in two ways:

- In comparison with perfect location information: For every decision in the routing algorithm, it is run once based on the source node's neighbour list, and again based on perfect information of the network. If the two choices agreed, then the correct decision was made; if they disagreed, then the routing decision was incorrect.
- In comparison with the actual progress that each decision makes: For every decision in the routing algorithm, a node chooses a neighbour that it believes will be most likely to make progress

towards the destination based on the calculated logical positions. If the chosen neighbour was physically closer to the destination, then the choice was correct, or incorrect, otherwise.

V. SIMULATION SCENARIO AND RESULTS

All the simulations were run on a 1 unit square area with the radio range of each node being set to 0.2 units. Each set of results would be based on scenarios generated from densities ranging from 10 to 200 nodes per unit square (in 10 node steps) and speeds between 0.0 to 0.2 units per message transmitted (in 0.01 unit steps). Each scenario was executed 20 times and the results were averaged. One point to note is that the speed is defined in the results as the unit distance travelled per message transmitted (unit per message) and is a normalisation of transmission rate (message per second) and velocity (unit per second).

Two sets of simulation results are next discussed. Firstly, the aligned error caused by the algorithm is shown in Figure 3. The maximum error that is possible in the worst-case scenario is approximately 0.35 units due to the geographical constraints of the simulations. The worst performance occurs when the density is 10 nodes per unit square. Moving at speeds of about 0.2 units per message, the algorithm in this case would reach the maximum error. As the density increases, the performance of the algorithm improves. The best possible results are presented when densities are above 90 nodes per unit square. The error rates are almost linear to the speed. From the results, one can surmise that the location algorithm presented in this paper would perform optimally in networks with nodes having approximately 11 neighbours (Radio Coverage area * density = $(3.14 * 0.2^2) * (90/1) = 11$). Therefore the algorithm should perform well for Speckled Computing applications that require highly dense, mobile networks.

Logical location information can also be used for location based network routing algorithms such as those presented in [7], [8], [9]. The simulation results in Figure 4 are based on TBR, which demonstrates the utility of the proposed algorithm for making routing decisions. Each node will forward messages towards the destination, and if the node that it forwards to is physically closer to the destination then the decision is deemed to be correct, or wrong, otherwise. For each simulation run, the number of correct decisions made is then divided by the total number of decisions made to give the probability of a correct decision.

The simulation results presented in Figure 4 demonstrates that in a static network, the proposed algorithm works perfectly by making the correct decisions all of the time. As the speed increases, the performance of the algorithm degrades: this decrease is initially gradual – at speeds of up to 0.06 unit per message, correct decisions are still made 95% of the time, for a density greater than 110 nodes per unit square; above this

speed, the performance drops much faster. The drop stabilises at a speed of 0.16 units per message, when 60%-55% of correct decisions are made most of the time. One point to note is that even at speeds of 0.2 units per message, the estimated logical position information still provides the routing algorithm with some assistance, and outperforms a random guess (probability of 0.5).

VI. IMPLEMENTATION ON THE PROSPECKZ (SPECK PROTOTYPE)

The feasibility of the algorithm was investigated on a hardware Speck prototype called the ProSpeckz (Programmable Specks over Zigbee Radio) which the size of a quarter credit card. Figure 2 shows a picture of the ProSpeckz and the layers within the prototype. ProSpeckz contains a Programmable Systems-on-chip (PSoC), which is an 8-bit microcontroller with 16Kbytes of ROM and 256 bytes of RAM. The PSoC also allows the user to reconfigure hardware analogue circuits such as ADCs, DACs, amplifiers and filters on-board the ProSpeckz under software control. This allows the ProSpeckz to be extremely flexible and easy to interface with existing electrical infrastructure. The ProSpeckz is fitted with a 2.4GHz radio for communication at a data rate of 250kbps, with an adjustable range between 20 centimetres and 20 meters, using an on-board antenna.



Figure 2. The Speck prototype “ProSpeckz” and its system overview diagram

Porting the algorithm to ProSpeckz presents a number of challenges. Real-world phenomena such as radio interferences and multi-path propagations will be present which are often unpredictable. Furthermore, a memory limitation of only 256 bytes in the PSoC limits the amount of local information that can be stored for location discovery and radio communication. The limited memory also restricts the complexity of the location estimation algorithm that can be implemented.

Despite all the above-mentioned constraints, the location estimation algorithm presented has been successfully implemented on the ProSpeckz. However, several problems

have emerged during the development. These problems and the solutions implemented will be discussed briefly.

- a. **Asymmetrical Inclusions** – This occurs when a node A is able to hear a distant node C, which is theoretically out of its range. But, Node C cannot hear A. Thus, when A receives a packet from C, A will pull its location towards C, which is an erroneous decision. To overcome this phenomenon, an additional constraint is placed such that before A pulls itself towards C, it will check whether it is on the list of neighbours detailed by the location information packet sent by C. This enforces symmetrical link constraints on the algorithm.
- b. **Somnambulism/New Nodes** – It is possible for a node to have to switch itself off for a while due to lack of power. After it has scavenged enough energy from the environment, through the use of a photovoltaic source or some similar renewable energy device, it will be able to resume its activities. This scenario presents an interesting challenge for the algorithm. The node may have moved during its sleep cycle, and so its knowledge of its own location will have become out-of-date. This problem will also occur when a new node joins the network. If the node then broadcasts its location information, as detailed in the algorithm, it will very likely lead to an increase in the error for the neighbours as its stale location estimate will adversely affect the neighbours’ own estimates about their current positions. Instead, an alternative approach has been implemented such that the node would clear its location data and neighbour table when it awakens/starts. It will then listen to the broadcast of the other nodes and only resume broadcasting either after it can make a sufficiently good initial guess of where it is, based on the transmissions of its new neighbours, or after a timeout.
- c. **Rotation and Translation Errors** – It was observed both in the simulator and in the hardware implementation that a global rotation and translation occurred. This caused the logical locations to change even though they were still logically correct at any time. This phenomenon can be overcome in two possible ways. Firstly, beacon nodes with fixed positions could be placed in the network. This will place some weight on the logical positions. For example, if 3 beacons were placed at 3 different physical corners of the network, rotation and translation errors would cease to occur. The second method solves the problem without the need for beacons. The solution is to make nodes “selfish” once their estimates are stable for some time. “Selfish” nodes will not recalculate their positions as long as there are no changes to its 1-hop neighbour list.
- d. **Erroneous data packets and Malicious nodes** – It is possible that nodes receive erroneous data from other nodes, either due to poor radio connectivity or via

malicious nodes. To limit the damage caused by such data, a constraint is placed on the algorithm that it would discard any location information that would cause it to shift drastically. This is not foolproof and other methods for detecting erroneous data packets and malicious node will be the subject of future research.

VII. FUTURE WORK

6.1 Modelling and investigating the effects of realistic radio models in a large scale network

The simulator will be developed to investigate the effects of realistic radio propagation models on a large Specknet. The radio model implemented currently in the simulator is idealistic in assuming a spherical maximum extent, which is simply not the case in the real world, as demonstrated in the physical implementation on the ProSpeckz.

The addition of unreliable radio communication and radio-opaque barriers in the network will be detrimental, but it is anticipated that some simple modifications to the algorithm can be made to mitigate the degradation. Currently, if two nodes cannot communicate, the algorithm makes the simple assumption that this is because they are too distant. Relaxing this assumption should compensate somewhat for imperfect radio communication and would be investigated in the future.

6.2 Evaluation of the algorithm for location discovery

In the results reported in this paper, the simulations have been carried out to evaluate the location maintenance capability of the algorithm. The nodes start life with correct location information. The challenge addressed here is to maintain that information in the face of movement. Future work will investigate how this algorithm would also perform location discovery.

6.3 Movement models

The waypoint movement model implemented in the simulator was chosen due to its simplicity of implementation and fairly pathological nature. It is unlikely, however, that this movement model will be encountered in the real world. It would therefore be useful to determine the performance of the algorithm in more realistic situations. For example:

- Having chunks of the network moving around as a unit, to simulate the effects of nodes being attached to some rigid body that then has some degree of motion, such as an articulate body
- Nodes in a turbulent fluid
- Networks in a mixture of static and mobile nodes
- Nodes and networks that have constrained movement.

The algorithm presented in this paper operates on a two-dimensional plane but need not be limited to one. Adding a

third dimension would be straightforward, and it would be interesting to see how the algorithm performs in a virtual 3D space.

6.4 Use of sensors to enhance location estimations

Most location algorithms focus on the radio as the tool for distance measurements, but there are a number of other sensors that lend themselves naturally to this area. One such is the accelerometer, which can also be used to dynamically adjust the transmission rates of the packets used for location discovery. As discussed in section V, as the velocity of the nodes increases, the transmission rates of the location information packets would also have to be increased proportionately to maintain a level of quality in the location estimates. Conversely, as the velocity of the nodes decrease, the transmission rates could be lowered to both save power and decrease the network overheads.

VIII. CONCLUSIONS

A novel lightweight distributed logical location approximation algorithm has been described. Simulation results have been presented to qualify the goodness of algorithm against different parameters. The efficacy of the algorithm was tested on a hardware prototype of the Specknet. Although the algorithm has been developed and demonstrated in the context of Specknets, it is sufficiently general for use in traditional sensor networks, vehicular tracking, and robotics, without the need to use expensive Global Positioning System (GPS) devices in applications where logical location estimates would be sufficient.

IX. ACKNOWLEDGEMENTS

The Scottish Higher Education Funding Council as part of the Strategic Research Development Grant funds the Research Consortium for Speckled Computing. The authors wish to acknowledge SHEFC for their support, and thank the members of the Consortium, especially the Speckled Computing Group at Edinburgh, for helpful discussions.

REFERENCES

- [1] D.K. Arvind, K.J.Wong, "Speckled Computing: Disruptive Technology for Networked Information Appliances", Proc. IEEE International Symposium on Consumer Electronics (ISCE'04), pp. 219-223, September 2004
- [2] Kay Romer, "The Lighthouse Location System for Smart Dust," *Proc. MobiSys '03*, pp. 15-30, May 2003.
- [3] L. Doherty, K. Pister, and L. El Ghaoui, "Convex Position Estimation in Wireless Sensor Networks," Proc. IEEE Infocom 2001, April 2001.
- [4] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less Low Cost Outdoor Localization For Very Small Devices," IEEE Personal Communications, Vol. 7, No. 5, October 2000.
- [5] D. P. Robinson & I. W. Marshall, "An Iterative Approach to Locating Simple Devices in an ad-hoc Network," Proc. London Communications Symposium 2002, London, September 2002.
- [6] J.Hightower, G.Borriello and R.Want, "SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength," UW CSE Technical Report #2000-02-02, February 2000.

- [7] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," Proc. ACM/IEEE MobiCom 2000, August 2000.
- [8] Y. Ko and N. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," Proc. ACM/IEEE MobiCom'98, October 1998.
- [9] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)," Proc ACM/IEEE MobiCom'98, pp. 76-84, October 1998.

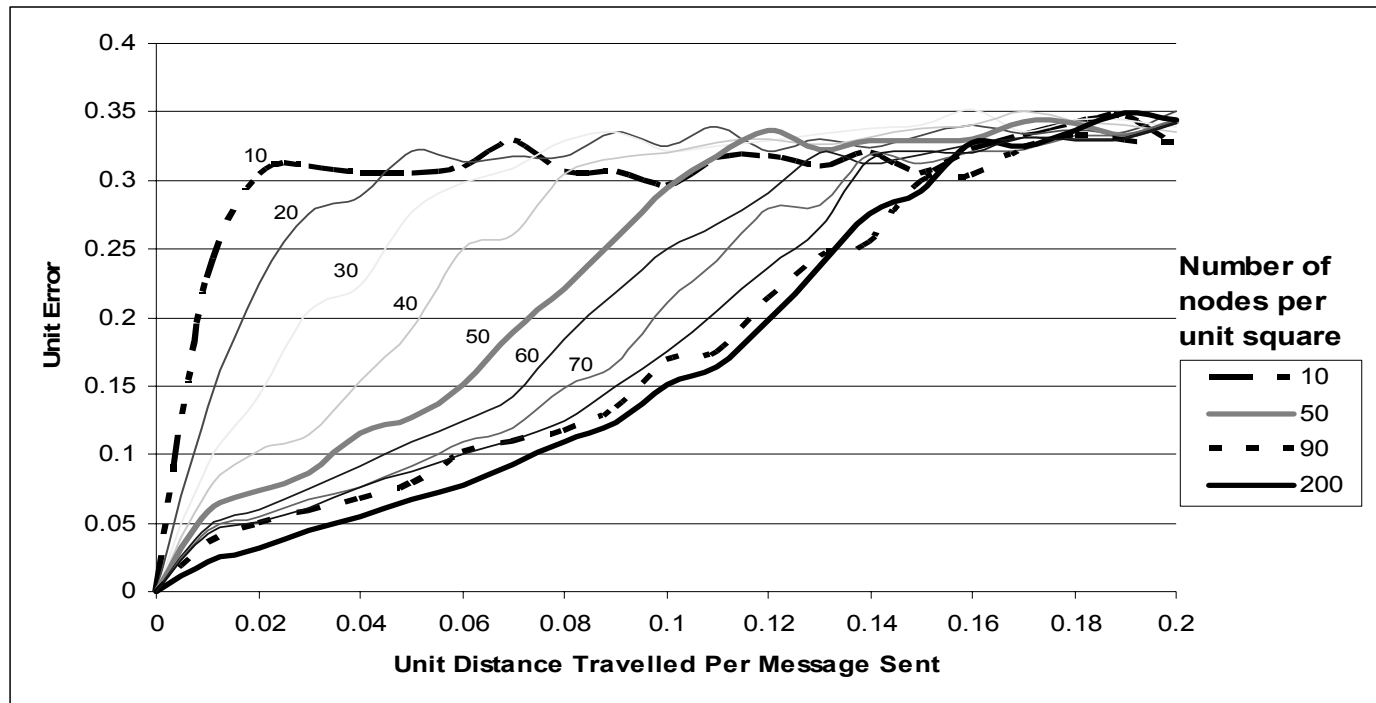


Figure 3. Simulation results showing the aligned error

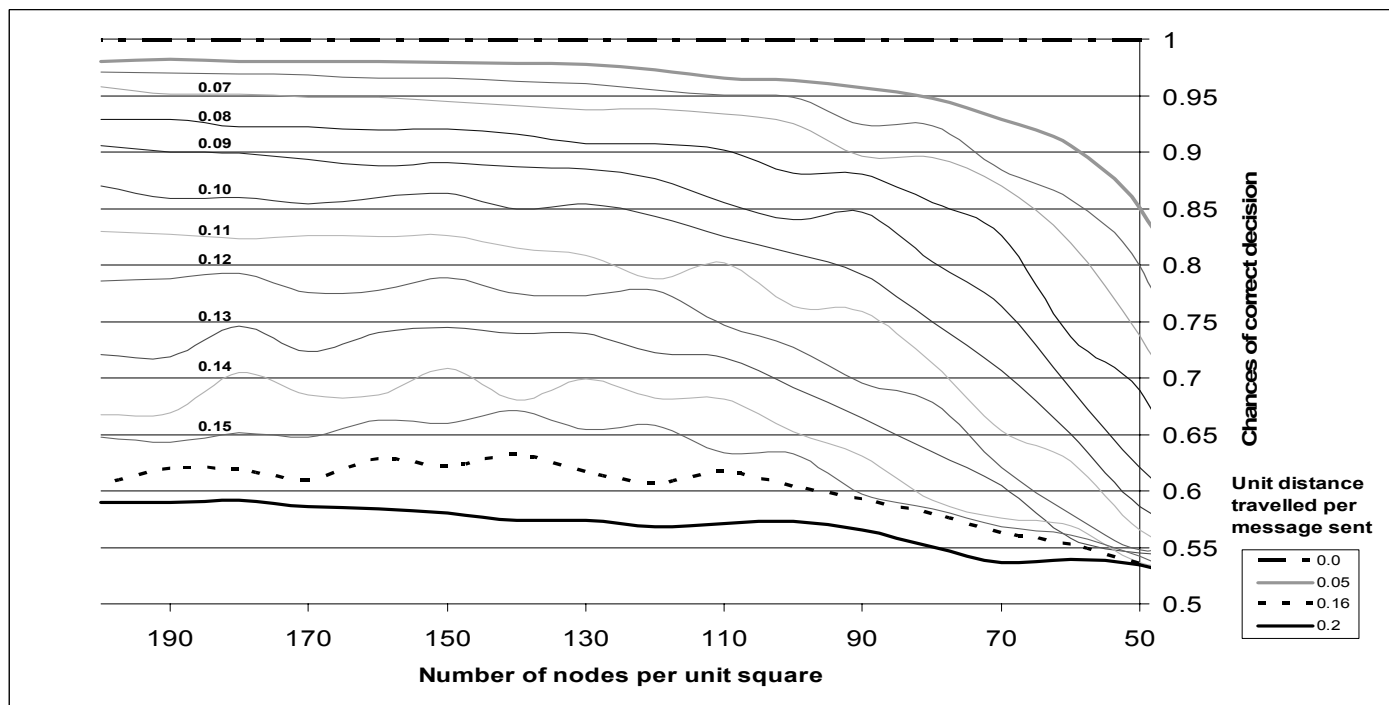


Figure 4. Simulation results showing the percentage of correct decisions made by Trajectory-Based Routing using the estimated logical locations