DECOR: Distributed Construction of Load Balanced Routing Trees for Many to One Sensor Networks

ARTICLE *in* AD HOC NETWORKS · MAY 2014

Impact Factor: 1.53 · DOI: 10.1016/j.adhoc.2013.12.014

2 AUTHORS:

Anthony Kleerekoper
Manchester Metropolitan University
11 PUBLICATIONS **7** CITATIONS

SEE PROFILE

Nick Filer
The University of Manchester
37 PUBLICATIONS **100** CITATIONS

SEE PROFILE

DECOR: Distributed Construction of Load Balanced Routing Trees for Many to One Sensor Networks

A. Kleerekoper[a,*], N. P. Filer[a]

[a]*School of Computer Science, The University of Manchester, Oxford Road, M13 9PL, Manchester, UK*

Abstract

Many sensor networks suffer from the energy hole problem which is a special case of load imbalance caused by the funnelling effect of many sensor nodes transmitting their data to a single, central sink. In order to mitigate the problem, a balanced routing tree is often required and this can be constructed with either a centralised or distributed algorithm. Distributed solutions are typically less effective but are significantly cheaper than centralised solutions in terms of communication overhead and they scale better for the same reason.

In this paper we propose a novel distributed algorithm for the construction of a load balanced routing tree. Our proposed solution, Degree Constrained Routing, is unique in that it aims to maximise global balance during construction rather that relying on rebalancing an arbitrary tree or only maximising local balance. The underlying principle is that if all nodes adopt the same number of children as each other while the routing tree grows, then the final tree will be globally balanced. Simulation results show that our algorithm can produce trees with improved balance which results in lifetimes increased by up to 80% compared to the next best distributed algorithm.

Keywords: Distributed Algorithms; Energy Hole Problem; Lifetime Maximisation; Load Balancing; Routing; Sensor Networks

1. Introduction

It is well known that sensor networks can suffer from a funnelling effect brought about by a many to one traffic flow. The resulting energy-hole problem, which was first analysed by Li and Mohapatra [1, 2], leads to the premature partition of the network when the vast majority of the nodes have energy remaining but the nodes forming the link between the sink and the rest of the network have all depleted their reserves. An energy hole is said to have formed which cuts the sink off from the rest of the network.

The ideal scenario is for all nodes to deplete their batteries at roughly the same time and numerous proposals have been made to ensure this happens. They roughly fall into two categories: in some way rotating the group of nodes that form the bridge between the sink and the network; and adding extra resources to those nodes that do form the bridge. Both of these approaches require making physical changes to the network so that it is no longer the homogeneous, static and uniformly distributed group of nodes that traditionally characterises sensor networks.

For traditional sensor networks, Stojmenovic and Olariu have proven that the energy hole problem cannot be solved completely and that the nodes closest to the sink will always run out of energy before the rest of the network [3]. In this case, the nodes closest to the sink are the critical nodes and the network lifetime is maximised by minimising their workload. Since the total number of packets that need to be transmitted to the sink is fixed, minimising the workload of the critical nodes amounts to balancing the fixed workload between them.

*Corresponding author
Email addresses: kleereka@cs.man.ac.uk (A. Kleerekoper), nick@cs.man.ac.uk (N. P. Filer)

Load balancing is therefore a crucial issue for many to one sensor networks. Centralised solutions have been proposed and, although extremely effective, they are expensive in terms of computation and communication. Some distributed solutions have also been proposed but these are either rebalancing algorithms, which can be expensive and sometimes poorly performing, or they focus only on local balancing and not global balancing.

Dynamic routing can also achieve load balancing through continual monitoring of current workloads and shifting work to under-utilised nodes. In relatively stable networks, however, where nodes are static and link qualities are rarely changing, the overhead of dynamic routing becomes expensive when compared to a routing tree approach.

In this paper we therefore propose a novel routing tree construction algorithm which we call Degree Constrained Routing (DECOR). DECOR is a distributed solution that aims to construct a globally balanced routing tree from the outset without relying on rebalancing an arbitrary tree. Our main contributions are:

- We show that dynamic routing solutions require significant added overhead when compared to routing tree solutions and that in many cases dynamic routing is not required
- We propose a novel routing tree construction algorithm, DECOR
- We show that DECOR can achieve higher balance and longer lifetime than the next best distributed tree construction algorithm
- We show that our proposed algorithm can be effectively adapted to networks with a non-uniform distribution of nodes

2. The Corona Model and Load Balancing

The corona model has been widely used to model sensor networks because its simplicity allows strong mathematical analysis though it is not always referred to as the "corona model". The model, illustrated in Fig. 1, consists of a single, central sink surrounded by a large number of sensor nodes that are randomly and uniformly distributed throughout a circular area. All nodes, including the sink, are assumed to have the same radio transmission range which extends in a circle around each node, termed the node's *reachable area*. The model is used to prove that nodes in corona x are always x hops from the sink and therefore in level x of a shortest path routing tree [4, 5]. While this is true for the majority of the nodes, the relay hole problem can result in some variation [6].

The corona model facilitates discussion of the energy hole problem and load balancing. The many to one traffic pattern causes a funnelling effect whereby increasingly more data is forwarded through fewer and fewer nodes. The nodes in the inner-most corona perform the most forwarding and therefore deplete their batteries first causing a partition of the sink from the rest of the network. This is the energy hole problem which is a special case of load imbalance.

Using the corona model it is possible to define four types of load balancing that can be achieved in a sensor network. Zhang and Shen have defined the two main types of load balancing as intra-corona and inter-corona [7]. In the first type, all nodes in the same corona have the same load but the load may be different between different coronas. In the second type, the load between coronas is the same as well so that all nodes in the entire network have the same load.

Intra-corona balancing is a prerequisite for inter-corona which is obviously the ideal form of load balancing. However, inter-corona balance can only be achieved in some cases and always by making changes to the physical network, eg by adding more nodes in specific areas. When the network is a traditional, uniform distribution of static, homogeneous nodes then Stojmenovic and Olariu have proven that inter-corona balance is not possible [3].

Although balance should be measured in terms of workload it requires global knowledge to calculate how much work every node must perform. Some researchers have therefore used the node degree as a proxy for the workload since this requires only local knowledge. We refer to intra-corona balance that uses node degree as its measure of work as *degree balancing*.

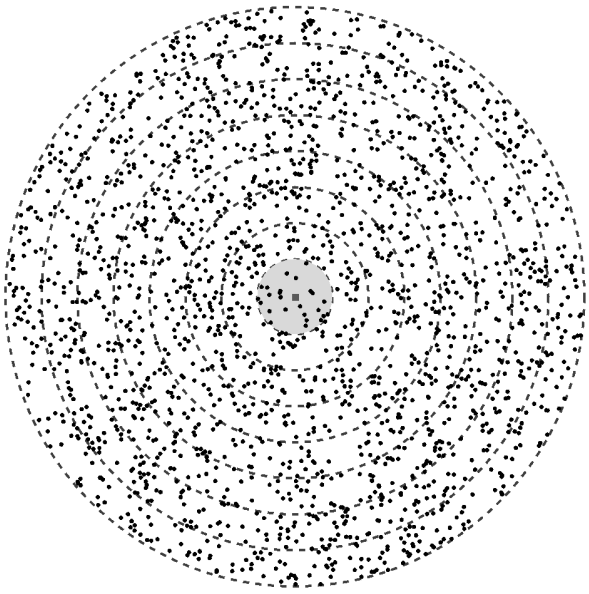

Figure 1: A sensor network can be modelled as a series of concentric circles or coronas with the sink (shown as a small square) in the centre. The nodes inside the central shaded area are the nodes in the inner-most corona that have the highest forwarding burden.

When inter-corona balance is not possible, the network lifetime is maximised by maximising the lifetime of the inner-most coronas only, regardless of the balance in other coronas. This is because it will always be a node in the inner-most corona that dies first and the total workload of the nodes in the inner-most corona is fixed. Therefore, to maximise the time until the first node dies, the balance among the nodes in the inner-most corona must be maximised. This leads to the fourth type of load balancing, namely *inner-corona balancing* in which the aim is to maximise the balance only of the nodes in the inner-most corona without concern for the other coronas.

The aim of this paper is to propose a novel, distributed tree construction algorithm that maximises inner-corona balance. It is important to note that although we initially adopt the corona model with its strong assumption of omnidirectional antennas and uniform propagation, we relax this assumption later and consider a more realistic propagation model.

3. Related Work

Routing techniques for sensor networks can be divided into two groups: hierarchical and flat [8]. In hierarchical routing different nodes play different roles with the network divided into clusters and some nodes acting as cluster heads. In flat routing all nodes play the same role and together form a routing tree. Flat routing techniques can further be divided into two groups: dynamic and stable. Although both techniques rely on routing trees, dynamic approaches allow nodes to switch between different parents in the tree relatively frequently meaning that the routing tree can change very quickly. Stable approaches, on the other hand, require nodes to utilise the same paths for extended periods of time, typically until failures completely prevent the use of that path.

One of the first dynamic routing methods was proposed by Shah and Rabaey [9]. In dynamic routing every node maintains a routing table with a list of potential parents and a path cost associated with each option. For every packet that needs to be relayed the nodes select one of the potential parents. The metrics used for calculating costs usually include some measure of workload which enables load balancing. Examples of such cost measures include the residual energy of the parents [9] and the ratio of the number of relayed data packets to locally generated packets [10]. By switching between parents to use the least loaded parent, inner-corona balance can be maximised. The localised nature of the switching may result in slow responses to imbalance.

Dynamic routing is absolutely necessary for dynamic networks in which there is either significant mobility among the nodes or variability in the radio link qualities [11]. However, it comes at the cost of increased overhead as nodes must maintain accurate costs for every parent meaning that there is added communication and potentially large routing tables. These extra costs somewhat limit the scalability of dynamic approaches as the cost increases very quickly with increased density. Since in sensor networks the control packets are expected to be a similar size to the data packets [12], the communication requirements of dynamic schemes can be expensive in energy terms.

In static sensor networks where the nodes do not move and radio link quality is relatively stable over long periods (as it is when high quality links are used [13, 14]) then stable routing trees should be used so that after an initial burst of communication to establish paths, only data packets require communication [11].

Stable routing tree solutions can be divided into two groups: centralised and distributed. Centralised solutions are generally more effective at creating a balanced tree because they have access to global information but they are more complex requiring significant computation and memory capacity. Moreover, accurate information about the entire network must first be collected and since there is no routing tree in place yet, this collection requires some form of flooding which is very expensive. Distributed solutions require far less resources (computational, memory and communication) but the resulting routing trees are usually less well balanced. However, the two can be used in combination with a distributed algorithm used to create an initially balanced routing tree at relatively low cost which can then be queried and rebalanced with a centralised algorithm at lower cost than running the centralised algorithm on its own.

The first work in this area is from Hsiao *et al.* who proposed a centralised algorithm for increasing the inner-corona balance of an arbitrary routing tree [15]. Their algorithm is iterative and in each iteration every node is examined to see if the balance of the tree can be improved by switching it (and its descendants) to a different parent. The switch that has the largest improvement in balance is made for that iteration. Hsiao *et al.* used Jain's fairness index as their balance metric because it is sensitive to all nodes and is strictly monotonic [16]. We will describe this metric more fully in section 7. Buragohain *et al.* proposed a similar algorithm but to calculate which node to switch they model the lifetime of the tree before and after each potential switch.

The centralised algorithms focus exclusively on inner-corona balance. To the best of our knowledge the only distributed algorithm that aims to construct a routing tree with high inner-corona balance was proposed by Chen *et al.*, called Adjustable Convergecast Tree (ACT) [17]. ACT assumes that a shortest path routing tree already exists and rebalances it in a distributed fashion. The rebalancing is conducted bottom-up from the leaf nodes to the sink. In each iteration a node (known as the grandparent) examines its subtree by querying its children (known as the parents) to find out how big their subtrees are. This information is used to calculate the grandparent's load balancing factor (LBF) which is the min/max ratio of the parents' subtrees. If the LBF is greater than one then the grandparent will attempt to increase the balance of its subtree. It does this by querying the parents to find out which of their children (known as the grandchildren) can be moved to which other parent's subtree. The grandparent then instructs one or more grandchildren to switch subtree in order to maximise the LBF. Once a grandparent has maximised the balance of its own subtree it informs its parent and becomes a parent in the next iteration. The process completes when the sink finishes rebalancing its subtree which is, of course, the routing tree.

In the original ACT, grandchildren are only moved from the heaviest loaded subtree to the lightest which is an obvious limitation. Even allowing for more complex changes, however, the overhead in terms of communication cost is significant as lots of information must be passed from grandchildren to parents to grandparent and back again. A more serious concern is that ACT only allows movements of grandchildren which limits the granularity of changes that are allowed. The largest drawback with ACT is that nodes cannot switch from one ancestor to another. Therefore, except for the changes made by the sink, no other changes affect the number of descendants of the nodes in the inner-most corona. Since it is these nodes that define the expected lifetime, ACT is limited in its effectiveness.

There have also been three distributed algorithms that aim to maximise degree balancing. A node's degree is not a complete measure of its work load because it only measures the number of children a

node has whereas its total load depends on the number of descendants it has. However, calculating the degree requires only local information whereas its load requires more global information. In a distributed algorithm it is therefore cheaper to work with a node's degree than its load and use the degree as a proxy for work load.

One such algorithm was proposed by Andreou *et al.* called Energy-Driven Tree Construction (ETC) [18]. The idea behind ETC is that if all nodes have the same number of children then balance is guaranteed and all nodes could have the same number of children if all were in direct communication with each other in which case well-known balanced tree algorithms could be used such as AVL Trees or B-Trees. In a balanced tree every node has the same number of children which is the branching factor, Φ, and can be calculated from the depth of the tree, Δ, and the number of nodes, N:

$$\Phi = \sqrt[\Delta]{N} \tag{1}$$

In ETC, a shortest-path routing tree is initially constructed and then queried by the sink to find Δ and N from which the branching factor is calculated. The branching factor is then transmitted throughout the network and every node is responsible for attempting to ensure that it has as close to Φ children as possible. When a node finds it has too many children it will instruct some of them to switch to new parents.

A second degree balancing algorithm is Minimum Balanced Tree (MBT) proposed by Huang *et al.* [19]. MBT is an iterative algorithm in which, in each iteration, every node queries its neighbours to discover their level in the routing tree and their degree. A node will select, as its parent, the neighbour with the lowest level which will be the neighbour closest to the sink. When more than one neighbour has the same level (which is likely) then the node with the lowest degree is chosen. In this way a shortest-path routing tree is constructed in which nodes have minimised degree.

The final algorithm was proposed by Chatzimilioudis *et al.* and is called Minimum HotSpot Query Routing Tree (MHS) [20]. MHS constructs a routing tree level by level with only a single iteration per level. In each iteration the nodes that joined the routing tree in the previous iteration broadcast an advert which is received by their potential children. Each potential child may receive a different number of adverts depending on how many parent options it has. The potential children then select their parents sequentially with the children with the most choices waiting longer before making their choice. When a node has its turn to choose its parent it selects the parent with the lowest node degree. The chosen parent broadcasts its new degree and the children waiting to select parents update their records accordingly.

Previous experiments showed that MHS, despite only aiming to maximise degree balancing, was the distributed algorithm that produced the highest inner-corona balance [21]. It also has the lowest overhead of the distributed algorithms and we therefore utilise MHS as the bench-mark for our proposed solution in section 7. We also use the centralised algorithm of Hsiao *et al.* to provide an upper-bound on the level of inner-corona balance that is achievable.

Our proposed solution is unique in that it is distributed, aims to maximise inner-corona balance and does so without rebalancing. Moreover, it is the first solution to trade-off tree depth (latency) for improved balance.

4. Inner-Corona Balance Through Adoption Limitations

Our approach to creating a routing tree with inner-corona balance is motivated by the observation that if every node in the network had the same number of children then the routing tree would be perfectly balanced. This is the idea behind the ETC algorithm proposed by Andreou *et al.* [18], described in Section 3. The problem is that in a shortest path routing tree constructed over a uniformly distributed network, it is impossible for all nodes to have the same number of children. Macedo analysed the average number of children per parent in such a routing tree and found that the number varies with the corona number [22]. The average number of children per parent in corona i, C_i, is:

Level	Children per Parent	Number of Nodes in the Level
1	3	n
2	1.67	$3n$
3	1.4	$5n$
4	1.29	$7n$
5	1.22	$9n$
6	1.18	$11n$
7	1.15	$13n$
8	1.13	$15n$
9	1.12	$17n$
10	1.11	$19n$

Table 1: Values derived from equation (2) showing the average number of children per parent and the number of nodes in each level, where n is the number of nodes in level 1.

$$C_i = \frac{2i+1}{2i-1} \tag{2}$$

Table 1 shows the average number of children per parent and the number of nodes in each level of the routing tree relative to the number in the inner-most corona, n. It is immediately apparent that, aside from the nodes in the first level, the nodes cannot all adopt the same number of children because the average number of children per parent is a fraction. Instead what must happen (in the best case) is that some nodes must adopt two children and others adopt only one and this causes imbalance because some nodes in the inner-most corona end up with more descendants than others. Of course there is no guarantee that the division will be so small and it is possible for some nodes to adopt three, four or more children while others adopt none which increases the extent of the imbalance.

The conclusion we draw from this is that in order to construct a balanced routing tree we must either drop the requirement that it be a shortest-path tree (ie sacrifice latency) or else drop the requirement for all nodes to connect to the tree (ie sacrifice connectivity). In our approach we start by sacrificing connectivity and then develop the algorithm so that connectivity is maintained at 100% and only latency is sacrificed.

To avoid inner-corona imbalance and attempt to ensure that all nodes adopt the same number of children, our proposed solution constructs a routing tree with rules limiting the number of children that may be adopted by each node. We call our solution Degree Constrained Routing (DECOR) because the underlying idea is that as the routing tree is constructed, the degree of every node is constrained so that, ideally, all nodes adopt the same number of children.

In the next sections we more fully describe the DECOR algorithm in an idealistic scenario before discussing practical issues and the changes we made to the basic algorithm to tackle those issues.

5. Degree Constrained Routing

In this section we describe the DECOR algorithm in an idealistic scenario where the distribution of nodes is perfectly uniform so that the number of nodes in each corona precisely matches the analysis of Macedo [22]. We also assume the simplistic unit disk model in which every node has a fixed, circular transmission range surrounding it and in which all links are perfect. In later sections we relax these assumptions and make minor modifications to DECOR to accommodate the more realistic scenario. In order to isolate the effects of the routing method from the MAC layer we assume that there are no collisions and no retransmissions. This is a reasonable assumption for static sensor networks using stable routing trees which will generally have low duty cycles enabling Time Division Multiple Access (TDMA) which avoids collisions. Additionally, high quality links have been shown to be stable over long periods (see [13]) leading to very few retransmissions and for energy efficiency only high quality links should be utilised [14].

Corona Number	Number of Nodes	Limit	Connected in Corona	Disconnected in Corona	Connectivity
1	n	3	n	0	100%
2	$3n$	1	$3n$	0	100%
3	$5n$	2	$3n$	$2n$	77.78%
4	$7n$	1	$6n$	$1n$	81.25%
5	$9n$	1	$6n$	$3n$	76.00%
6	$11n$	2	$6n$	$5n$	69.44%
7	$13n$	1	$12n$	$1n$	75.51%
8	$15n$	1	$12n$	$3n$	76.56%
9	$17n$	1	$12n$	$5n$	75.31%
10	$19n$	1	$12n$	$7n$	73.00%

Table 2: The ideal adoption limit values for DECOR and the effect on connectivity. The highlighted cells illustrate which levels can have increased limits without reducing balance.

The aim of DECOR is to enforce inner-corona balance by constructing a routing tree in which every node in a given corona has the same number of children as every other node in that corona. If this is achieved then the number of descendants of the nodes in the inner-most corona will be identical and therefore their workloads will also be identical.

As mentioned, the starting point for DECOR is to prioritise balance and sacrifice connectivity. Balance is achieved by insisting that all parents in a given corona adopt the same number of children as each other even though this will prevent some nodes from connecting to the routing tree. Later we modify DECOR to regain connectivity at 100% and instead sacrifice latency.

A naive approach would be to limit the number of children that a parent can adopt to $\lfloor C_i \rfloor$ but this would mean that, apart from the first corona, all parents adopt only a single child in all coronas since $\lfloor C_i \rfloor = 1 \ \forall i > 1$. The proportion of nodes that would be unable to connect to the routing tree would be high and become increasingly higher as the network size increased. However, imposing an adoption limit means that the actual number of nodes in the tree is different to the number of nodes in the corona and therefore Macedo's analysis is no longer accurate when determining C_i. For example, Macedo's analysis states if there are n nodes in the inner-most corona then there would be $5n$ nodes in the third corona and $7n$ in the fourth. This means that the parents in the third corona have an average of 1.4 children each. But if the nodes in the second corona were only allowed to adopt one child each then there would, in fact, be only $3n$ nodes in the third corona that are connected to the routing tree. Since there are still $7n$ nodes in the fourth corona this leads to an average of 2.33 children per parent. Those nodes may be allowed to adopt two children each without reducing balance.

DECOR takes advantage of this fact to increase the limit whenever the number of children per parent available for adoption increases to greater than two. When this happens parents may adopt two children each without reducing balance. Table 2 shows the adoption limits for parents in different coronas and the proportion of nodes connected to the routing tree. Some cells have been highlighted to show where the limit can be increased without reducing balance. We chose to increase the limit at the earliest opportunity because every corona with a lower than necessary limit reduces the number of nodes connected to the routing tree. Extrapolating from the table we can find the levels in the routing tree which can have increased limits, namely $\{3, 6, 12, 24, \dots\}$, with a limit of three children per node in level one.

The algorithm grows the routing tree iteratively starting with the sink and then level by level, with negotiation between parents and children. The parents are the nodes that joined the routing tree in the previous iteration and the children are all nodes that are in direct communication with the parents. At the beginning of the iteration all parents broadcast an advert packet which contains the information necessary for children to communicate with them including IDs and perhaps scheduling information. Where possible the advert should also contain location information which can be used by the children to select a parent. The children gather all these advert packets and create a list of potential parents.

In the next phase of the algorithm the children select the best parent from their list of options. To maximise energy efficiency and minimise the relay hole problem (see [6]) children should choose the parent that is physically closest to the sink. If location information is available then this can be used to calculate the distances between the potential parents and the sink. If no such information is available then parents are chosen at random. The children transmit an adoption request packet to their chosen

parent which includes information such as the node ID, scheduling information (where required) and location information (where available).

Parents gather all the adoption requests sent to them and construct a list of potential children, ranking them according to their distance from the parent. The parents then select the top q children where q is the parent's quota or adoption limit. The parents broadcast an adoption confirmation packet listing their adopted children and how many more children, if any, they can adopt. The adoption confirmation serves also as a rejection for the children who were not adopted by that parent and all children update their list of potential parents according to the contents of the packets.

Any children that were rejected by their chosen parent must choose again and transmit another adoption request. The process of adoption request and confirmation continues until all children have been adopted. This process cannot be coordinated and so a fixed number of rounds must be assumed by all nodes. Since the network is uniformly distributed, all nodes will have approximately the same number of neighbours although not all of these will be potential parents. The sink is aware of the approximate number of neighbours shared by all nodes because all of its neighbours become its children. The sink can therefore pass on this information to the nodes and this number can be used as the number of rounds that the nodes leave for adoption requests and confirmations. For many of these rounds the nodes will not be doing anything which wastes time but not energy.

Once the number of negotiation rounds has been reached the iteration ends and any children that were unable to connect to the routing tree remain disconnected. All other children become parents for the start of the next iteration.

The number of packets required per node for parent-children negotiation depends on the number of neighbours each node has. However, because of the low duty cycle and the use of only high quality links, the overhead is relatively small and is a one-off cost which becomes very small when compared to the lifetime of the network.

Table 2 shows the affect that the adoption limits have on connectivity which falls to as low as 69.44%. This is only our starting point and in the next section we make minor changes to DECOR that make it more appropriate for practical situations and which also increase connectivity to 100%.

6. Practical Issues and Solutions

The algorithm described above for DECOR was based on the analysis of Macedo concerning shortest path routing trees constructed over uniformly distributed networks [22]. His analysis relied on the key assumption that the nodes are distributed in perfect accordance with a uniform distribution such that the number of neighbours per node is constant for all nodes. In practice this is very unlikely to be the case and the result is that different nodes have a different number of neighbours capable of acting as parents in the routing tree. The consequence of this is that there is no guarantee that nodes will be able to adopt as many children as they have been allowed to because they may not have enough potential children to do so.

It is worth noting that even if the nodes are distributed uniformly, the relay hole problem will prevent some nodes from finding parents in the next inwards corona [6]. Moreover, the length of high quality radio links vary which will also result in some nodes having more potential children than others.

There are two changes we make to the basic algorithm in order to reduce the impact of the variation in the number of neighbours. The first and most important is to reduce the adoption limit for the nodes in the inner-most corona from three to two children each. This provides lee-way so that there can be differences in the number of potential children and yet nodes all adopt the same number of children. This is most important because any differences in the number of children adopted by the nodes in the inner-most corona become amplified by adoptions in subsequent coronas and can lead to very large imbalances.

The consequence of this first change is that connectivity falls because not all nodes in the second level may join the tree. Soon, however, we will discuss how to increase the connectivity to 100% without

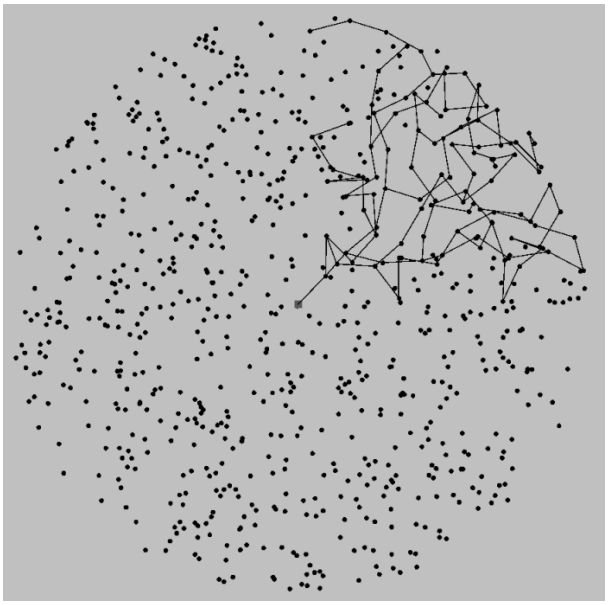

Figure 2: The subtrees may contain "twists" because nodes become children of nodes further from the sink than themselves.

significantly impacting on balance. Another consequence of this change is that the number of nodes connected to the tree in each corona is different and so, using a variant of Table 2, the coronas in which the adoption limit can be raised changes to $\{2, 4, 8, 16, \dots\}$.

The second change is to the method of ranking potential children. Since different children will have different numbers of potential parents it is beneficial to prioritise the children with the fewest options. The children with the most options can be rejected by their first choice and still have alternatives whereas the children with few choices may find themselves unable to connect to the routing tree if they are rejected by their first choice. Therefore, when the children transmit their adoption requests they include the number of potential parents they have in the packet. The parents then rank their potential children primarily by this number and choose the child with the fewest parent choices. Only in the case of a tie do they use their distance from the children as a ranking measure.

This method of ranking potential children was also used by Chatzimilioudis *et al.* [20]. Ironically, ranking potential children this way may result in nodes with very few options being adopted and nodes with lots of options never being adopted. Since all nodes are identical, however, it makes no difference which nodes are adopted and which are not so long as connectivity and balance are maximised. Ranking nodes in this way makes it more likely that parents can fill their quotas which increases balance and connectivity and is therefore desirable.

We now turn our attention to modifying DECOR to achieve 100% connectivity. As discussed, DECOR misses out some nodes during construction because parents are limited in the number of children they may adopt. The results in Table 2 suggest that as many as 30% of the nodes may be unable to connect to the routing tree. Those results, however, assume that once a node misses its first opportunity to connect it will never again be able to do so.

If parents may only adopt children which are physically further from the sink than they are, then once a node misses its first opportunity to join the tree then it will never be given a second chance. This restriction, however, is not required and, in some cases, it may not be possible to even identify whether a child is closer to the sink than the parent or not. Therefore, it turns out that nodes may get more than one opportunity to join the routing tree.

Fig. 2 shows an example of a subtree rooted at a level one node that was constructed by the DECOR algorithm as presented above. It lacks the classical tree structure because nodes that were unable to connect at their first opportunity connected to the tree at a later stage. The branches of the subtree do

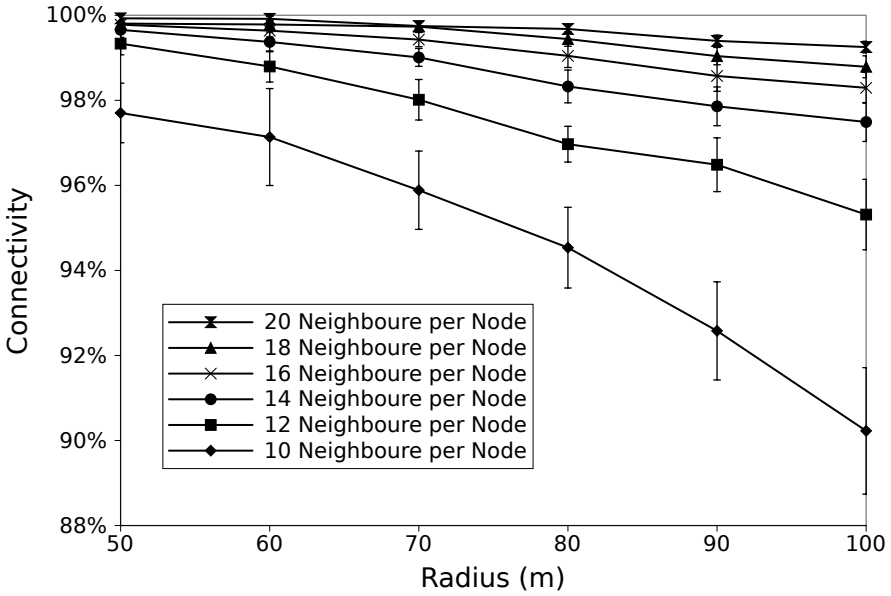

Figure 3: In simulations the connectivity with DECOR was much higher than predicted by the theory. This is because the routing tree grew back towards the centre of the network connecting to the previously disconnected nodes.

not grow straight from the centre to the outer edge but often twist back towards the centre to connect to nodes that missed their first opportunity. As Fig. 3 shows, this results in connectivity levels which are significantly higher than those predicted in Table 2.

It turns out that DECOR sacrifices far less connectivity than previously thought but sacrifices some connectivity as well. The average path length is increased because nodes have connected to the tree at levels which are higher than their physical distance from the sink would require. In other words, a node in corona five, for example, would normally be expected to join the routing tree in level five but in DECOR it may join the tree many levels further away. For convenience we refer to nodes that are in the "incorrect" level of the routing tree as *misplaced nodes*.

We introduce a second phase that is capable of reducing the added latency and guaranteeing that connectivity is maximised. They key observation motivating this second phase is that inner-corona balance is determined only by the number of descendants each of the nodes in the inner-most corona have. If we let each node from the inner-most corona be the root of a subtree then balance is determined by the sizes of these subtrees and is independent of their length. Many of the misplaced nodes will be within direct communication range of other nodes from the same subtree which are higher up the routing tree than the misplaced node. Allowing misplaced nodes to switch parents to another node that is higher in the tree but from the same subtree will reduce the average path length without reducing balance at all because the subtree sizes remain the same. This is the essence of the second phase of DECOR.

In order to achieve this, during the first phase the sink assigns a unique subtree number to each node in the inner-most corona which are also the sink's children in the routing tree. Every node descended from the same inner-most corona node will have the same unique subtree number. After the routing tree is constructed by DECOR the sink starts the second phase which constructs the routing tree again. Nodes "forget" their parents and children but remember which subtree they were in at the end of the first phase. During the second phase there are no limits to the number of children a node may adopt and child nodes join the routing tree at their first opportunity. The only caveat is that a node can only join the routing tree in the same subtree as it was in at the end of the first phase. This means that the subtree sizes remain unchanged but misplaced nodes can join the routing tree closer to the sink.

During this second phase we can also achieve guaranteed maximum connectivity. Any nodes that did not join the tree at all during the first phase are free, during the second phase, to join whichever subtree

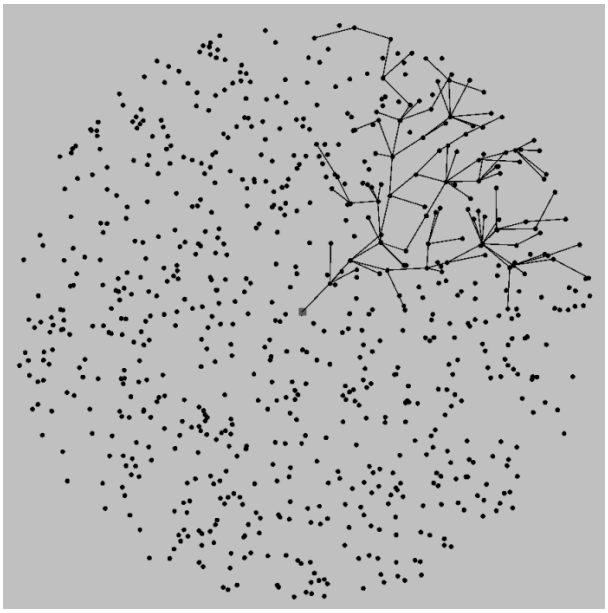

Figure 4: A second phase is introduced into DECOR to remove the twists without compromising balance.

they want. In our experiments we use a first come first served approach so that previously disconnected nodes become children, during the second phase, of whichever parent they hear from first. Experiments showed that more complex schemes increased the communication overhead significantly with insignificant improvements to balance or latency. By allowing previously disconnected nodes to join the tree during the second phase, we guaranteed that any node that can connect to the routing tree (ie any node which has a potential path to the sink) will connect by the end of the second phase.

Fig. 4 shows the same subtree as Fig. 2 but after the second phase of DECOR has completed. The subtree now looks like a traditional tree because the elongated paths have been removed as misplaced nodes connected much earlier.

7. Experimental Setup, Metrics and Results

In this section we present a number of experimental results showing the performance of DECOR. Initially we show results for DECOR in a somewhat idealistic scenario with a uniformly distributed network (although not a perfectly uniform one) and using the unit disk model in which the transmission range of all nodes is a perfect circle of fixed range. Later on, however, we repeat the experiments with a realistic model of radio link quality and with networks with more uneven node distributions.

In all cases we assume a circular network with a single, central sink. In order to isolate the routing protocol's effects from other factors we assumed an ideal MAC layer which completely avoids collisions and retransmissions. Although this is a strong assumption, we note that recent research has shown that energy-efficient networks should utilise only high quality links which have low probability of retransmissions [14]. We also note that sensor networks are likely to have low duty cycles allowing the use of TDMA scheduling where collisions can be minimised. Techniques for avoiding or minimising collisions are outside the scope of this work but we note that the additional overhead of DECOR will be exacerbated if the collision rate is high.

The transmission range of the nodes was set to 10m and the circular network radius was varied from 50m to 100m in 10m intervals. Additionally, the network density was varied from 10 neighbours per node to 20 neighbours in intervals of two. Each configuration was repeated 25 times and the results presented are the average of those runs with 95% confidence intervals. The results are presented compared to MHS (see Section 3) which we found, in previous work, to be the best performing distributed algorithm in terms of both balance and communication overhead [21]. We also compare to the centralised rebalancing

algorithm of Hsiao *et al.* noting that our implementation is incomplete as we did not include the simulated annealing element that Hsiao *et al.* mention only in passing. The result is that the centralised algorithm can become stuck at local maxima. Since we intend to use this algorithm as an upper-bound on the balance and max/mean ratio, we use the single best result over the 25 runs for our results.

We followed the approach of Hsiao *et al.* and measured the balance of the routing tree using Jain's fairness index [16] (which Hsiao *et al.* call the *balance index*). The index, shown in equation (3), has a number of desirable properties for measuring how equal a distribution of weights is. The first is that it is sensitive to all values so that any change in the distribution causes a change in the index. Secondly, the metric is continuous and monotonic which allows it to be used to measure whether a change increases balance or not. Finally, it is bounded between 0 and 1 with 1 indicating perfect balance where all weights are equal. For use in measuring the balance of the routing tree we let the weights be the number of nodes in each subtree rooted at a node in the inner-most corona. Hereafter, for simplicity, when we talk of a subtree we are referring to one whose root node is in the inner-most corona.

$$\beta = \frac{\left(\sum_{i=1}^{n} w_i\right)^2}{n \sum_{i=1}^{n} w_i^2} \tag{3}$$

Although the balance index is very useful it does not provide an accurate measure of the lifetime. This is because the lifetime of the network is determined by the shortest living node which is the root of the heaviest subtree. Since balance can be improved without reducing the size of the heaviest subtree it is obvious that the balance index alone cannot predict lifetime exactly. A second metric is used to indicate lifetime - the mean/max ratio. In a perfectly balanced routing tree all subtrees have the same weight which is the mean weight of all subtrees in any routing tree. As the heaviest subtree becomes heavier, the lifetime of the routing tree falls and it falls in proportion to how much heavier the subtree is than the others. The mean/max ratio directly gives the proportion of the maximum available lifetime that the routing tree achieves. If the tree is perfectly balanced then the heaviest subtree has the same weight as the mean and the ratio is 1. As the heaviest subtree becomes heavier the ratio falls showing a smaller proportion of the maximum available lifetime.

Since the trade-off for balance is latency we also measure this by recording the average number of hops between each node and the sink. Finally, we record the communication overhead of the algorithms as the average number of packets transmitted and received by each node during tree construction. It is worth noting that we also found the connectivity of the network but, as expected, this value was always 100% because of the second phase.

7.1. Balance and Lifetime Results

The results for balance, some of which are shown in Fig. 5, show that the balance of the routing trees produced by DECOR can be significantly higher than that produced by MHS. In all cases the balance is lower when the radius is higher because there are more coronas and this means that the effect of any imbalance is increased. Consider that if a node adopts more children than another then the more descendants those children have the bigger the effect of that original imbalance. For MHS the balance barely varies with density and ranges from an average of 0.66 to 0.52 for networks with a radius of 50m and 100m respectively.

For DECOR the balance actually increases with density because each node has more neighbours and therefore it is easier for the parents to fill their quotas which is necessary to produce balance. DECOR generates trees with an average balance of 0.58, for a network of radius 100m and a density of 10 neighbours per node, and 0.82 for a network of radius 50m and density of 20 neighbours per node. Over all configurations, the balance using DECOR was on average between 1.7% and 52.59% higher than MHS.

The Mean/Max ratio gives the proportion of the maximum lifetime that the routing tree will live. The results in Fig. 6 show that in all cases the lifetime achievable is barely more than half of the maximum lifetime available from a perfectly balanced routing tree. Even a centralised algorithm frequently fails to achieve the same lifetime as a perfectly balanced tree, especially when the density is high. Moreover,

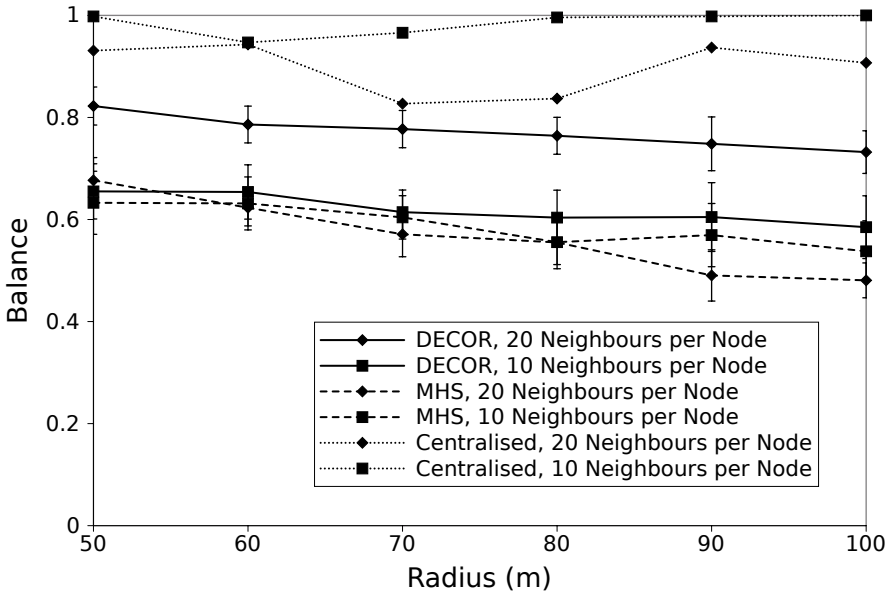

Figure 5: DECOR shows significant improvements in balance at higher densities compared to MHS. At lower densities the improvement relative to MHS is smaller and in some instances MHS may outperform DECOR.

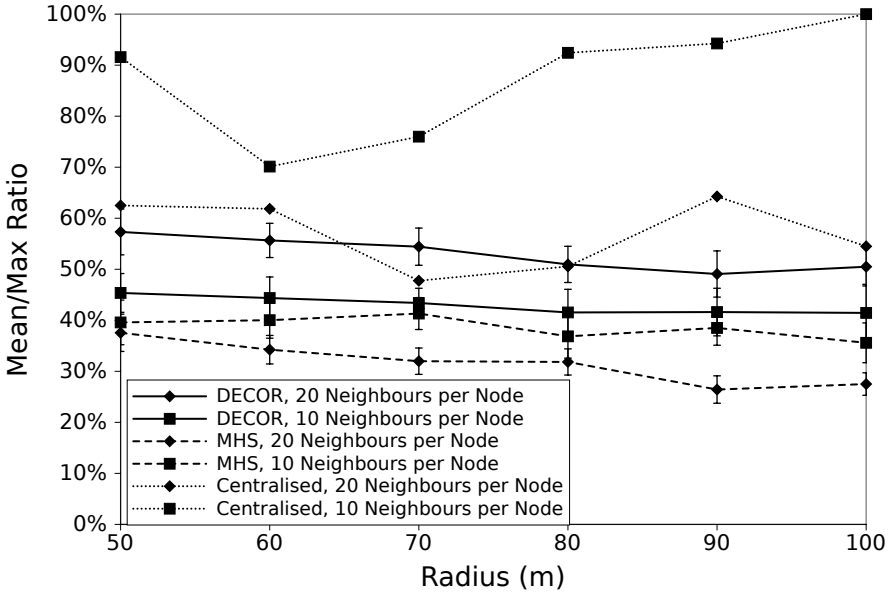

Figure 6: The results for the mean/max ratio again show that the difference between DECOR and MHS is more pronounced at higher densities with DECOR achieving significantly smaller ratios which correspond to significantly longer lifetimes. At high densities DECOR achieves expected lifetimes very close to the centralised algorithm.

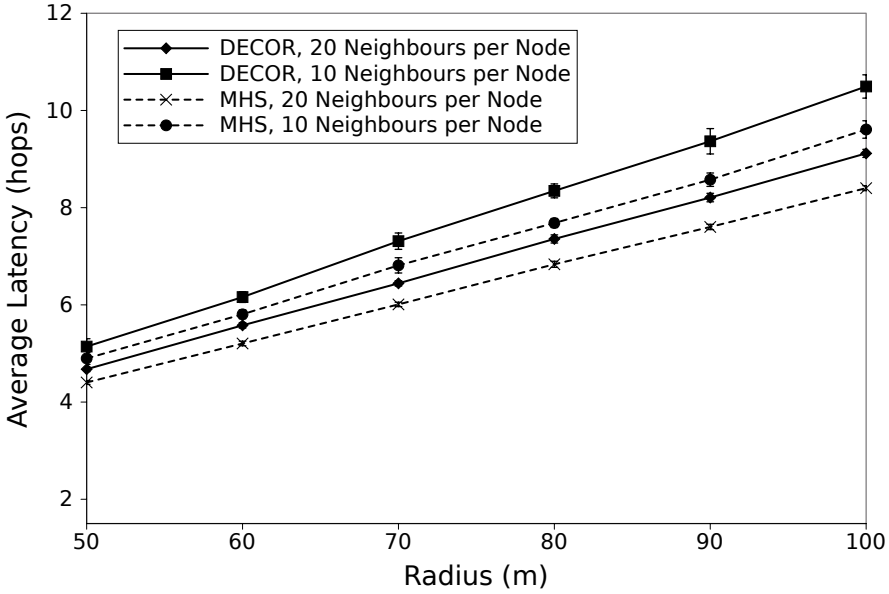

Figure 7: Latency of course increases with the radius of the network. The cost of increased lifetime under DECOR is an increase in latency but the increase is never more than 10%.

they show that the lifetime falls with radius like the balance and, in the case of MHS, falls with density as well. For DECOR, though, the lifetime is longer when the network is more dense which accords with the results for balance.

In the case of MHS, the lifetime of the tree is between 27.5% and 41.32% of the maximum lifetime, compared to ranges of 41.45% to 57.32% for DECOR. Over all configurations, DECOR outperforms MHS by up to 85.66%. This shows that while the lifetime remains far shorter than the maximum available, the new algorithm significantly extends the lifetime of the network compared to the best performing alternative distributed algorithm. In dense networks, DECOR even achieves an expected lifetime very close to the best result of the centralised algorithm.

7.2. Latency and Overhead Costs

Fig. 7 shows the latency of the network in terms of the average number of hops between each node and the sink. Inevitably the latency is higher for larger networks but it is lower for more dense networks which makes sense because more nodes connect in their correct coronas. Latency is the main cost of the algorithm exchanged for the increase in lifetime but nevertheless the cost is relatively small. The maximum increase in latency over the configurations we simulated was 9.21% for DECOR.

The other cost of DECOR is added communication overhead because of the need for some form of negotiation over child adoptions as well as the second phase of tree construction. The number of packets transmitted per node is almost constant under MHS but increases with radius for DECOR, as shown in Fig. 8. In MHS every node only needs to transmit a single adoption request, one advert and then one adoption confirmation per child (and there is usually only one child). This explains why the number of packets transmitted per node under MHS is approximately three per node. For DECOR, on the other hand, there is some negotiation surrounding child adoptions as well as the second phase. As the network gets bigger and balance falls nodes find themselves being rejected more often and therefore there is more negotiation.

Although there is an increase in the number of control packets transmitted per node, over the range of radius values simulated the increase is limited and is always fewer than four extra packets per node. Compared to the lifetime of the network this is negligible.

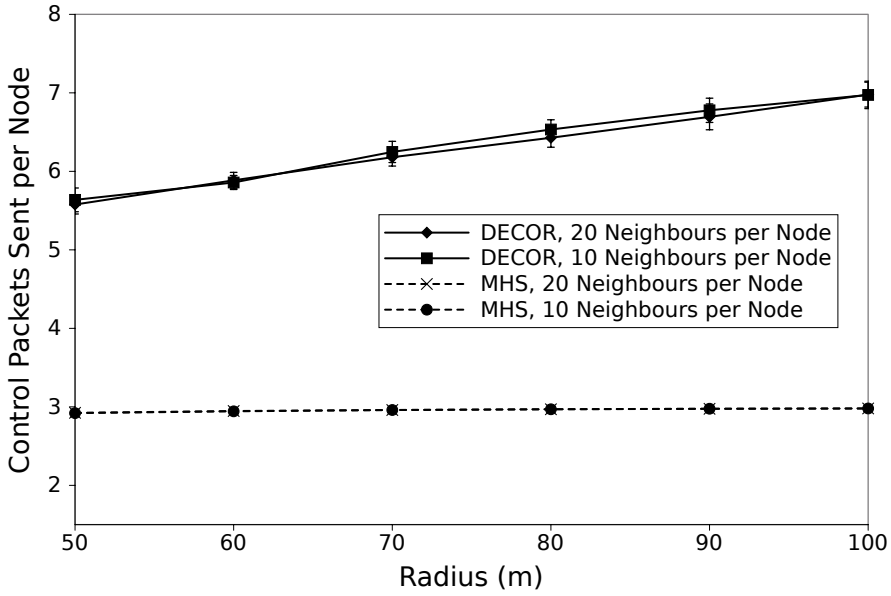

Figure 8: DECOR requires more packets to be transmitted per node but the extra cost is limited and is insignificant when compared to the total number of data packets over the lifetime of the network.

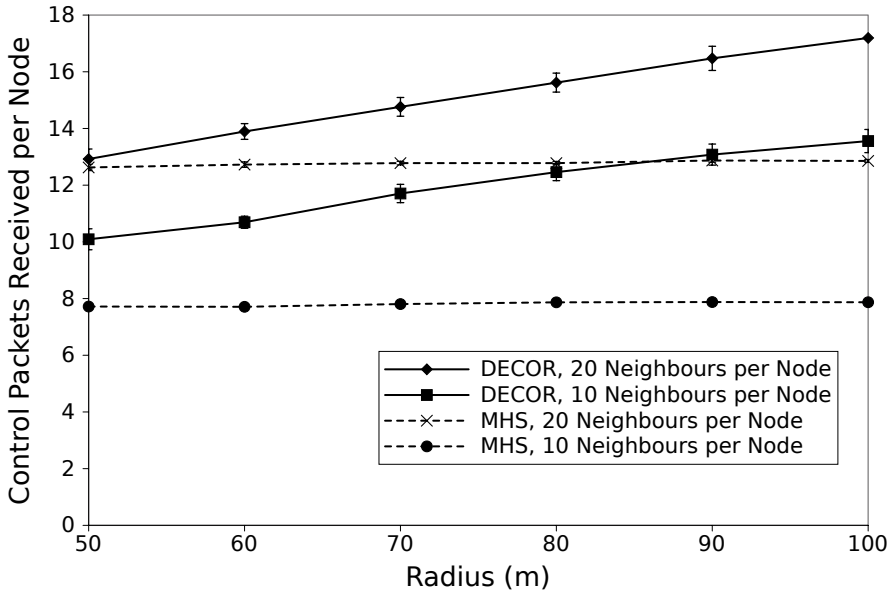

Figure 9: Because DECOR includes multiple adoption confirmations in a single packet it can sometimes require fewer control packets to be received per node than MHS. Even when this is not the case the extra cost is negligible when considering the expected lifetime of the network.

When it comes to control packets received per node, Fig. 9, the picture becomes more complex. Under MHS the number is constant with radius but increases with density. This is because each adoption must be confirmed with a broadcast to all potential children. In the case of DECOR many adoptions are confirmed together which means that the number of packets received per node grows less quickly than under MHS. Over the range of densities we simulated DECOR always requires that more packets are received but at slightly higher densities DECOR would start to require fewer control packets. In any case, compared to the lifetime of the network the extra initial communication cost is negligible.

7.3. Experiments with Realistic Radio Model

We have shown that with the overly simplistic unit disk radio model, DECOR achieves significant balance and lifetime improvements compared to MHS. We now show that these improvements remain when using a more realistic model. For this we adopt the model derived by Zuniga and Krishnamachari which utilises the log-normal shadowing model [23]. The model is given in equation (4) where $PRR(d)$ is the packet reception rate at a distance, d, $\gamma(d)$ is the signal-to-noise ratio at that distance, calculated from the log-normal shadowing model and b is the number of bits in the packet.

$$PRR(d) = \left(1 - \frac{1}{2}\exp^{-\frac{\gamma(d)}{2}\frac{1}{0.64}}\right)^{b} \tag{4}$$

Using this realistic radio model means that the number of neighbours per node is far less uniform than with the unit disk model. With this model it is necessary to define which nodes are considered neighbours because the link quality between different nodes varies. Since research has shown that for energy efficiency only high quality links should be used so as to avoid retransmissions (see for example [14]) we define neighbours as those nodes which have a packet reception rate of 1. This means that there are no retransmissions. However, since the link quality is no longer a function simply of distance and can vary randomly, the number of neighbours per node can vary significantly. This should impair the performance of both MHS and DECOR as it creates inherently more imbalance in the network.

For these experiments we found that the effective density was higher than with the unit disk model and therefore we used lower densities. Unfortunately because of the model it is not possible to succinctly define the density in terms of the number of neighbours per node. Therefore we utilised two densities that we refer to as *low* and *high*. The high density has 0.03 nodes per square metre and the low density has 0.02 nodes per square metre.

The results show that with the more realistic model, and the higher density, both MHS and DECOR have improved performance. This is because the average link length is increased and so the network has fewer coronas and therefore imbalance at higher levels has less impact. Nevertheless, DECOR continues to show significantly improved performance over MHS despite the network being significantly less uniform.

Fig. 10 shows the results for balance which clearly show that DECOR can achieve near perfect balance whereas MHS can only achieve such high balance with very small networks. The results for the mean/max ratio, in Fig. 11, confirm that DECOR now achieves lifetimes that are much closer to the optimal than MHS is capable of achieving. Moreover, as the network becomes larger the performance of DECOR increases while MHS degrades.

The costs of the balance and lifetime improvements remain, of course, in both increased latency and added communication overhead. The results follow a similar pattern to those with the unit disk model.

8. Discussion

The results in the previous section show that DECOR is able to create routing trees with significantly higher balance than MHS which is the next best performing algorithm. The new algorithm controls the number of children each node may adopt assuming that the nodes follow an approximately uniform distribution. DECOR can be adapted to other types of distributions, such as Gaussian, so long as a

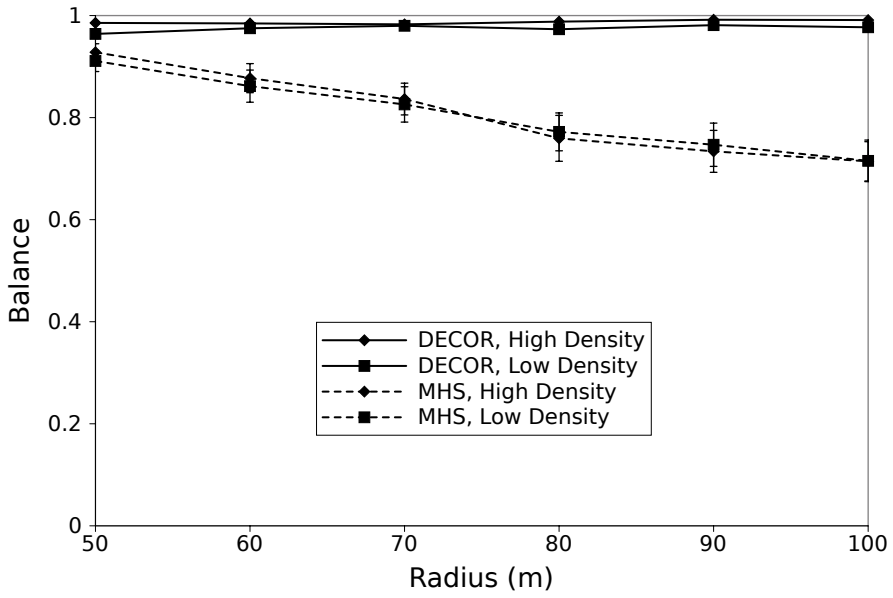

Figure 10: Even though the network is less uniform with the Zuniga model, DECOR continues to significantly outperform MHS, achieving near perfect balance.

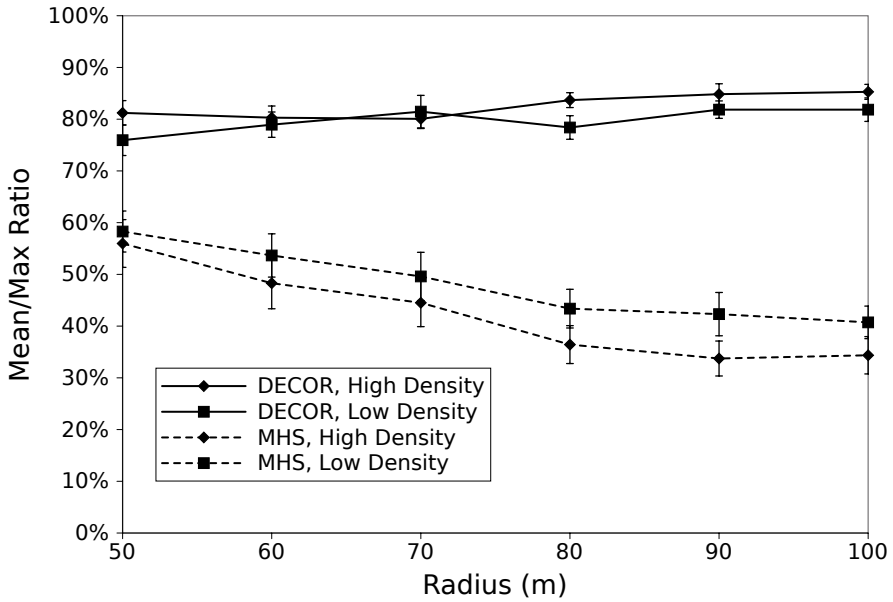

Figure 11: DECOR can achieve near optimal lifetimes even though the network is topologically far less uniform.

reasonable approximation can be made for the number of nodes in each corona. From this, adoption limits can be generated. Experiments with a netowrk with a Gaussian distribution confirm this (see [24] for more details).

It is important to note that while DECOR produces more balanced trees than MHS, in many cases the resultant tree is still significantly suboptimal. On the other hand centralised algorithms can produce trees with close to optimal balance but the cost of initially gathering information is very high because there is no routing tree in place. We suggest, therefore, that our distributed algorithm could be used to create an initial tree that is quite well balanced which can then be used to more cheaply gather the information needed for a centralised algorithm. Moreover, a centralised algorithm that uses a more balanced tree as a starting point is likely to converge faster and produce a more balanced final tree than one starting with a highly imbalanced tree.

9. Conclusion

The energy hole problem has been identified as a major problem for sensor networks, particularly monitoring networks where the nodes are likely to be static and uniformly distributed. A number of approaches have been proposed and analysed for solving the problem completely but each imposes some constraints on the network. In many cases the problem is inherent to the network and cannot be avoided which means that the best that can be aimed for is to mitigate the problem by maximising the load balancing among the most critical nodes which are those in the inner-most corona.

In this paper we propose a new distributed algorithm for the construction of load balanced routing trees. Our solution works by placing limits on the number of children nodes are allowed to adopt during tree construction. DECOR attempts to ensure that all nodes in a given level adopt the same number of children. This has the effect of growing the subtrees by the same number of nodes in each level so that balance is maintained.

Our results show that the new algorithm can produce routing trees with higher balance and longer lifetime than MHS, the next best performing alternative. The price paid is slightly increased latency which never amounts to more than a 10% increase which is a small price to pay for the much larger gains to lifetime. There is also extra overhead in terms of packets transmitted and received but these are negligible compared to the number that will be transmitted and received throughout the network lifetime.

Although DECOR was developed for a uniformly distributed network, our results show that even when the network is far from regular DECOR continues to produce highly balanced trees. Moreover, the DECOR algorithm can be easily adapted to any regular distribution type, such as a Gaussian network, and can tolerate irregularities as well.

It is important to note that distributed algorithms are not the only option. They are effective at producing an initially balanced tree but one of the centralised rebalancing algorithms can be used afterwards to increase the balance even further. Initial tests indicate that using a centralised algorithm to rebalance a routing tree generated by DECOR converges faster and results in better balance than rebalancing a shortest-path routing tree. Cooperation between distributed and centralised algorithms in order to leverage the advantages of both is a rich area for future research in sensor networks.

In future work, we would also like to investigate whether the second phase of DECOR can be used to actually improve balance by having previously disconnected nodes prioritise lighter loaded subtrees. This would of course require more communication costs but typically one off communication costs become negligible when compared to the much longer lifetime of the network. This would lead us to research the problem of distributed rebalancing. While some existing distributed algorithms are designed to rebalance routing trees, their overheads are high and their effectiveness is limited. If an efficient and effective distributed rebalancing algorithm could be developed it may prove to be cheaper than relying on a centralised rebalancing algorithm.

[1] J. Li and P. Mohapatra. An analytical model for the energy hole problem in many-to-one sensor networks. In *IEEE Vehicular Technology Conference*, volume 62, page 2721. IEEE; 1999, 2005.

[2] J. Li and P. Mohapatra. Analytical modeling and mitigation techniques for the energy hole problem in sensor networks. *Pervasive and Mobile Computing*, 3(3):233–254, 2007.

[3] Ivan Stojmenovic and Stephan Olariu. *Data-Centric Protocols for Wireless Sensor Networks*, pages 417–456. John Wiley & Sons, Inc., 2005.

[4] S. Olariu and I. Stojmenovic. Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006.

[5] Chao Song, Ming Liu, Jiannong Cao, Yuan Zheng, Haigang Gong, and Guihai Chen. Maximizing network lifetime based on transmission range adjustment in wireless sensor networks. *Comput. Commun.*, 32(11):1316–1325, July 2009.

[6] A. Kleerekoper and N. Filer. The relay area problem in wireless sensor networks. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–5. IEEE, 2012.

[7] H. Zhang and H. Shen. Balancing energy consumption to maximize network lifetime in data-gathering sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 20(10):1526–1539, 2009.

[8] Jamal N Al-Karaki and Ahmed E Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28, 2004.

[9] R.C. Shah and J.M. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 1, pages 350–355. Ieee, 2002.

[10] D. Puccinelli and M. Haenggi. Arbutus: Network-layer load balancing for wireless sensor networks. In *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pages 2063–2068. IEEE, 2008.

[11] Sameer Tilak, Nael B Abu-Ghazaleh, and Wendi Heinzelman. A taxonomy of wireless micro-sensor network models. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(2):28–36, 2002.

[12] F. Bouabdallah, N. Bouabdallah, and R. Boutaba. On balancing energy consumption in wireless sensor networks. *Vehicular Technology, IEEE Transactions on*, 58(6):2909–2924, 2009.

[13] Alberto Cerpa, Jennifer L Wong, Miodrag Potkonjak, and Deborah Estrin. Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 414–425. ACM, 2005.

[14] A. Kleerekoper and N. Filer. Revisiting blacklisting and justifying the unit disk graph model for energy-efficient position-based routing in wireless sensor networks. *Wireless Days*, 2012.

[15] P.H. Hsiao, A. Hwang, HT Kung, and D. Vlah. Load-balancing routing for wireless access networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 986–995. IEEE, 2001.

[16] R. Jain, D.M. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *Technical Report, Digital Equipment Corporation*, DEC-TR-301, 1984.

[17] T.S. Chen, H.W. Tsai, and C.P. Chu. Adjustable convergecast tree protocol for wireless sensor networks. *Computer Communications*, 33(5):559–570, 2010.

[18] P. Andreou, A. Pamboris, D. Zeinalipour-Yazti, P.K. Chrysanthis, and G. Samaras. Etc: Energy-driven tree construction in wireless sensor networks. In *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*, pages 513–518. IEEE, 2009.

[19] C. Huang, R.H. Cheng, T.K. Wu, and S.R. Chen. Localized routing protocols based on minimum balanced tree in wireless sensor networks. In *Mobile Ad-hoc and Sensor Networks, 2009. MSN'09. 5th International Conference on*, pages 503–510. IEEE, 2009.

[20] G. Chatzimilioudis, D. Zeinalipour-Yazti, and D. Gunopulos. Minimum-hot-spot query trees for wireless sensor networks. In *Proceedings of the Ninth ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 33–40. ACM, 2010.

[21] A. Kleerekoper and N. Filer. Trading latency for load balancing in many-to-one wireless networks. In *Wireless Telecommunications Symposium (WTS), 2012*, pages 1–9. IEEE, 2012.

[22] M. Macedo. Are there so many sons per node in a wireless sensor network data aggregation tree? *Communications Letters, IEEE*, 13(4):245–247, 2009.

[23] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 517–526. IEEE, 2004.

[24] Anthony Kleerekoper. *Distributed load balancing in many-to-one wireless sensor networks.* PhD thesis, the University of Manchester, 2013.