Towards an Energy Balancing Solution for Wireless Sensor Network with Mobile Sink Node*

Craig Thomson[a,**], Isam Wadhaj[a,*], Zhiyuan Tan[a,*] and Ahmed Al-Dubai[a,*]

[a]Edinburgh Napier University, School of Computing, 10 Colinton Road, Edinburgh, UK, EH10 5DT

ARTICLE INFO

Keywords:
Wireless Sensor Networks
Sink Mobility
Energy Holes
Balanced Energy
Duty Cycle

ABSTRACT

The issue of energy holes, or *hotspots*, in wireless sensor networks is well referenced. As is the proposed mobilisation of the sink node in order to combat this. However, as the mobile sink node may communicate with some nodes more than others, issues remain, such as energy spikes. In this study we propose a lightweight MAC layer solution - Dynamic Mobility and Energy Aware Algorithm (DMEAAL). Building on existing solutions utilising a communication threshold between static nodes and a sink node using a predictable mobility pattern, DMEAAL takes knowledge of optimum energy consumption levels and implements a cross-layer approach, utilising current energy consumption and dynamically adjusting communication threshold size based on target energy consumption. This approach is shown to balance energy consumption across individual nodes without increasing overall energy consumption compared to previous solutions. This without detrimentally affecting frame delivery to the sink. As such, network lifetime is improved. In addition we propose Mobile Edge Computing (MEC) applications for this solution, removing certain functionality from static nodes and instead deploying this within the mobile sink at the network edge.

1. Introduction

In the use of wireless sensor networks (WSN), one of the greatest challenges lies in the area of energy consumption. WSNs may also be referred to as low power and lossy networks, due to the nature of the devices used. Tiny devices of low memory and processing capacity as well as limited battery power. Sensors and sensor networks are now found in many aspects of modern life as the move toward smart cities gains pace [1]. However, given there is also the potential for many inhospitable locations in which these networks may reside, with uses such as in deep sea oil and gas [2], disaster recovery [3] and agriculture [4], another issue arises in the ease of battery replacement. As such, these devices must adopt duty cycling methods in order to conserve power and increase network lifetime. Duty cycling involves nodes sleeping when idle and can thus have great benefit in reducing energy conservation [5]. Duty cycling is controlled at the MAC layer, with different approaches utilised by the protocols implemented at this layer. In carrier-sense multiple access (CSMA) MAC implementations such as the IEEE 802.15.4 standard [6, 7], clear channel assessment (CCA) is utilised to determine if the channel is clear or not before sending data. This is used alongside the sending of preambles in order to keep the channel open to receive data. A transmitted preamble required to be at least the length of the receiver's sleep period, with this shown to be of benefit regarding energy consumption [8]. However, as a node may now be asleep for a considerable amount of time, issues arise in terms of how the network may continue to function properly. An inherent challenge is in ensuring that Neighbour Discovery (ND) may still happen with duty cycling in place, explicitly, that an overlap of wake-up schedules may take place between nodes such that ND may actually take place. Otherwise, communication between nodes would become impossible and the network would be rendered useless.

Another issue arises in static implementations of WSNs, due to the multi-hop to sink basis of these networks. In this event the WSNs are prone to the issue of nodes closest to the sink taking on greater load and therefore running out of energy faster. This would leave the other nodes in the network unable to communicate with the sink node, effectively ending the lifetime of the entire WSN as a functioning entity. This is known as the *energy hole* [9] or *hotspot* problem. In combating the hotspot problem, a common approach is to mobilise the sink node. The assumption in taking this approach is that as the sink node moves, as does the responsibility for final delivery to it. Thus, the load would be spread across several nodes in the network, rather than just a select few. These are termed as *significant nodes*[10, 11]. Mobile sink nodes (MSNs) are already utilised in different applications such as to be placed in a vehicle, robot or carried by a person. An emerging area is that of the use of unmanned aerial vehicles (UAVs), more commonly referred to as drones [12], with applications for use in areas such as dealing with forest fires [13] . However, in merely mobilising the sink node there can be no guarantee that all nodes the sink passes shall expend energy at the same rate, with any energy *spikes* being problematic in the longer term, as even a small difference could still result in certain nodes running

*This document is the result of a PhD study funded by Edinburgh Napier University.

*Corresponding author
**Principal corresponding author
C.Thomson3@napier.ac.uk (C. Thomson); I.Wadhaj@napier.ac.uk (I. Wadhaj); Z.Tan@napier.ac.uk (Z. Tan); A.Al-Dubai@napier.ac.uk (A. Al-Dubai)
napier.ac.uk/people/craig-thomson (C. Thomson); napier.ac.uk/people/isam-wadhaj (I. Wadhaj); napier.ac.uk/people/thomas-tan (Z. Tan); napier.ac.uk/people/ahmed-aldubai (A. Al-Dubai)
ORCID(s): 0000-0001-7320-7262 (C. Thomson)

out of energy earlier than others as time passes.

Traditional approaches to ND in WSNs involve the use of probabilistic [14] and deterministic [15] algorithms to either guarantee an overlap in the case of deterministic approaches, or result in a high probability in the case of probabilistic approaches. Although both approaches are beneficial, deterministic approaches are found to be the most commonly used. The long-tail discovery issue is uniquely problematic to the probabilistic approach, and may result in a node not being discovered at all [16]. Otherwise, it has been demonstrated to be more efficient than its deterministic counterpart in ensuring ND. New studies, however, have emerged in the area of ND in WSNs, such as opportunistic routing approaches, which result in decisions being made *on the fly* [17]. However, it is in the area of mobility awareness in WSNs [18], with knowledge of mobility utilised to influence network behaviour, in this case ND, that this study takes place. Given the complex nature of ND in WSNs, the introduction of a mobile element adds another layer of complexity to the issue. Such as, in the case of the use of a MSN, how may nodes know to awaken when the sink node is nearby. However, whereas other studies may seek to negate the effect of mobility, in the use of mobility awareness, the prediction of mobility patterns of mobile nodes is utilised to improve elements such as routing and data delivery [19]. Our approach instead aims to use the mobility pattern of a MSN as an effective metric by which to positively influence the energy consumption of static nodes in the network and, subsequently, attempt to balance energy consumption across all the nodes the sink directly communicates with.

To achieve this, we build on our previous work to propose a novel MAC layer algorithm to be utilised in WSNs with a MSN. In our original study, the Mobility Aware Duty Cycling Algorithm (MADCAL) [10, 11] created a dynamic communication threshold between a MSN and static sink node when a predictable sink mobility pattern is in use. This threshold is then used to influence the SLEEP function of a MAC implementation within a static, significant node. As such, communication between a static node and the MSN is only possible when the sink is within this threshold, otherwise waiting until the threshold is reached. This work was then extended in Mobility Aware Duty Cycling and Dynamic Preambling Algorithm (MADCaDPAL) [20], in which the relationship between the MSN and static sink threshold was examined in finer detail, further improving energy consumption. However, an issue of *energy spikes* amongst significant nodes was found, where despite average energy across significant nodes improving, some nodes still take on a greater network load than others. This issue results from lack of knowledge of neighbouring nodes as the communication threshold is implemented in each node independently without using expensive beacons between nodes. As such, network density may cause overlaps of communication, while others wait for a clear channel. Whilst this issue has been negated to a degree, it is not eliminated completely. As such, the elimination of these energy spikes, and the subsequent improvement in network lifetime, is the motivation for this study.

In this paper we propose the Dynamic Mobility and Energy Aware Algorithm (DMEAAL), which implements a completely dynamic approach to energy conservation, by adjusting the aforementioned communication threshold in *real time*. In utilising a target, optimum, level of energy consumption, we propose a cross-layer approach in consideration of a threshold adjusted accordingly as time passes, based on current node energy consumption. In this way we seek to eliminate *dead energy consumption* in significant nodes, where the energy consumption of one node is unnecessarily higher than others, when the load could be spread evenly across all significant nodes. Such an approach, which results in the consumption of energy more evenly across individual nodes, has merit when considering the potential long life of nodes in WSNs. With the importance of the consideration of energy balancing having been demonstrated in other work [21].

In summary, this work ensures that nodes shall not die out more quickly than necessary. Even one node dying in a network will have a detrimental effect, with others then having to assume extra load and, in some cases, this not proving possible due to node location. As such, communication with the sink may not be possible for some nodes. Therefore, ensuring that significant nodes all consume energy equally extends network lifetime and ensures a consistent communication with the sink node.

The application of this work is seen to come under the Mobile Edge Computing (MEC) banner. An important difference from a more typical view of MEC in mobile WSNs, where energy consuming and processor-heavy functionality is passed to edge servers [22, 23], is that in this case the functionality is conducted by the actual mobile device itself. In this case the MSN. Mobile sinks have previously been proposed as an important part of the functionality of MEC for data gathering alone [24]. However, in the case of this study it is envisaged that further functionality be deployed to the MSN at the network edge. In this way, this approach may contribute to an important aspect of the MEC, that being to utilise edge computing in mobile environments to conserve energy [25, 26, 27]. As such, we envisage two possibilities:

1. Individual nodes send current battery levels to the sink node via a message. The sink node may then calculate the average energy consumption across all significant nodes and send this figure back to each node. The nodes then implement the DMEAAL algorithm, using the figure received from the sink as the target energy consumption. This solution would be completely integrated in network operation. However, additional messaging is required, adding to network overhead and potential delay. The target energy consumption would also only be based on one network run. Whereas the target energy consumption utilised in the tests detailed in this study are derived from five tests combined. This would raise the possibility of an outlying result influencing the calculation.
2. A second application would see target energy consumption pre-programmed in the sink node. This based on multiple tests, removing the risk of outlying re-

sults. This figure may then be sent to each node as target energy consumption, with the DMEAAL algorithm implemented. This solution requires that only the sink sends a message to each node, reducing extra load on the network. Whilst not completely integrated, this solution removes the need for each node to be pre-programmed with target energy consumption. However, there remains the possibility of data loss and delay, with significant nodes unable to implement the DMEAAL algorithm until the message is received.

Ultimately it is envisaged that responsibility for all common network functionality be deployed at the network edge within the MSN. Such as sink start position, start time and speed. With other factors dependent on the mobility pattern in use. As shall be detailed, within this study circular mobility is in use therefore factors such as the centre of the circle would be of importance. It would then be the responsibility of the MSN to send this information to the other nodes in the network.

The rest of the paper is organised as follows. Section 2 examines related work in the area of the proposed algorithm, while Section 3 gives a technical background of Mobility Aware Duty Cycling and the original MADCAL algorithm. Section 4 details the sink mobility pattern and network topologies used to test the new algorithm. Section 5 details the DMEAAL algorithm, the methodology, resultant algorithm, testing and results. Section 6 concludes this paper and proposes future work.

2. Related Work

When examining existing work in the area of MSNs a great many focus on network layer routing protocols, with little interest shown in the MAC layer. When it comes to the mobility of the sink itself, the desire to determine and optimal path has also inspired a considerable number of studies, as evidenced by Aggarwal and Kumar [28]. With some works covering both areas [29]. The use of a MSN to influence the MAC layer duty cycling of static nodes is a pioneer approach and of high novelty. Consequently, related work reflects the journey towards this solution.

2.1. Energy Efficient Routing to MSN

A study by Wang et al. [30] recognises the further challenges that arise in the use of MSNs, despite their benefit as a potential solution to the *hotspot* problem. An "energy-efficient cluster-based dynamic routes adjustment approach (EECDRA)" [30] is proposed. This clustering approach bases the selection of cluster heads on residual energy levels in nodes. As such, maintaining the best and most energy efficient routes to the MSN. This approach shows benefit in eliminating the use of expensive messaging in order that knowledge of the sink position is maintained in the network. As a network layer solution, network lifetime is shown to improve with this approach.

Yarinezhad et al. [31] recognises the need for MSNs in order to negate the hotspot issue, and that if this is used correctly, energy consumption may be balanced out. The proposal here is for a routing algorithm in which the network is divided into cells. This so that only a certain number of cells require to store the location of the sink node as it moves through the network. This works with any sink trajectory as the sink location is sent to other nodes in the network via beacon messages. As such, network lifetime and energy consumption is shown to improve versus other approaches as each node may more easily find the shortest route to the sink node. This is an important approach to routing with a MSN in order to attain the best results from its use and would have potential for future use with other studies, potentially utilising different approaches at other layers. This, however, is again a network layer approach to utilising a MSN, and also relies on the exchange of messages, which may add to network overhead.

Another study rooted in the network layer [32], however, does recognise the importance of including duty cycling in considerations for the use of a MSN. This study synchronises the LEACH routing protocol such that the duty cycle of high-level nodes near the cluster-head is based on the wake-up slot of previous nodes. Such that wake-up schedules are synchronised accordingly. This study claims improvement in energy consumption over the standard LEACH routing protocol. Although ultimately, the link with sink mobility and duty cycling is unclear in this study, it is of interest. It could be argued that any duty cycling implementation, located at the MAC layer, would ultimately need to work in conjunction with the network layer.

In reference to the additional network cost of constantly updating nodes in a network of the MSN position, [33] proposed a new algorithm to more effectively advertise mobile sink position and thus conserve energy. This is achieved by utilising nested rings such that when the sink position is needed in order for a node to send data, it first requests the sink location from the closest node within the closest ring. A geographic routing algorithm is then used to send the data to the mobile sink. This study supports the use of multiple sinks and claims benefits in network lifetime and packet delay. This study recognises a the benefit in terms of energy consumption in reducing messaging in a WSN in order that nodes may know the sink position. Although this is only relevant when using unpredictable sink mobility. Again, the theme of MSN solutions being located in the network layer continues with this study, with efficient packet delivery the ultimate aim.

2.2. Predictable or Proactive Sink Mobility

Predictable sink mobility, or in this case constrained mobility, is highlighted in [34]. There is also a reference to the nodes most likely to communicate in one-hop with the sink as *sub sinks* [34]. The approach proposed here focuses on the data traffic itself, and how it can be expected to behave as a result of the sink mobility. The desire is to balance energy consumption across network hierarchy by establishing a grid. This would appear to be a common approach to the issue of sink mobility and how to subsequently communi-

cate with the sink. This approach shows benefit in terms of network lifetime, and the recognition of the nodes closest to the sink path is of interest also as is their role in delivering data to the MSN. Sink mobility, however, is seen as an issue to be solved in this case. Despite predictable sink mobility being used in [34], it takes a different approach to the same problem in contrast with our own studies [10, 11, 20].

Mitra et al. proposed another study utilising a grid structure for delivery to a MSN [35]. This work used a proactive sink mobility pattern, changing to account for the state of the network. This work also revisited the use of sub sinks and extended this to use candidate sub sinks. As such, not all candidate sub sinks would become sub sinks but they are proactively identified as having that capability. The actual sub sinks are then selected as those passed by the sink in a set period of time. This work showed benefit in terms of a reduction in energy consumption, hop count and delay. The approach of identifying sub sinks from a group of candidates is of interest, as is the proactive mobility. Sink mobility, however, remains an issue seen to be negated.

An example of the use of a MSN with a predefined path can be found in [36]. This study aimed to collect the maximum possible amount of data while the mobile sink is traversing the shortest path throughout the network. As such, this proposed the selection of sub-sinks which would communicate directly with the sink. Each of the static nodes in the network must select one sub-sink to communicate with. This approach claimed benefits in terms of data collection and energy consumption and is an example of how a predefined sink mobility path may be a metric which can be utilised. However, this approach relies completely upon message exchange between the sink node and static nodes and between the static nodes themselves. This is in order to establish the sub-sinks and, subsequently, the shortest path to these nodes. It can be assumed that this would place additional load on the network that would be desirable to reduce or eliminate.

3. Mobility Aware Duty Cycling

In creating the MADCAL algorithm [10, 11], a predictable sink mobility pattern approach was utilised. Then, given the network parameters of sink starting position, speed and the time it has been travelling for, it is possible for each static node to accurately calculate the current sink position. This calculation made independently in each node, without the use of energy expensive beacon messages. The mobility pattern utilised by MADCAL is a circular path around the network. This such that an approach is taken where the sink directly reaching each node was not possible, more accurately recreating a real-life scenario such as in disaster recovery. Therefore, the nodes to be identified, the *significant* nodes, are those which are one-hop from the path of the sink based on interference range. MADCAL features two algorithms as follows.

3.1. MADCAL-1

The first to create a dynamic threshold between static node and mobile sink. In taking the point closest to the circular sink path in relation to the significant node, deemed the *circlePoint*, an angle may then be calculated based in interference range. This would be the maximum communication threshold before and after the circlePoint. However, to avoid extremes in the size of thresholds, with the size being considerably large of the node is close to the path and vice versa, if some distance away, a factor is applied in order to regulate this. Node distance to the sink path is divided by interference distance and then sink speed is taken into account to adjust the factor further. With smaller thresholds more effective with faster sink speeds and a slower sink requiring larger thresholds, due to less opportunities to communicate with each node, the following calculation is made. For speeds less than 10 mps the threshold cannot be reduced by less than a factor of 0.5, for less than 20 mps this factor reduces to 0.35, reducing again to no less than 0.25 for less than 40 mps. If the speed is exactly 40 mps we take no action other than the factor remaining at the initial calculation of node distance over interference distance. It should be noted that this *factor check* is based on test speeds of 40, 20, 10 and 2mps (meters per second). As such, in this new study we shall be implementing a more dynamic approach to cater for any speed within this range.

3.2. MADCAL-2

The second MADCAL algorithm focuses on how to utilise this threshold in order to then effectively influence duty cycling within a static node. By calculating the current MSN position within the SLEEP procedure of the MAC layer implementation in use, the node can calculate if the sink is within its threshold or not. If it is not then the node may delay the move from SLEEP to CCA, in effect remaining asleep whilst still able to receive messages. This is achieved by calculating the time it will take for the sink node to reach the start of the threshold based on the current sink coordinates, coordinates of the threshold start, current time and sink speed. In comparing this to a standard duty cycling approach with CCA, preambles and check interval, an improvement of up to 15% in energy consumption was shown. This as well as keeping frame delivery at similar levels or better.

The MADCaDPAL algorithm, an extension of MADCAL-2, further seeks to ensure the threshold is *closed* once the MSN has reached its end. As such, in utilising the sink position in relation to CCA and the sending of preambles, ensuring these processes end once the end of the threshold is reached, significant benefits are achieved. Energy consumption is now improved by up to 80% with frame delivery also increased to the sink. This approach was also shown to reduce the energy spikes which could still be found in the use of MADCAL, with certain significant nodes expending considerably more energy than others.

3.3. A Long Running Issue with Static Threshold

A communication threshold between a static *significant* node and a MSN has been shown to have benefit in terms of energy consumption. Thus far, however, once the threshold is established it remains the same, despite the level of battery

consumption within the static node [10, 11, 20]. In both the MADCAL and MADCaDPAL algorithms, the communication threshold utilised between static significant node and MSN is adjustable in that it is dependent on the sink speed, the distance between the static node and the sink path and finally the interference range of the static node. However, once this threshold is established it then is not altered again. At this stage, therefore, it is assumed that the threshold distance is optimum for however long the WSN exists for.

Whilst this approach has demonstrated benefit in terms of energy consumption, there must also be consideration given to the fact that any nodes expending energy greater than the average consumption of all significant nodes in the network shall continue to do so, until eventually running out of power earlier than other nodes. Considering how long these networks may operate for, even a small difference over a short period, could add up to a node expiring significantly sooner than necessary over the lifetime of the network. Therefore, an improvement in balancing the energy consumption of significant nodes in line with the average consumption, even if this improvement is initially small, would become more significant over the a longer period.

4. The Proposed DMEAAL Algorithm

To address this remaining issue with MADCAL and MADCaDPAL algorithms, a real-time, dynamic threshold approach to mobility aware duty cycling is proposed and termed DMEAAL, short for Dynamic Mobility and Energy Aware Algorithm.

4.1. Mobility Pattern

An important factor to consider is that this study does not seek to establish an optimal mobility pattern. The aim of both the MADCAL and MADCaDPAL algorithms is to utilise a *predictable* mobility pattern, to establish if pre-knowledge of sink mobility may be used to positively affect duty cycling and therefore, improve energy consumption. Consequently, the particular mobility model in use is not as important as that the pattern is predictable. When considering the mobility pattern, it was decided to use a simple pattern but with an implementation inspired by real-world applications. Therefore, an environment where all nodes are not treated equally, such as a disaster recovery situation where reaching each node directly is not possible. Other studies have aimed to have the MSN pass as many nodes as possible [37, 38, 39], and whilst such an approach has merit, it is clear that this is not always possible in a real-world scenario. Hence, throughout this study the approach is taken to replicate a network scenario where the sink reaching all nodes would not be possible. As such, the pattern utilised must move around the periphery of the network and give static nodes the opportunity to communicate with the sink in the least number of hops, whatever the particular network topology. This is achieved by moving the sink around the periphery of the network in a circular pattern.

In building upon the MADCAL and MADCaDPAL algorithms to develop DMEAAL, we again utilise a circular mobility pattern around the static nodes in the network.

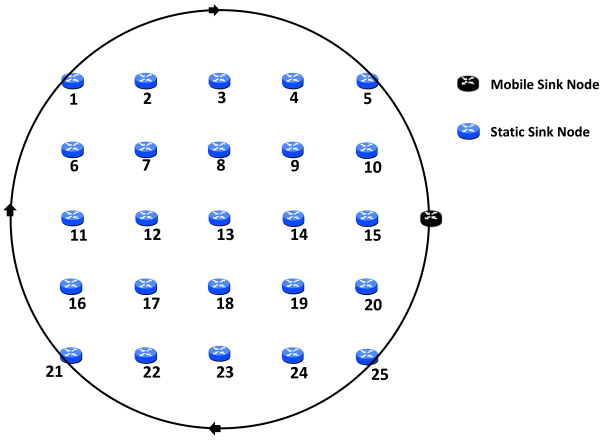


Figure 1: Network topology - grid.

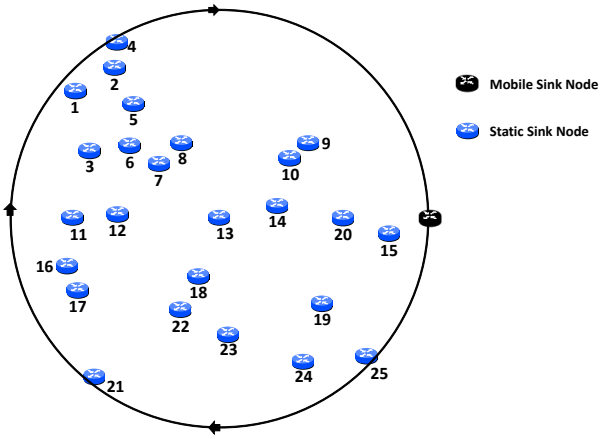


Figure 2: Network topology - random.

4.2. Network Topology

Two approaches are taken to network topology. Firstly, a controlled grid formation is utilised, with each of the 25 static nodes within one-hop of neighbours. This in order to observe the effect sink mobility has on the nodes in the network when node density and location has been controlled. This can be seen in Figure 1, with the sink node travelling in a clockwise direction around the network.

Secondly, a random network topology is used. While there are still 25 static nodes, their location, with some large spaces between nodes and some clustered together, is designed to test the ability of the algorithms to positively affect duty cycling in a situation where node location is not controlled. Especially where there may be a propensity for significant nodes to overlap and interfere with each other's transmissions. It is important to demonstrate that even in this scenario DMEAAL would still be effective. This is illustrated in Figure 2.

We utilise the grid and random network topologies with the test parameters, the interference range of nodes is varied across four values. Within the grid topology this results in

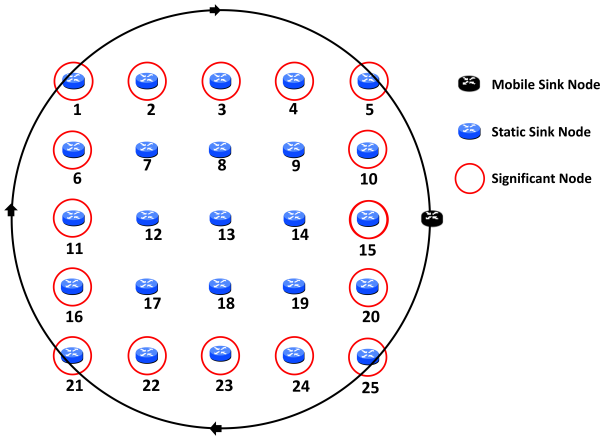


Figure 3: Network topology - grid with significant nodes.

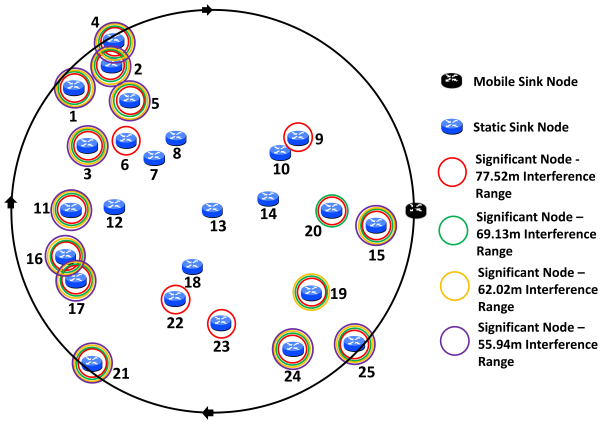


Figure 4: Network topology - random with significant nodes.

the same significant nodes each time, as can be seen in Figure 3. However, with the random topology in use the number of significant nodes reduces as the interference range of nodes becomes smaller. This can be seen in Figure 4.

Among both network implementations the assumption is that static nodes will retain their positions throughout. Each static node implements DMEAAL independently and although aware of its own location, is unaware of neighbouring nodes. Although four different interference ranges are used across different test scenarios, each range is consistent across all nodes in each test.

4.3. A Real-time Approach to Mobility Aware Duty Cycling

In this study, the interaction between the communication threshold and the internal workings of the MAC protocol is no longer the focus. Instead, we now aim to adjust the threshold for a static significant node as time passes, such that when energy is being consumed above or below the average rate, the threshold may be altered accordingly.

Hypothesis: Our previous work has shown that once a communication threshold between static node and MSN is established using the MADCAL algorithm, the energy consumption plunges by 80% when the MADCaDPAL algorithm governs the determination of whether the MSN is within the threshold or not and ending communication at the correct time. Based on this set-up, our hypothesis is that if the network is now operating at near full efficiency in terms of the performance of significant nodes, then the total energy consumed by all significant nodes is unlikely to be improved by any great degree.

Concern: However, if some nodes are consuming more energy than others, then it must be possible to lessen the reliance on these nodes and increase reliance accordingly on other nodes. Such that whilst the overall level of consumed energy remains, it is now spread more evenly across significant nodes. This without being to the detriment of frame delivery. Even given the benefit of the MADCaDPAL algorithm, at now much lower level of energy consumption then previously, it can still be observed that there is a gap in energy consumption between the highest and lowest significant nodes that could potentially be closed.

Proposition: As such, we seek to eliminate *dead* energy consumption. That being, energy which need not be consumed by one node and could be consumed by another instead with similar results but an increase in network lifetime. Here, we first utilise the MADCAL algorithm to set up the base communication threshold between a static *significant* node and a MSN. An illustration of this can be seen in Figure 6, from the viewpoint of node 10. The threshold is created based on significant node distance from the circular sink path, but an adjustment is then made based on a factor which takes into account MSN speed combined with the distance to the path. The position of the MSN in relation to this threshold is used to influence the SLEEP function at the MAC layer, intercepting the move to CCA when data is to be sent, if the MSN is not within a significant node's threshold. The sink position in relation to the threshold is also utilised to influence CCA and the sending of preambles, beyond merely awakening the node. Thus, closing the threshold of communication immediately once the MSN has reached its end. Further, however, we now seek to integrate knowledge of energy consumption whilst utilising a cross-layer approach. Thus, the current energy consumption within the node battery may be compared with the target level of consumption - in this case built from the previous test scenarios utilising the MADCaDPAL algorithm.

4.4. Dynamic Mobility and Energy Aware Algorithm (DMEAAL)

We have now developed the DMEAAL. Such that the communication threshold is altered accordingly, dependant upon the comparison between target energy consumption and actual energy consumption of the node. Target energy consumption is achieved via the MADCaDPAL algorithm, with a database of results created per scenario. This cross-layer approach is achieved by the MAC protocol accessing the battery module after each pass of the MSN, as the network scenario runs, utilising the DMEAAL algorithm. Target en-

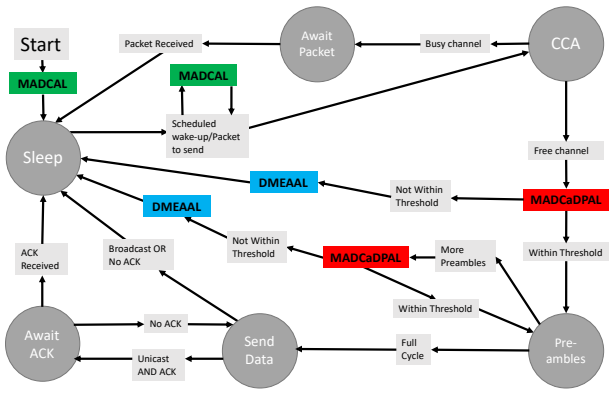


Figure 5: Lightweight CSMA MAC Implementation with DMEAAL

ergy consumption is then input at the start of each scenario. A simple adjustment value *threshAdjust* is then calculated based on the actual energy consumption's relation to target energy consumption as shown in Eq. (1).

$$threshAdjust = \left(\frac{targetEnergy}{\left(\frac{batteryCapacity - batteryResidual}{simTime} \right) \times 60} \right) \tag{1}$$

Where *targetEnergy* is the target energy consumption per minute in mWs; *batteryCapacity* is the capacity of the battery in mWs retrieved from the battery module in a cross-layer approach; *batteryResidual* is the residual energy of the battery module; *simTime* is the current simulation time in seconds.

A maximum threshold is set in the calculation of the initial threshold value, based on node interference range and distance from the path of the sink. The threshold is then adjusted based on this factor. The lightweight Carrier-Sense Multiple Access (CSMA) MAC implementation within which our work is implemented can be seen in Figure 5; showing where the DMEAAL algorithm is inserted to alter the threshold, alongside MADCAL for the initial calculation of the communication threshold and implementation in the SLEEP procedure, with MADCaDPAL for the implementation of this threshold in the CCA and SEND PREAMBLE sections.

The development of a dynamic communication threshold between static significant node from the viewpoint of node 10, has been illustrated in Figure 6. This is created by the MADCAL algorithm and then utilised by the MADCaDPAL algorithm in order to fully influence the SLEEP and CCA procedures as well as the sending of preambles. As such, communication between static significant node and MSN is closed once the sink has moved past the threshold as shown in Figure 6. However, the DMEAAL algorithm is a completely dynamic, real-time solution. Figure 7 illustrates how the threshold may now be adjusted as long as it remains within a maximum size. It also demonstrates how

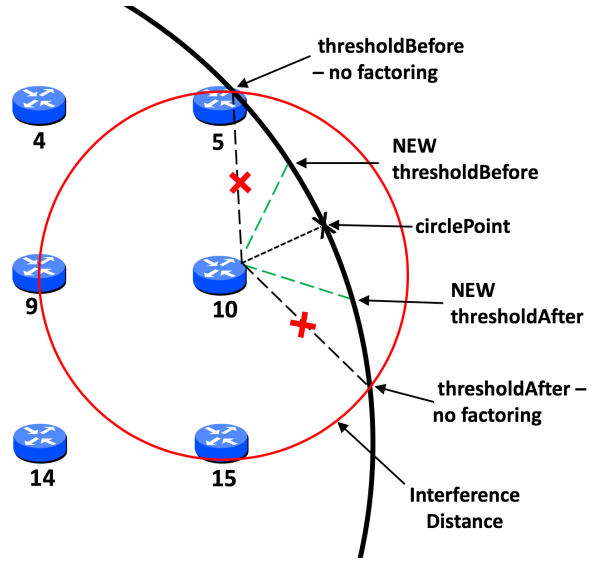


Figure 6: Illustration of Initial Threshold

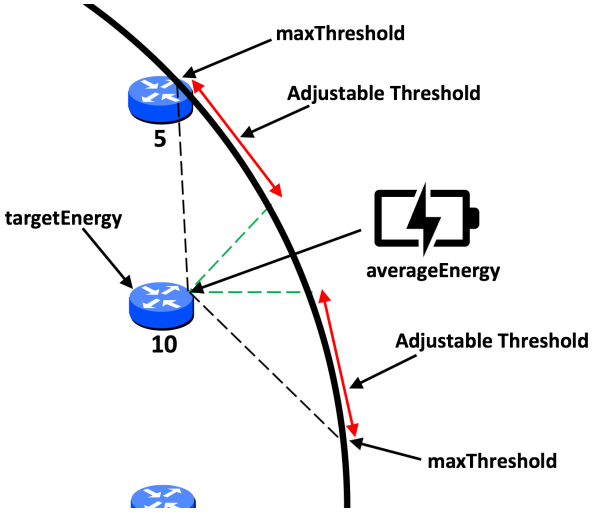


Figure 7: Adjustment of Threshold

power levels from the battery are compared to a target energy consumption level.

As such, our approach is to determine the initial threshold as currently determined by MADCAL. However, this now utilises a completely dynamic approach to establishing the factor by which the initial threshold calculation is reduced, based upon sink speed. This is detailed in Algorithm 1 followed by a detailed description. Algorithm 1 is run during the initialisation stage of all static nodes. However, whereas with MADCAL or MADCaDPAL in use this threshold would remain the same, in the case of DMEAAL this is merely an initial value. Hence, the value of this threshold will now only remain until the first recalculation based on current energy consumption.

Algorithm 1: Algorithm 1 details the initialisation and original establishment of a communication threshold during the initial startup of a static node, as initially developed in the

Algorithm 1 Initial Communication Threshold

```
1:  procedure INITIALISATION
2:      set sinkSpeed
3:      set maxSpeed
4:      set minSpeed
5:      set maxFactor
6:      set minFactor
7:      speedDiff ← sinkSpeed − minSpeed
8:      significantNode ← false
9:      set interDist            ▷ Calculate interference distance as in Eq. 2
10:     set Circumference
11:     set firstSinkPos
12:     set firstSinkQuartile
13:     set distToCircle
14:     if distToCircle < interDist then
15:         significantNode ← true
16:     end if
17:     if significantNode then
18:         set circlePoint
19:         set nodeQuartile
20:         set distanceBetweenPoints
21:         set angleOfNode
22:         thresholdAfter ← true
23:         sinkThresholdAfter ←
    establishThreshold(sinkRadius, thresholdAfter)
24:         thresholdAfter ← false
25:         sinkThresholdBefore ←
    establishThreshold(sinkRadius, thresholdAfter)
26:         set thresholdDistance
27:         set beforeQuartile
28:         set thresholdOpposite
29:     end if
30: end procedure
31: function ESTABLISHTHRESHOLD(radius, after)
32:     nodeDist ← (radius − distToCircle)
33:     angleTemp ← (radius² + nodeDist² − interDist²) / (2 ∗ radius ∗ nodeDist)
34:     angleRadians ← arccos(angleTemp)
35:     angle ← (angleRadians ∗ (180/PI))
36:     maxAngle ← angle         ▷ To be utilised later by DMEAAL. The
    largest angle possible.
37:     factor ← distToCircle / interDist
38:     factorCheck ← ((maxFactor − minFactor) / (maxSpeed − minSpeed)) ∗ speedDiff)
39:     if factor < factorCheck then
40:         factor ← factorCheck
41:     end if
42:     angle ← (angle ∗ factor)
43:     if after then
44:         threshAngleDegrees ← (angle + angleOfNode)
45:     else
46:         threshAngleDegrees ← (angleOfNode − angle)
47:     end if
48:     threshAngleRadians ← threshAngleDegrees / (180 ∗ PI)
49:     threshold.x ← circleCentre.x + (radius ∗
    cos(threshAngleRadians))
50:     threshold.y ← circleCentre.y + (radius ∗
    sin(threshAngleRadians))
51:     return Coord threshold
52: end function
```

MADCAL algorithm and then utilised further by MADCaDPAL. Within the *initialisation* procedure it is established whether this node is significant or not, with the *establishThreshold* function then establishing the size of the actual threshold. Different from the original MADCAL algorithm are the setting of *maxAngle* on Line 36 and the now completely dynamic *factorCheck* on Line 38. *maxAngle* is set to the value of the threshold before adjustment, calculated solely on the angle created by the distance from the node to the circular path and the interference distance. This shall then be utilised later in the DMEAAL algorithm to ensure the limit of the adjusted threshold. The new calculation of *factorCheck* now ensures the factor by which the threshold is reduced is a sliding scale between a maximum and minimum sink speed. In the case of this study those values are 40mps and 2mps. A maximum and minimum factor is also set as 0.5 and zero, meaning the most the factor may be reduced by is 0.5, for the slowest speeds, and not at all for the fastest speeds.

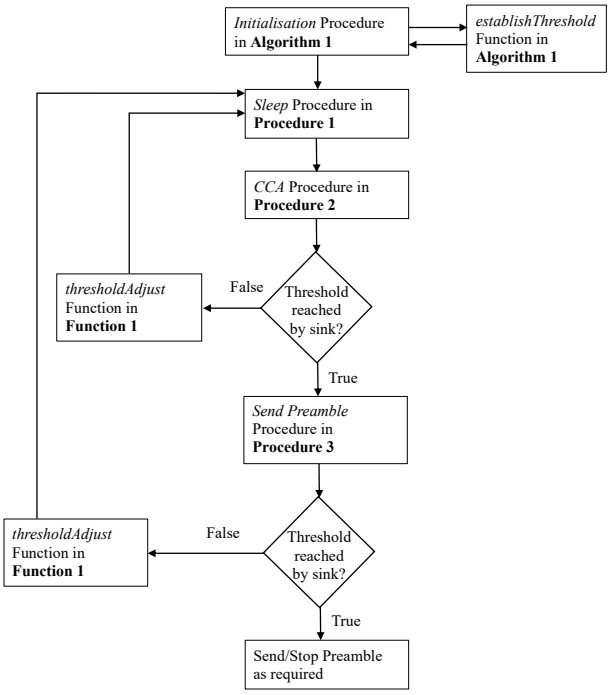


Figure 8: DMEAAL Control Flow

Figure 8 illustrates the flow of the procedures affected by DMEAAL, which can be seen in greater detail in Figure 5. This shows the MAC protocol Sleep, CCA and Send Preamble procedures as seen in Procedures 1, 2 and 3. Within each of the CCA and Send Preamble procedures the new *thresholdAdjust* function is utilised, as detailed in Function 1. The maximum threshold is set as the largest size the threshold could be when considering the node interference range and distance to the path of the sink. A cross-layer solution is then implemented where the battery module is accessed after each pass of the sink node, in order to ascertain the current energy consumed. This is calculated on a per minute basis by utilising the simulation time and then compared to the target energy consumption per minute. By dividing the target energy consumption by the current energy level we create a factor by which to apply to the current size of the threshold. In this way, if energy levels are lower than the target the threshold will be increased and vice versa. Whilst nodes have no knowledge of neighbours, as the algorithm is implemented in all nodes in the network, it will be used in the same way by each significant node. Thus, it can be assumed that as one node reduces its threshold, another shall be increasing its own.

Procedure 1: The Sleep Procedure is utilised in the MADCAL algorithm to determine the move from Sleep to CCA based on the position of the sink node in relation to the communication threshold of the static node in which this takes

Procedure 1 Sleep

```
1:  procedure SLEEP
2:      set checkInterval from input        ▷ Set the default sleep interval
3:      if significantNode then
4:          thresholdTime()                 ▷ If MSN within threshold set
        thresholdReached otherwise set timeToThreshold as time until
        MSN reaches threshold
5:          if thresholdReached then
6:              interval ← checkInterval    ▷ The interval to wake-up
        reverts to the checkInterval
7:          else
8:              interval ← timeToThreshold  ▷ Set to the time it will
        take for the sink to reach the threshold
9:          end if
10:     else
11:         interval ← checkInterval
12:     end if
13:     schedule CCA at simTime + interval  ▷ simTime is current
        simulation time in seconds
14: end procedure
```

Procedure 2 CCA

```
1:  procedure CCA
2:      if macQueue > 0 then
3:          withinThreshold()       ▷ Call withinThreshold() function to
        establish if the MSN is within the threshold currently
4:          if thresholdreached then
5:              macState ← SEND PREAMBLE    ▷ Send preambles
6:              schedule STOP PREAMBLES at simTime +
        slotDuration                ▷ Schedule the stopping of preambles
7:          else
8:              macState ← SLEEP        ▷ Return the macState to Sleep
9:              schedule SLEEP at simTime + slotDuration    ▷ Node
        awakens with SLEEP Procedure at defined time
10:             thresholdAdjust()   ▷ Call thresholdAdjust() function to
        adjust the Threshold
11:         end if
12:     else
13:         macState ← SLEEP
14:         schedule SLEEP at simTime + slotDuration
15:     end if
16: end procedure
```

Procedure 3 Send Preamble

```
1:  procedure SEND PREAMBLE
2:      withinThreshold()
3:      if thresholdreached then
4:          if SEND PREAMBLE then
5:              sendPreamble()
6:              macState ← SEND PREAMBLE
7:              schedule SEND PREAMBLE at simTime +
        (0.5 ∗ checkInterval)
8:          else if STOP PREAMBLES then
9:              cancel SEND PREAMBLE
10:             macState ← SEND DATA    ▷ Preambles over, send data
11:         end if
12:     else
13:         cancel SEND PREAMBLE/STOP PREAMBLES
14:         macState ← SLEEP
15:         schedule SLEEP at simTime + slotDuration
16:         thresholdAdjust()       ▷ Call thresholdAdjust() function to
        adjust the Threshold
17:     end if
18: end procedure
```

place.

Procedure 2: The CCA Procedure is utilised initially in the MADCaDPAL algorithm to determine whether the node should return to the Sleep procedure or continue to sending preambles, based on the position of the sink node in relation to the communication threshold. Additionally now, for DMEAAL, as well as scheduling a return to the Sleep procedure, the *thresholdAdjust* function is called to determine of the size of the threshold should be adjusted and by how much. This function is only called if the sink node is not within the threshold anymore, as adjusting it whilst communication is taking place would be problematic.

Procedure 3: The Send Preamble Procedure is again utilised initially in the MADCaDPAL algorithm to determine whether the node should return to the Sleep procedure or continue in the process of sending preambles, based on the position of the sink node in relation to the communication threshold. Additionally now, for DMEAAL, as well as scheduling a return to the Sleep procedure, the *thresholdAdjust* function is again called to determine of the size of the threshold should be adjusted and by how much. Again, this function is only called if the sink node is not within the threshold anymore.

Function 1: The *targetEnergy* is set from input as the desired average energy. In this case this is based on previous data when the MADCaDPAL algorithm is in use. The capacity of the battery is established in a cross-layer approach, directly from the battery module. This is then used in the *thresholdAdjust* function to calculate the current consumed energy by deleting the residual battery capacity from the actual battery capacity.

The purpose of the *thresholdHappened* variable is to establish if the threshold has been adjusted since the last time the sink reached it. There is no gain to be made from adjusting the threshold more than once in a circuit of the network as this could lead to this process recurring many times. The average energy per minute is calculated as is the threshold adjustment factor by simply dividing the target energy consumption by the current average energy consumption. We also set the highest and lowest values for the size of the angle of the threshold. In this case we use the *maxAngle* value as the highest and zero as the lowest. However, this may be adjusted according to a particular scenario. The *maxAngle* is set as the largest angle possible when calculating the initial threshold size at startup, based on interference range and distance to sink path. The angle is then adjusted for before and after the threshold by multiplying it by the adjustment factor within the *adjustThresholdC* function. We then establish if the new angle is within the parameters of the highest and lowest angles and adjust accordingly if not. The angle is then calculated as coordinates and returned to be saved as the new threshold.

5. Evaluation and Results

In order to test this approach, we utilise both the grid and random network topologies, with a predictable circular sink mobility pattern moving around the network in a clockwise direction as seen in Figures 1 and 2, with significant nodes illustrated in Figures 3 and 4. The network is built using the OMNeT++ [40] simulation framework, utilising both MiXiM [41] and inetmanet [42].

5.1. Network Layer

It should be noted that this work is located at the MAC layer. Therefore, a routing protocol is only used to ensure a route, and subsequent packet delivery, to the MSN. In this

Function 1 Adjust Threshold

```
1: function THRESHOLDADJUST
2:     set targetEnergy from input
       ▷ Cross-layer approach to utilise values within the battery module
3:     set batteryCapacity to nominalCapacity in batteryModule
4:     set batteryResidual to residualCapacity in batteryModule
5:     consumed ← (batteryCapacity − batteryResidual)    ▷ Current value of consumed energy
6:     if thresholdHappened then
7:         thresholdHappened ← False
8:         averageEnergy ← (consumed/simtime) × 60
9:         threshAdjust ← targetEnergy/averageEnergy
10:        angleLimitHigh ← maxAngle    ▷ As set when initial threshold established.
11:        angleLimitLow ← 0
12:        thresholdAfter ← true
13:        sinkThresholdAfter ← adjustThresholdC(sinkRadius, thresholdAfter, threshAdjust)
14:        thresholdAfter ← false
15:        sinkThresholdBefore ← adjustThresholdC(sinkRadius, thresholdAfter, threshAdjust)
16:        set thresholdDistance
17:        set beforeQuartile
18:        set thresholdOpposite
19:    end if
20: end function
21: function ADJUSTTHRESHOLDC(radius, after, adjustment)
22:    if after then
23:        angleAfter1 ← angleAfter × adjustment
24:    else
25:        angleBefore1 ← angleBefore × adjustment
26:    end if
27:    if after then
28:        if ((angleAfter1 < angleLimitHigh) and (angleAfter1 > angleLimitLow)) then
29:            angleAfter ← angleAfter1
30:        else
31:            if angleAfter1 >= angleLimitHigh then
32:                angleAfter ← angleLimitHigh
33:            else
34:                angleAfter ← angleLimitLow
35:            end if
36:        end if
37:    else
38:        if ((angleBefore1 < angleLimitHigh) and (angleBefore1 > angleLimitLow)) then
39:            angleBefore ← angleBefore1
40:        else
41:            if angleBefore1 >= angleLimitHigh then
42:                angleBefore ← angleLimitHigh
43:            else
44:                angleBefore ← angleLimitLow
45:            end if
46:        end if
47:    end if
48:    if after then
49:        threshAngleDegrees ← (angleAfter + angleOfNode)
50:    else
51:        threshAngleDegrees ← (angleOfNode − angleAfter)
52:    end if
53:    threshAngleRadians ← threshAngleDegrees/(180∗PI)
54:    threshold.x ← circleCentre.x + (radius × cos(threshAngleRadians))
55:    threshold.y ← circleCentre.y + (radius × sin(threshAngleRadians))
56:    return Coord threshold
57: end function
```

case, the Optimized Link State Routing Protocol (OLSR) [43] is used. This due to the heavy load it places on the network in terms of energy consumption. This is an unconventional use of this protocol but for our requirements it proves effective.

5.2. Network Parameters

The emphasis of this study is on updating the threshold in real-time, after each pass of the MSN. As such, in order to produce a fair and consistent comparison across the three different sink speeds, it is important to ensure that for each test scenario the node has the same number of opportunities to adjust its threshold. Therefore, simulation times are such that the MSN will complete 40 circuits of the network, whatever speed it may be travelling at. The time taken to traverse the network is based on the network size and thus, the size of the circle path circumference.

The interference distance of each node is calculated using Eq. (2) [44]:

$$interferenceDistance = \left(\frac{\left(\frac{SoL}{Freq}\right)^2 \times Power}{16 \times PI^2 \times 10^{\frac{SAT}{10}}} \right)^{\frac{1.0}{Alpha}} \quad (2)$$

Here *SoL* represents the speed of light; *Freq* is the carrier frequency of the node; *Power* is the transmitter power of the node; *SAT* represents the signal attenuation threshold; *Alpha* is the path loss alpha. This is of particular importance as it is altered to affect the interference distance. As such, four different path loss alpha value are used, resulting in the four different interference ranges. As the path loss alpha increases, so the interference range decreases. With the path loss alpha simulating signal loss in normal situations such as interference from obstacles.

The sensitivity value used is an adjustment to ensure there is no node overload in the network. As such, node overload may result from a high sensitivity value and therefore, to reduce the number of received signals, sensitivity is adjusted from $-85dBm$ to $-75dBm$. All other parameters can be found in Table 1.

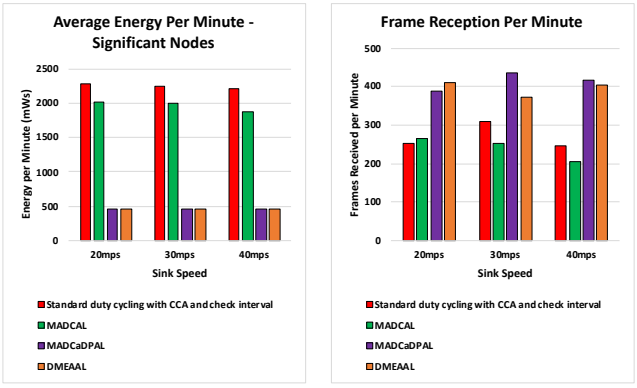
All parameters are consistent across all simulation runs, with power levels high to work in conjunction with the aforementioned network layer protocol, such that power consumption is subsequently also high in order to accelerate tests and achieve results faster. However, the speed of the sink node and the interference distance of the nodes are the significant metrics in this study and are altered accordingly. The path loss alpha is adjusted across four different values, as detailed in the test parameters, this in order to alter the size of the interference distance, which decreases as the alpha value increases.

5.3. MAC Protocols

Four different MAC implementations are evaluated. Firstly, the existing lightweight MAC implementation using standard duty cycling techniques with no allowance for sink mobility. This implementation reflects the core of the IEEE 802.15.4 standard [6, 7]. Next the MADCAL [10][11] algorithm, with a communication threshold implemented between significant nodes and the sink in order to affect the SLEEP function. Then the MADCaDPAL [20] algorithm is evaluated, with the threshold remaining but now influencing CCA and the sending of preambles in order that communication ends as the sink exits the threshold. Finally we evaluate the new DMEAAL algorithm, with the aim now to balance energy consumption across significant nodes.

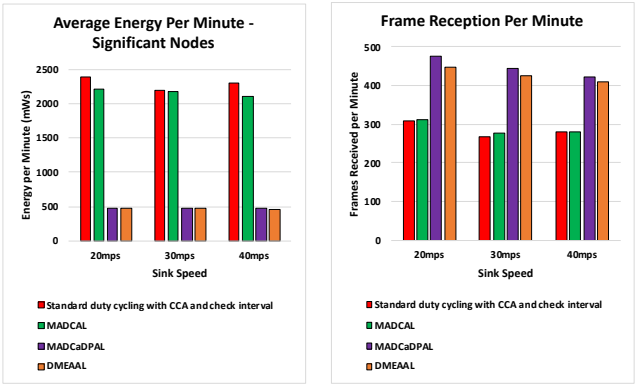
Table 1
Simulation Parameters

Parameters	Values
Number of Nodes (Static)	25
Grid Topology Size	$200m \times 200m$
MSN Path Radius	$150m$
MSN Start Position	$x = 400m, y = 250m$
MSN Speed (metres per second)	$20mps, 30mps, 40mps$
Simulation Time	$1884.95559215388s$, $1256.63706143592s$, $942.47779607694s$
Number of Circuits of Network	40
Interference Distance (4 candidate values)	$77.52m, 69.13m, 62.02m$, $55.94m$
Number of Runs	5
Path-loss Alpha (4 candidate values)	$1.85, 1.9, 1.95, 2$
Max Sending Power	$1.0mW$
Signal Attenuation Threshold	$-85dBm$
Sensitivity	$-75dBm$
Carrier Frequency	$2.4GHz$
Transmitter Power	$1.0mW$
Thermal Noise	$-85dBm$
Signal to Noise Ratio Threshold	$4dB$
Battery Capacity	$594000mWs$
Check Interval	$0.01s$
Slot Duration	$0.1s$



(a) Average Energy (b) Frame Delivery

Figure 9: Average Energy Consumption of Significant Nodes (mWs) and Frame Delivery to Sink. Interference Range $77.52m$



(a) Average Energy (b) Frame Delivery

Figure 10: Average Energy Consumption of Significant Nodes (mWs) and Frame Delivery to Sink. Interference Range $69.13m$

5.4. Evaluation Results

Results show the effect of an MSN on significant nodes, as highlighted in Figures 3 and 4. Results are two-fold, firstly comparing the results of the four different MAC implementations in order to demonstrate the effect of the DMEAAL algorithm on average energy consumption and frame delivery in comparison.

The second set of results demonstrate the energy balancing effect of the DMEAAL algorithm. Results are shown for MADCaDPAL and DMEAAL in order to compare the range of energy consumption across individual significant nodes.

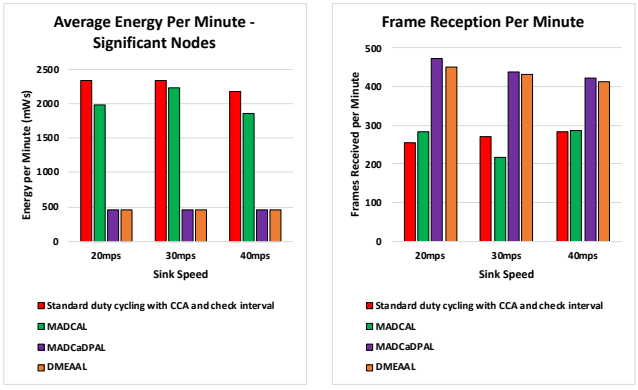
5.4.1. Results - Average Energy Consumption and Frame Delivery, Grid Topology

Figures 9-12 show the average energy consumption of all significant nodes for all four MAC implementations. This is presented alongside the frame delivery to the MSN for each scenario. The purpose of these results is no longer to show improvement in energy consumption or frame delivery by any measurement. In this case these results are to show that implementing the DMEAAL algorithm does not adversely affect performance in these areas. In terms of average energy consumption, in particular, as much as possible we would seek for the results for the MADCaDPAL algorithm and those when DMEAAL is in use to be the same or very close. The subsequent results of energy balancing across significant nodes show the sought improvements that we have been highlighted.

What is immediately clear from observing the average energy consumption levels in Figure 9a is that the same benefit to be gained from using the MADCaDPAL algorithm remains when using DMEAAL. The target of having energy consumption virtually the same is also seen to be achieved. In terms of frame delivery it can be seen that in some cases, balanced energy consumption may result in a slight decrease. However, in all cases illustrated in in Figure 9b, frame delivery remains more efficient than when MADCAL and the original MAC implementation are in use.

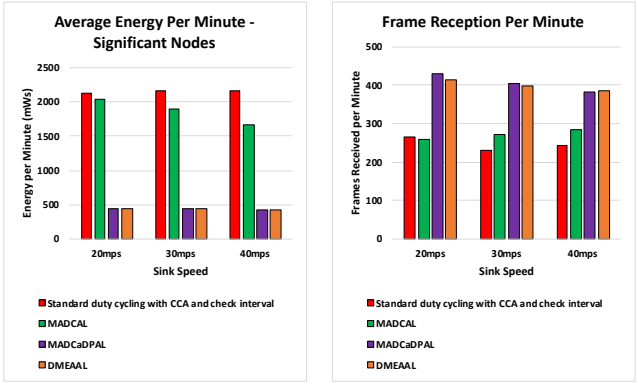
In Figure 10 it can be seen that this pattern repeats as the interference range lessens. Average energy consumption remains at the same level, as seen in Figure 10a, but frame delivery can be seen to have decreased slightly in Figure 10b.

Again, when observing results in Figure 11, it can be observed that DMEAAL keeps average energy consumption at the same level, as can be seen in Figure 11a, with frame delivery only slightly less, as seen in Figure 11b. This even as



(a) Average Energy (b) Frame Delivery

Figure 11: Average Energy Consumption of Significant Nodes (mWs) and Frame Delivery to Sink. Interference Range 62.02m



(a) Average Energy (b) Frame Delivery

Figure 12: Average Energy Consumption of Significant Nodes (mWs) and Frame Delivery to Sink. Interference Range 55.94m

the interference range has constricted to just over 60m. Even as the interference range lessens to just over 1 hop results remain much the same. as can be seen in Figure 12a with regard to energy consumption. In this case frame delivery is now within a small margin of MADCaDPAL, as seen in Figure 12b.

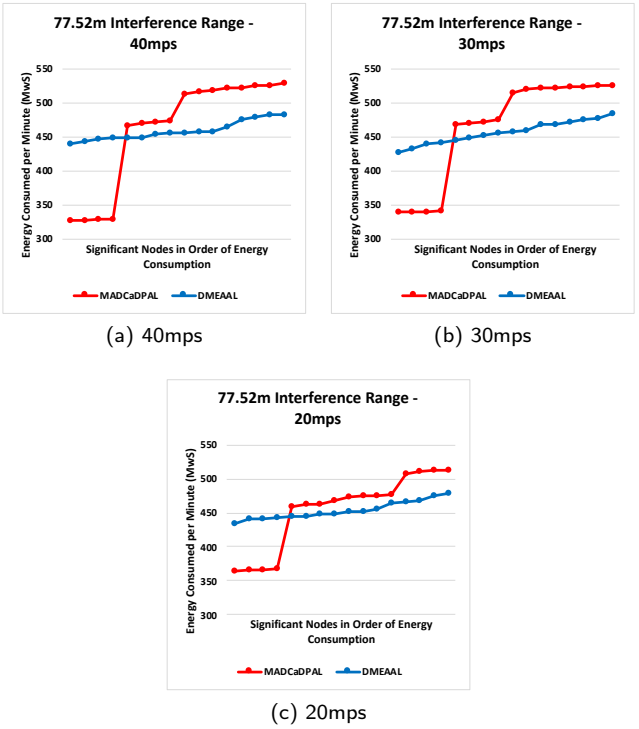
5.4.2. Results - Balanced Energy Consumption, Grid Topology

Figures 13-16 now demonstrate the ultimate result of the DMEAAL algorithm. For each different interference range the results of each sink speed are shown with the energy consumption per minute of each significant node from lowest to highest. As such, this demonstrates the difference in energy consumption between when the MADCaDPAL algorithm is in use and the DMEAAL algorithm. The aim being for energy consumption across significant nodes to all be closer in value.

Figure 13 shows the results with DMEAAL in use for an interference range of 77.52m. With the differing sink speeds shown of 40mps in Figure 13a, 30mps in Figure 13b and 20mps in Figure 13c. It can be clearly observed that with DMEAAL in use, the significant nodes all expend energy around the same level. As such, in all cases this brings down the energy consumption of 12 of the 16 significant nodes by utilising the excessive energy consumption of just 4. Despite how effective MADCaDPAL has been in reducing energy consumption, over an extended period of network time, a benefit such as this could have a major effect on network lifetime.

Figure 14 shows the results with DMEAAL in use for an interference range of 69.13m. With the differing sink speeds shown of 40mps in Figure 14a, 30mps in Figure 14b and 20mps in Figure 14c. Again, similar results can be observed to the previous figure. The DMEAAL algorithm is clearly evening out energy consumption across significant nodes, unaffected by the smaller interference range.

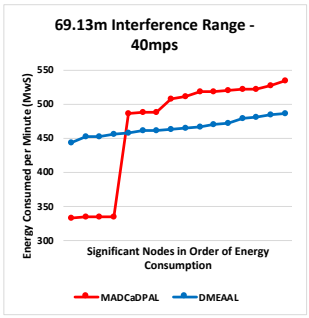
Figure 15 shows the results with DMEAAL in use for an interference range of 62.02m. With the differing sink speeds shown of 40mps in Figure 15a, 30mps in Figure 15b and 20mps in Figure 15c. Even as the interference range lessens this does not affect the results, with all significant nodes now much closer to the target energy consumption, that being the average energy consumed with the MADCaDPAL algorithm in use. This is repeated again in Figure 16 with an interference range of 55.94m.



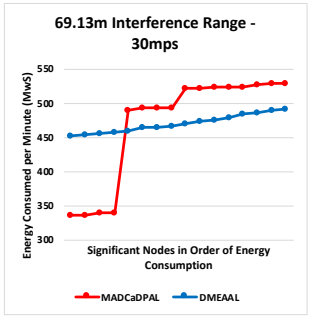
(a) 40mps (b) 30mps

(c) 20mps

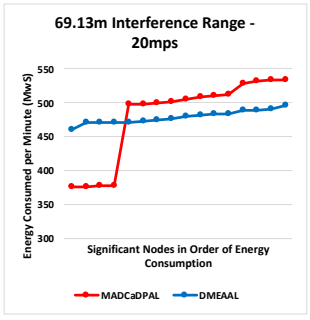
Figure 13: Balanced Energy Consumption of Significant Nodes (mWs). Interference Range 77.52m



(a) 40mps

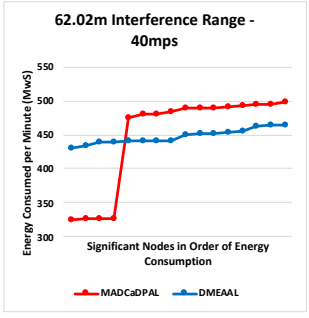


(b) 30mps

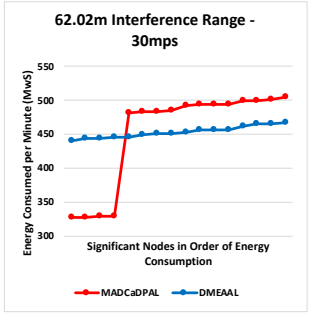


(c) 20mps

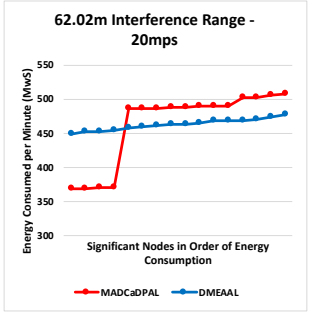
Figure 14: Balanced Energy Consumption of Significant Nodes (mWs). Interference Range $69.13m$



(a) 40mps

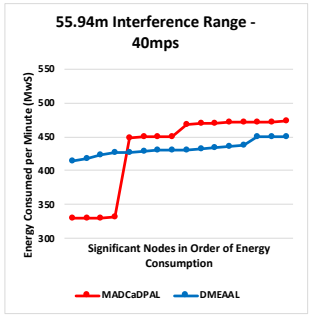


(b) 30mps

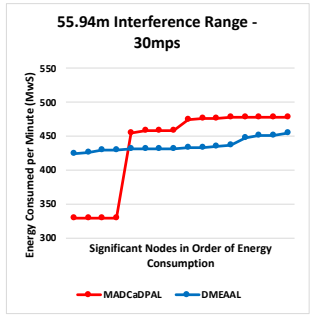


(c) 20mps

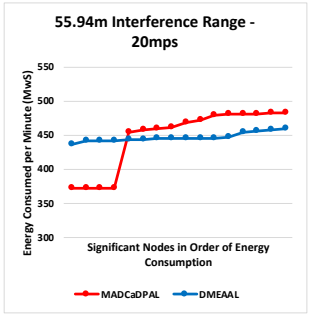
Figure 15: Balanced Energy Consumption of Significant Nodes (mWs). Interference Range $62.02m$



(a) 40mps



(b) 30mps



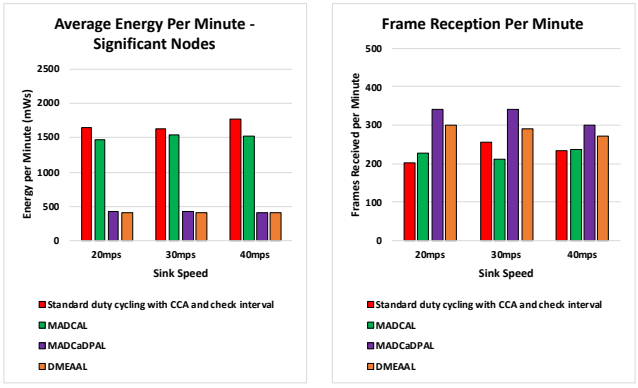
(c) 20mps

Figure 16: Balanced Energy Consumption of Significant Nodes (mWs). Interference Range $55.94m$

5.4.3. Results - Average Energy Consumption and Frame Delivery, Random Topology

Figures 17-20 show the average energy consumption of all significant nodes for all four MAC implementations. This is presented alongside the frame delivery to the MSN for each scenario. Again, we do not seek to show improvement in energy consumption or frame delivery by any measurement. In this case these results are to show that implementing the DMEAAL algorithm does not adversely affect performance in these areas, especially when considering a more strained topology. In terms of average energy consumption, in particular, as much as possible we would seek for the results for the MADCaDPAL algorithm and those when DMEAAL is in use to be the same or very close. With frame reception proving more problematic with this topology in use, it is also of interest to see if there has been an adverse effect on these results when using DMEAAL. However, the subsequent results of energy balancing across significant nodes remain the aim of this study.

It is clear, as with the grid topology, from observing the average energy consumption levels in Figure 17, that the same benefit to be gained from using the MADCaDPAL algorithm remains when using DMEAAL. The target of having energy consumption virtually the same is also seen to be achieved in Figure 17a. Again though, in terms of frame delivery, it can be seen in Figure 17b that balanced energy consumption may result in a slight decrease. However, in all cases illustrated frame delivery remains higher than when MADCAL and the original MAC implementation are in use.



(a) Average Energy (b) Frame Delivery

Figure 17: Average Energy Consumption of Significant Nodes (mWs) and Frame Delivery to Sink. Interference Range 77.52m

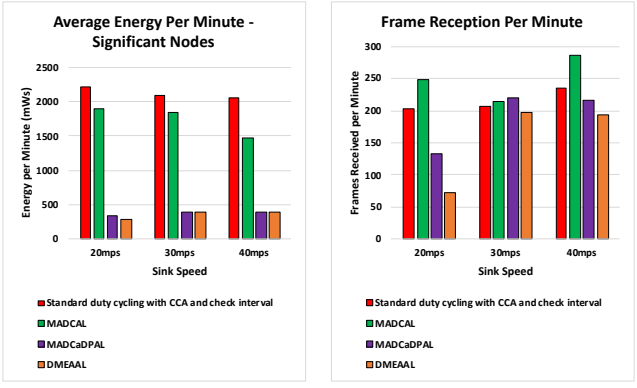
In Figure 18 it can be seen that this pattern repeats as the interference range lessens. Average energy consumption remains at the same level, as seen in Figure 18a, but frame delivery is slightly less as seen in Figure 18b.

When observing results in Figure 19 it can again be seen that DMEAAL keeps average energy consumption at the same level, as seen in Figure 19a, with this also true for Figure 20a. This is significant as the interference range lessens, now to just above the distance of one-hop. When considering frame delivery, this constriction of interference range also proved problematic for the MADCaDPAL algorithm. The topology in use has clusters of nodes but also large spaces between them. As such, attempts to control energy consumption and a reduction in communication thresholds may result in a slight reduction in frame delivery. This repeats with the use of DMEAAL, although not excessively aside from one result when the sink moves at 20mps with a 62.02m interference range, as seen in Figure 19b. This combination clearly results in a threshold that is not effective in this environment. As such, this highlights the need for a certain amount of control in the density of node placement.

5.4.4. Results - Balanced Energy Consumption, Random Topology

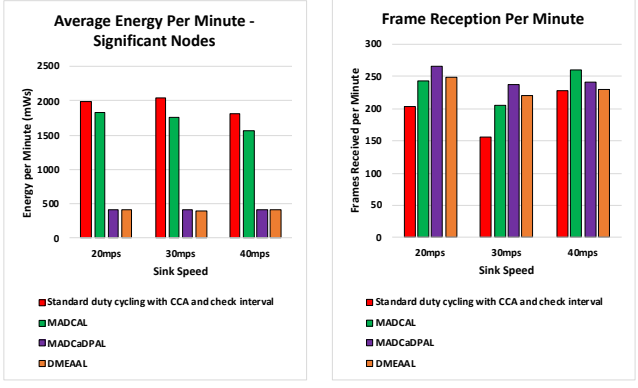
Figures 21-24 illustrate the results of the ultimate aim of the DMEAAL algorithm, that being to balance energy consumption across significant nodes. For each different interference range the results of each sink speed are shown with the energy consumption per minute of each significant node from lowest to highest. Different from the use of the grid topology is that as the interference range constricts, fewer nodes take the role of significant node. As before, however, results demonstrate the difference in energy consumption between when the MADCaDPAL algorithm is in use and the DMEAAL algorithm. The aim being for energy consumption across significant nodes to all be closer in value.

Figure 21 shows the results with DMEAAL in use for an interference range of 77.52m. With the differing sink speeds shown of 40mps in Figure 21a, 30mps in Figure 21b and 20mps in Figure 21c. It can be clearly observed that with DMEAAL in use, energy consumption is spread more evenly



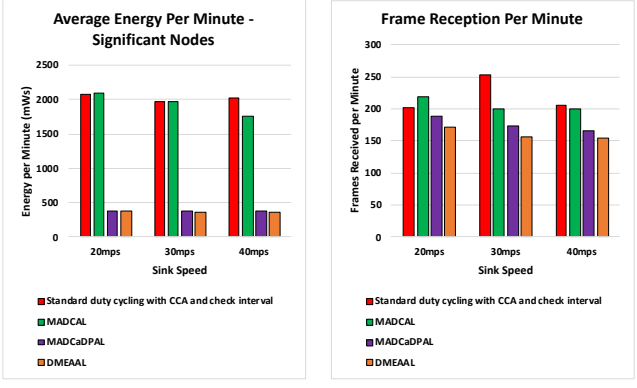
(a) Average Energy (b) Frame Delivery

Figure 19: Average Energy Consumption of Significant Nodes (mWs) and Frame Delivery to Sink. Interference Range 62.02m



(a) Average Energy (b) Frame Delivery

Figure 18: Average Energy Consumption of Significant Nodes (mWs) and Frame Delivery to Sink. Interference Range 69.13m



(a) Average Energy (b) Frame Delivery

Figure 20: Average Energy Consumption of Significant Nodes (mWs) and Frame Delivery to Sink. Interference Range 55.94

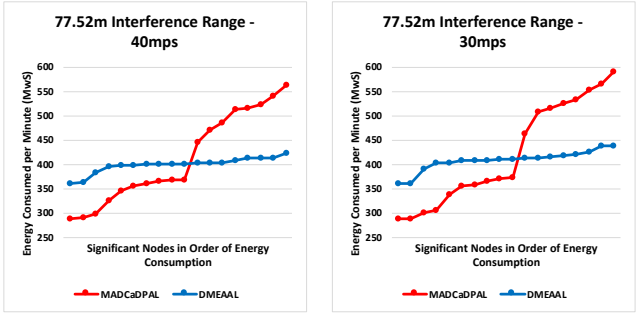
across significant nodes. What can be observed in contrast to the grid topology, is that with MADCaDPAL in use the spread of energy is even more stark. As such, the potential to increase network lifetime is even greater and DMEAAL succeeds in this.

Figure 22 shows the results with DMEAAL in use for an interference range of 69.13m. With the differing sink speeds shown of 40mps in Figure 22a, 30mps in Figure 22b and 20mps in Figure 22c. Again, similar results can be observed to the previous figure, with DMEAAL effective in evening out energy consumption across significant nodes. Despite the smaller interference range.

Figure 23 shows the results with DMEAAL in use for an interference range of 62.02m. With the differing sink speeds shown of 40mps in Figure 23a, 30mps in Figure 23b and 20mps in Figure 23c. Results again follow a similar pattern. However, Figure 23c showing a sink speed of 20mps can be largely discounted due to the low level of frame delivery in this scenario. In this case it would have to be accepted that the MADCaDPAL algorithm is more effective. However, even in the much smaller interference range in Figure 24, the effectiveness of the DMEAAL algorithm can be observed.

6. Conclusion and Future Work

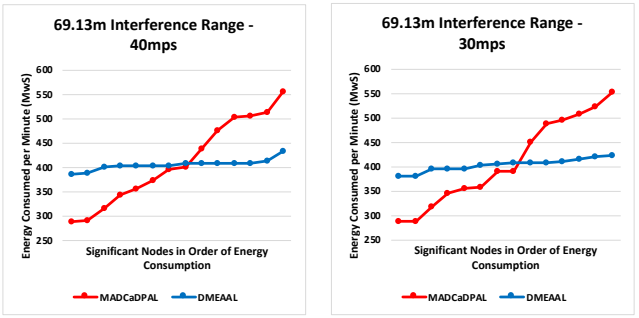
In this study we have proposed DMEAAL, for use with WSNs utilising a MSN, built on the communication threshold originally developed in the MADCAL algorithm [10, 11]. DMEAAL utilises predictable sink mobility and self



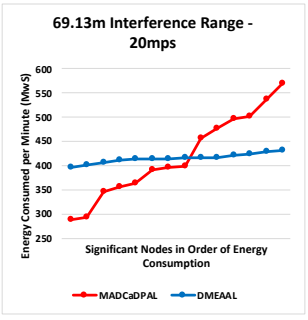
(a) 40mps (b) 30mps

(c) 20mps

Figure 21: Balanced Energy Consumption of Significant Nodes (mWs). Interference Range $77.52m$

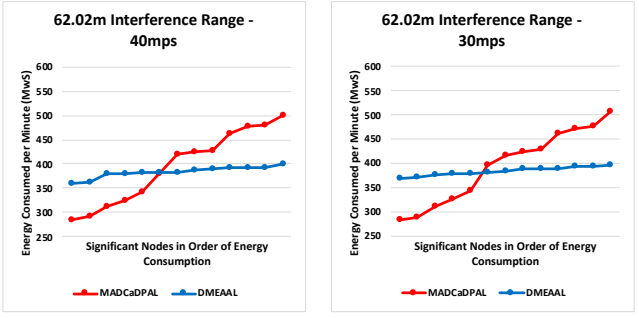


(a) 40mps (b) 30mps

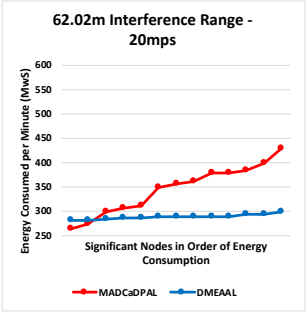


(c) 20mps

Figure 22: Balanced Energy Consumption of Significant Nodes (mWs). Interference Range $69.13m$

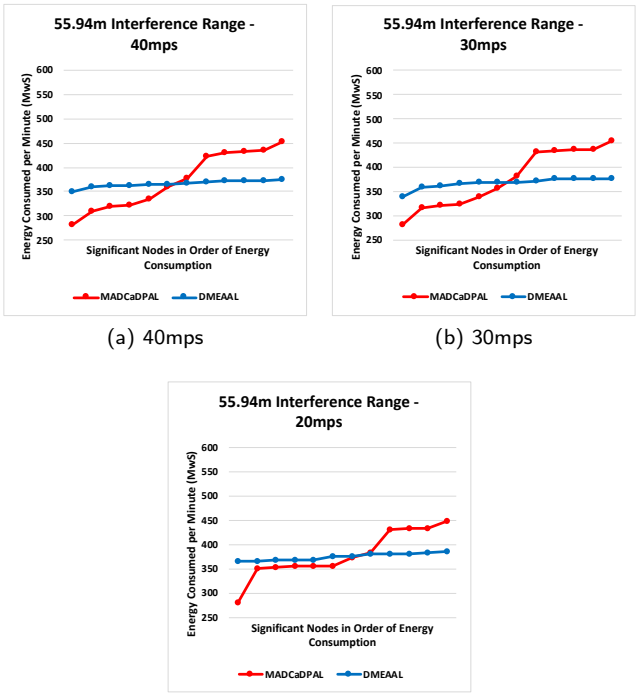


(a) 40mps (b) 30mps



(c) 20mps

Figure 23: Balanced Energy Consumption of Significant Nodes (mWs). Interference Range $62.02m$



(a) 40mps (b) 30mps

(c) 20mps

Figure 24: Balanced Energy Consumption of Significant Nodes (mWs). Interference Range 55.94m

knowledge of location amongst static nodes. This in order that energy consumption across significant nodes may be balanced and that *dead* energy consumption may be reduced. The evaluation results show that the levels of energy consumption and frame delivery to the sink achieved by the MADCaDPAL algorithm, are generally retained with DMEAAL in use. Additionally, we have demonstrated that a cross-layer approach, determining the current battery levels within a node along with knowledge of target energy consumption, results in balanced energy consumption across significant nodes. Thus, improving network lifetime. As such, finally eliminating energy spikes which contribute to the *energy hole* [9] or *hotspot* problem. In altering the communication threshold accordingly based on target energy consumption versus current average energy consumption, we now have an entirely dynamic solution. This is shown to be effective in both a controlled, grid network formation and when we utilise a more *strained*, random topology.

For future work we would seek to implement the practical applications proposed for the DMEAAL algorithm and ultimately utilise this within test beds, rather than merely simulation environments.

References

[1] Chen Chen, Yuru Zhang, Mohammad R. Khosravi, Qingqi Pei, and Shaohua Wan. An Intelligent Platooning Algorithm for Sustainable Transportation Systems in Smart Cities. *IEEE Sensors Journal*, pages 1–1, aug 2020. ISSN 1530-437X. doi: 10.1109/jsen.2020.3019443.

[2] Salman Ali, Adnan Ashraf, Saad Bin Qaisar, Muhammad Kamran Afridi, Husnain Saeed, Sidra Rashid, Emad A. Felemban, and Adil Amjad Sheikh. SimpliMote: A Wireless Sensor Network Monitoring Platform for Oil and Gas Pipelines. *IEEE Systems Journal*, 12 (1):778–789, mar 2018. ISSN 1932-8184. doi: 10.1109/JSYST.2016.2597171.

[3] Adnan Ahmed, Kamalrulnizam Abu Bakar, Muhammad Ibrahim Channa, Abdul Waheed Khan, and Khalid Haseeb. Energy-aware and secure routing with trust for disaster response wireless sensor network. *Peer-to-Peer Networking and Applications*, 10(1):216–237, jan 2017. ISSN 1936-6442. doi: 10.1007/s12083-015-0421-4.

[4] Mohammad Ammad Uddin, Ali Mansour, Denis Le Jeune, Mohammad Ayaz, and El Hadi M. Aggoune. Uav-assisted dynamic clustering of wireless sensor networks for crop health monitoring. *Sensors (Switzerland)*, 18(2), 2018. ISSN 14248220. doi: 10.3390/s18020555.

[5] Lin Chen and Kaigui Bian. Neighbor discovery in mobile sensing applications: A comprehensive survey. *Ad Hoc Networks*, 2016. ISSN 15708705. doi: 10.1016/j.adhoc.2016.05.005.

[6] IEEE 802.15.4, 2016. URL http://www.ieee802.org/15/pub/TG4.html.

[7] G Montenegro, N Kushalnagar, J Hui, and D Culler. RFC4944: Transmission of IPv6 Packets over IEEE 802.15.4 Networks. 2007.

[8] C. Cano, B. Bellalta, A. Sfairopoulou, and M. Oliver. Low energy operation in WSNs: A survey of preamble sampling MAC protocols. *Computer Networks*, 55(15):3351–3363, 2011. ISSN 13891286. doi: 10.1016/j.comnet.2011.06.022.

[9] Xiaofeng Tang and Li Xie. Data Collection Strategy in Low Duty Cycle Wireless Sensor Networks with Mobile Sink. *International Journal of Communications, Network and System Sciences*, 10(05): 227–239, 2017. ISSN 1913-3715. doi: 10.4236/ijcns.2017.105B023.

[10] Craig Thomson, Isam Wadhaj, Zhiyuan Tan, and Ahmed Al-dubai. Mobility Aware Duty Cycling Algorithm (MADCAL) in Wireless Sensor Network with Mobile Sink Node. In *2019 IEEE International Conference on Smart Internet of Things (IEEE SmartIoT 2019)*, Tianjin, China, 2019. doi: 10.1109/SmartIoT.2019.00037.

[11] Craig Thomson, Isam Wadhaj, Zhiyuan Tan, and Ahmed Al-Dubai. Mobility Aware Duty Cycling Algorithm (MADCAL) A Dynamic Communication Threshold for Mobile Sink in Wireless Sensor Network. *Sensors*, 19(22):4930, nov 2019. ISSN 1424-8220. doi: 10.3390/s19224930.

[12] Tao Tang, Tao Hong, Haohui Hong, Senyuan Ji, Shahid Mumtaz, and Mohamed Cheriet. An Improved UAV-PHD Filter-Based Trajectory Tracking Algorithm for Multi-UAVs in Future 5G IoT Scenarios. *Electronics*, 8(10):1188, oct 2019. ISSN 2079-9292. doi: 10.3390/electronics8101188.

[13] S. Sudhakar, V. Vijayakumar, C. Sathiya Kumar, V. Priya, Logesh Ravi, and V. Subramaniyaswamy. Unmanned Aerial Vehicle (UAV) based Forest Fire Detection and monitoring for reducing false alarms in forest-fires. *Computer Communications*, 149:1–16, jan 2020. ISSN 1873703X. doi: 10.1016/j.comcom.2019.10.007.

[14] M Jamalabdollahi and S A R Zekavat. Joint Neighbor Discovery and Time of Arrival Estimation in Wireless Sensor Networks via OFDMA. *IEEE Sensors Journal*, 15(10):5821–5833, 2015. doi: 10.1109/JSEN.2015.2449079.

[15] Keyu Wang, Xufei Mao, and Yunhao Liu. BlindDate: A neighbor discovery protocol. *IEEE Transactions on Parallel and Distributed Systems*, 26(4):949–959, 2015. ISSN 10459219. doi: 10.1109/TPDS.2014.2316159.

[16] R. Pozza, M. Nati, S. Georgoulas, K. Moessner, and A. Gluhak. Neighbor discovery for opportunistic networking in internet of things scenarios: A survey. *IEEE Access*, 3:1101–1131, 2015. ISSN 21693536. doi: 10.1109/ACCESS.2015.2457031.

[17] Shusen Yang, Usman Adeel, Yad Tahir, and Julie A Mccann. Practical Opportunistic Data Collection in Wireless Sensor Networks with Mobile Sinks. *IEEE Transactions on Mobile Computing*, (99):1–14, 2016. doi: 10.1109/TMC.2016.2595574.

[18] Georgios Z. Papadopoulos, Vasileios Kotsiou, Antoine Gallais, Periklis Chatzimisios, and Thomas Noel. Low-power neighbor discovery for mobility-aware wireless sensor networks. *Ad Hoc Networks*, 2016. ISSN 15708705. doi: 10.1016/j.adhoc.2016.05.011.

[19] Andrea Hess, Esa Hyytia, and Jorg Ott. Efficient neighbor discovery in mobile opportunistic networking using mobility awareness. In *2014 6th International Conference on Communication Systems and Networks, COMSNETS 2014*, 2014. ISBN 9781479936359. doi: 10.1109/COMSNETS.2014.6734890.

[20] Craig Thomson, Isam Wadhaj, Ahmed Al-Dubai, and Zhiyuan Tan. A New Mobility Aware Duty Cycling and Dynamic Preambling Algorithm for Wireless Sensor Network. In *The 2020 IEEE 6th World Forum on The Internet of Things (WF-IoT 2020)*, pages 1–6. Institute of Electrical and Electronics Engineers (IEEE), oct 2020. doi: 10.1109/wf-iot48130.2020.9221036.

[21] Shaohua Wan, Yudong Zhang, and Jia Chen. On the Construction of Data Aggregation Tree with Maximizing Lifetime in Large-Scale Wireless Sensor Networks. *IEEE Sensors Journal*, 16(20):7433–7440, oct 2016. ISSN 1530437X. doi: 10.1109/JSEN.2016.2581491.

[22] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys and Tutorials*, 19(4):2322–2358, oct 2017. ISSN 1553877X. doi: 10.1109/COMST.2017.2745201.

[23] Ao Zhou, Shangguang Wang, Shaohua Wan, and Lianyong Qi. LMM: latency-aware micro-service mashup in mobile edge computing environment. *Neural Computing and Applications*, 32(19):15411–15425, oct 2020. ISSN 14333058. doi: 10.1007/s00521-019-04693-w.

[24] Xin Li, Songtao Guo, and Pan Li. Energy-efficient data collection scheme based on mobile edge computing in WSNs. In *Proceedings - 2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks, MSN 2019*, pages 95–100. Institute of Electrical and Electronics Engineers Inc., dec 2019. ISBN 9781728152127. doi: 10.1109/MSN48538.2019.00030.

[25] Lichao Yang, Heli Zhang, Ming Li, Jun Guo, and Hong Ji. Mobile edge computing empowered energy efficient task offloading in 5G. *IEEE Transactions on Vehicular Technology*, 67(7):6398–6409, jul 2018. ISSN 00189545. doi: 10.1109/TVT.2018.2799620.

[26] Yixue Hao, Min Chen, Long Hu, M. Shamim Hossain, and Ahmed Ghoneim. Energy Efficient Task Caching and Offloading for Mobile Edge Computing. *IEEE Access*, 6:11365–11373, mar 2018. ISSN 21693536. doi: 10.1109/ACCESS.2018.2805798.

[27] Jie Xu, Lixing Chen, and Shaolei Ren. Online Learning for Offloading and Autoscaling in Energy Harvesting Mobile Edge Computing. In *IEEE Transactions on Cognitive Communications and Networking*, volume 3, pages 361–373. Institute of Electrical and Electronics Engineers Inc., sep 2017. doi: 10.1109/TCCN.2017.2725277.

[28] Shubhani Aggarwal and Neeraj Kumar. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, 149:270–299, jan 2020. ISSN 1873703X. doi: 10.1016/j.comcom.2019.10.014.

[29] Moorthi and R. Thiagarajan. Energy consumption and network connectivity based on Novel-LEACH-POS protocol networks. *Computer Communications*, 149:90–98, jan 2020. ISSN 1873703X. doi: 10.1016/j.comcom.2019.10.006.

[30] Jin Wang, Jiayi Cao, Sai Ji, and Jong Hyuk Park. Energy-efficient cluster-based dynamic routes adjustment approach for wireless sensor networks with mobile sinks. *Journal of Supercomputing*, 73(7):3277–3290, jul 2017. ISSN 15730484. doi: 10.1007/s11227-016-1947-9.

[31] Ramin Yarinezhad and Amir Sarabi. Reducing delay and energy consumption in wireless sensor networks by making virtual grid infrastructure and using mobile sink. *AEU - International Journal of Electronics and Communications*, 84(April 2017):144–152, 2018. ISSN 16180399. doi: 10.1016/j.aeue.2017.11.026.

[32] Lina Aliouat and Zibouda Aliouat. Improved WSN life time duration through adaptive clustering, duty cycling and sink mobility. In *ACM International Conference Proceeding Series*, pages 36–40, 2016. ISBN 9781450347617. doi: 10.1145/3012258.3012269.

[33] Ramin Yarinezhad. Reducing delay and prolonging the lifetime of wireless sensor network using efficient routing protocol based on mobile sink and virtual infrastructure. *Ad Hoc Networks*, 84:42–55, 2019. ISSN 15708705. doi: 10.1016/j.adhoc.2018.09.016.

[34] Chao Wu, Yuan'An Liu, Fan Wu, Wenhao Fan, and Bihua Tang. Graph-Based Data Gathering Scheme in WSNs with a Mobility-Constrained Mobile Sink. *IEEE Access*, 5:19463–19477, 2017. ISSN 21693536. doi: 10.1109/ACCESS.2017.2742138.

[35] Ratijit Mitra and Suraj Sharma. Proactive data routing using controlled mobility of a mobile sink in Wireless Sensor Networks. *Computers and Electrical Engineering*, 70(September 2016):21–36, 2018. ISSN 00457906. doi: 10.1016/j.compeleceng.2018.06.001.

[36] Shuai Gao, Hongke Zhang, and Sajal K Das. Efficient data collection in wireless sensor networks with path-constrained mobile sinks. *Mobile Computing, IEEE Transactions on*, 10(4):592–608, 2011.

[37] Sanu Thomas and Thomaskutty Mathew. Intelligent path discovery for a mobile sink in wireless sensor network. In *8th International Conference on Advances in Computing and Communication (ICACC-2018)*, volume 143, pages 749–756. Elsevier B.V., 2018. doi: 10.1016/j.procs.2018.10.430.

[38] Nimisha Ghosh and Indrajit Banerjee. An energy-efficient path determination strategy for mobile data collectors in wireless sensor network. *Computers and Electrical Engineering*, 48:417–435, 2015. ISSN 00457906. doi: 10.1016/j.compeleceng.2015.09.004.

[39] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Sudipta Sinha, Ashish Kapoor, Madhusudhan Sudarshan, and Sean Stratman. FarmBeats: An IoT Platform for Data-Driven Agriculture. In *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI '17)*, pages 515–529, 2017. ISBN 978-1-931971-37-9. doi: 10.1093/abbs/gmw048.

[40] OMNeT++ Discrete Event Simulator, 2015. URL https://omnetpp.org/.

[41] MiXiM. URL http://mixim.sourceforge.net/.

[42] INETMANET INSTALLATION | Inetmanet Installation Guide.

[43] T Clausen. RFC 3626 - Optimized Link State Routing Protocol - OLSR. Technical report, 2003.

[44] IntRange, 2019. URL https://github.com/inetmanet/inetmanet/blob/master/src/underTest/wpan/linklayer/ieee802154/phyLayer/Ieee802154Phy.cc.

Author biography without author photo. Author biography.

Author biography with author photo. Author biography.

Author biography with author photo. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography.