Wireless Networks
https://doi.org/10.1007/s11276-021-02580-8

A mobility aware duty cycling and preambling solution for wireless sensor network with mobile sink node

Craig Thomson[1] · Isam Wadhaj[1] · Zhiyuan Tan[1] · Ahmed Al-Dubai[1]

Accepted: 23 February 2021

Abstract
Utilising the mobilisation of a sink node in a wireless sensor network to combat the energy hole, or *hotspot* issue, is well referenced. However, another issue, that of *energy spikes* may remain. With the mobile sink node potentially communicating with some nodes more than others. In this study we propose the Mobility Aware Duty Cycling and Dynamic Preambling Algorithm (MADCaDPAL). This algorithm utilises an existing solution where a communication threshold is built between a mobile sink node using predictable mobility and static nodes on its path. MADCaDPAL bases decisions relating to node sleep function, moving to clear channel assessment and the subsequent sending of preambles on the relation between the threshold built by the static node and the position of the mobile sink node. MADCaDPAL achieves a reduction in average energy consumption of up to 80%, this when used in conjunction with a lightweight carrier-sense multiple access based MAC implementation. Maximum energy consumption amongst individual nodes is also brought closer to the average, reducing energy spikes and subsequently improving network lifetime. Additionally, frame delivery to the sink is improved overall.

Keywords Wireless sensor networks · Sink mobility · Energy holes · Preambles · CCA

1 Introduction

Wireless sensor networks (WSN), represent great challenges in the area of energy consumption. These tiny devices of low memory and processing capacity, as well as limited battery power, have potential for use in inhospitable locations with applications such as in deep sea oil and gas [1], disaster recovery [2] and agriculture [3]. As such the issue of battery replacement arises. Given the difficulty this may represent, these devices must adopt duty cycling methods in order to conserve power and increase network lifetime. Duty cycling involves nodes sleeping when not idle and can thus have great benefit in reducing energy conservation [4]. This is controlled at the MAC layer, with different approaches to be implemented. For example, in carrier-sense multiple access (CSMA) MAC implementation, the IEEE 802.15.4 standard [5, 6], clear channel assessment (CCA) is utilised before sending data. This operates alongside the sending of preambles, keeping the channel open as data is sent. When transmitted, a preamble is required to be at least the length of the sleep period of the receiver, and this has been shown to have benefit regarding energy consumption [7]. However, network functionality arises as an issue given the time a node may be asleep for. An inherent challenge is in ensuring that nodes may actually discover each other such that Neighbour Discovery (ND) can occur with duty cycling in place. In particular, that wake-up schedules overlap between nodes in order that ND may actually take place. Without this, communication between nodes would become impossible and the network would be rendered useless.

The IoT has moved into mobile environments in recent times [8–10], however, when considering a WSN as a static implementation, another issue arises. Due to the multi-hop to sink basis of WSNs they are prone to nodes closest to the sink taking on greater load and subsequently running out of energy faster. This may result in other nodes in the network being unable to communicate with the sink node, effectively ending the functionality of the entire WSN. This is

✉ Zhiyuan Tan
z.tan@napier.ac.uk

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

Published online: 20 March 2021

known as the *energy hole* [11] or *hotspot* problem. Sink node mobility is a common approach to combating this. The assumption here is that as the sink node moves, as does the responsibility for final delivery to it. Thus, the role of hotspot is spread across several nodes in the network, rather than just those who would be closest to the sink if stationary. These nodes are termed as *significant nodes* [12, 13]. Mobile sink nodes (MSNs) are utilised in various applications, such as within vehicles, located on robots or merely carried by a person. An emerging area is to utilise drones, or unmanned aerial vehicles (UAVs) to give their scientific name [14, 15]. However, merely mobilising the sink node does not guarantee that all nodes the sink passes shall expend energy at the same rate. Energy *spikes* may be problematic in the long term, with even a small difference in energy consumption resulting in certain nodes dying earlier than others as time passes.

Probabilistic [16] and deterministic [17] algorithms have formed the legacy approach to ND in WSNs. With deterministic approaches guaranteeing an overlap and probabilistic approaches resulting in a high probability. Although value is found in both approaches, the deterministic approach is the most commonly used. The long-tail discovery issue is considerable issue regarding the probabilistic approach, where a node may never be discovered at all [18]. This counters its otherwise greater efficiency in ensuring ND when compared to its deterministic counterpart. However, within the area of ND in WSNs, new studies such as those focusing on opportunistic routing approaches, have emerged. In opportunistic routing the approach is taken of decisions being made *on the fly* [19].

In this study, however, the key problem to be solved is within the area of mobility awareness in WSNs [20]. Here network behaviour is influenced by knowledge of mobility, in this case the mobility of the sink. With the network behaviour in question being that of ND. In using a MSN within a WSN another layer of complexity is added to the issue of ND, with nodes awakening when the sink is nearby preferable. Unlike other approaches, when utilising mobility awareness the aim is not to negate the mobility in the network. That being, that it would be a problem requiring to be *worked around*. A mobility aware approach seeks to use the very mobility patterns of mobile nodes in order to improve elements such as routing and data delivery [21]. This study seeks to improve and balance energy consumption in static nodes by using the mobility pattern of a MSN by which to influence duty cycling in the nodes it has direct communication 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.

As such, this work utilises the dynamic communication threshold between a MSN and static sink developed in our original study, the Mobility Aware Duty Cycling Algorithm (MADCAL) [12, 13], with predictable sink mobility. Within a static, significant node, the move from the SLEEP function to clear channel assessment (CCA) only occurs when the sink node is within the threshold of the static node. Otherwise this will wait until the threshold is reached, effectively leaving the node asleep. This has already shown benefit in both frame delivery to the sink and, ultimately, energy consumption. In this paper we propose the novel Mobility Aware Duty Cycling and Dynamic Preambling Algorithm (MADCaDPAL) to utilise the communication threshold created by MADCAL in order to further improve energy consumption and network efficiency.

This study extends our conference paper [22] as follows:

- *Algorithm* A new determination of the communication threshold is detailed in Algorithm 3 with a dynamic approach to the factor applied to the threshold size based on sink speed. Both algorithms, for the determination of a communication threshold and the threshold interval, described in detail including relevant equations.
- *Test Scenarios* In this journal version we utilise both a random topology in addition to the grid topology utilised in the conference paper. This demonstrates benefit in the use of MADCaDPAL even when used in a more *strained* topology. The topology layout can be found in Fig. 2, with significant nodes demonstrated for each interference range in Fig. 4. Resultantly, twice as many test results are provided, proving the effectiveness of the MADCaDPAL algorithm more extensively. In addition, related work has been extended and classified in Section 2.

In the development of MADCaDPAL we examine the relationship between the MSN and static sink threshold in finer detail. Establishing that whilst MADCAL is a crucial first step, in utilising sink position in relation to the static node to influence the behaviour of CCA and the sending of preambles, that further benefits can be achieved in relation to energy consumption and frame delivery. This whilst also countering the issue of *energy spikes* amongst significant nodes. Where, despite average energy across significant nodes improving, some nodes still take on a greater network load than others. MADCAL and, subsequently, MADCaDPAL are both implemented into each node independently. As such, there is no use of expensive beacons between nodes. However, this also means there is no knowledge of neighbouring nodes. As such, network density may cause overlaps of communication, resulting in the aforementioned energy spikes in some nodes, while others wait for a clear channel.

This paper is organised as follows. Section 2 examines related work in the area of the two proposed algorithms,

while Sect. 3 gives a review of Mobility Aware Duty Cycling, the development of the original MADCAL algorithm and the sink mobility pattern and network topologies in use. Section 4 details the MADCaDPAL algorithm, the problem statement, methodology and resultant algorithm with the testing and results following in Sect. 5. Section 6 concludes this paper and details future work.

2 Related work

In examining related work it can be found that the use of a MSN to influence the duty cycling of static nodes is a novel approach. Existing work where a MSN is utilised tend to focus on either network layer routing protocols or optimal path determination.

2.1 Dynamic use of preambles

In considering a dynamic approach to the use of preambles and the beneficial relationship between this and energy consumption, Chen et al. [4] analyse several factors in the delivery of packets. Initial preamble length is shortened as a result of the use of opportunistic forwarding. Subsequently, the relationship between delivery probability, length of preamble, density of the network and sleep duration of nodes is utilised, such that the preamble length may be adjusted based on this relationship. This is shown to have benefit in packet delivery, without incurring additional delay. Also, energy consumption is reduced by as much as twice.

2.2 Routing with mobile sink nodes

When examining research in the use of MSNs it becomes clear that the vast majority focus on network layer routing. As such, routing protocols for use with MSNs can be referred to as being in the categories of *hierarchical* and *non-hierarchical* [23]. *Hierarchical protocols* are defined as giving some nodes greater value than others, in order that the MSN is not required to advertise its presence to the entire network. In regard to hierarchical routing protocols this is generally in reference to the use of grids and clusters, used to divide the WSN into specific areas, requiring position awareness of nodes. Mobility of a sink node within a grid sector is invisible, but these approaches suffer from problems such as the high overhead of grid construction and nodes that form part of the grid becoming hotspots [24, 25]. As such, hotspot avoidance seen as one of the supposed main benefits of the use of MSNs. The hotspot issue could also occur in the use of clusters, with strain being placed on the cluster head. To avoid this the cluster heads can be rotated. TDMA schedules may also be enforced to aid in energy conservation, but with the additional requirement of utilising the MAC layer. Sink mobility may also need to be controlled when utilising a cluster method [26, 27]. Delay-tolerance can be utilised along with clustering techniques, although this of course adds to delay. However, this is shown to produce good results in regard to energy efficiency [28].

In this area [29] 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 network layer 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. However, this does rely on the exchange of messages, which may add to network overhead.

A similar study by [30] seeks to utilise *landmark-nodes* for use with a MSN in a WSN, again to reduce energy consumption. With these landmark-nodes identified as being one-hop from all nodes within its cluster and on the path of the sink, utilising Kalman filtering. As such, the MSN will stop and collect data from these nodes. Using a random walk sink mobility pattern, the landmark-node locations are continuously updated. Again this study demonstrates how proper use of an MSN can improve network behaviour in terms of energy consumption and packet delivery. By reducing the average hop count to the sink this demonstrates energy improvements of over 20% compared to existing alternative studies. This is also tested on a real test bed using robots. However, this network layer solution makes no allowances for MAC or Physical layer parameters being influenced by sink mobility despite identifying this in related works.

The path of the MSN is a consideration in previous works such as by [31]. Utilising a Voronoi diagram and rendezvous points, this work constructs a path for the MSN where routing may be conducted most efficiently with regard to delivery delay. As such, the authors claim improvement over existing works when considering fault tolerance and average waiting time.

The use of a predefined MSN path is proposed by [32], integrating both geographic and hierarchical routing. In this case networks are divided into partitions consisting of virtual grids. The MSN then travels through the virtual

grids and includes the possibility of having multiple MSNs, subsequently further reducing latency.

Predictable sink mobility is again in evidence as utilised by [33]. In this case a predictable circular path around the network is used, with forwarded nodes along the sink path used in combination with a virtual infrastructure to reduce network overhead. This demonstrating the potential inherent in the use of predictable sink mobility, which generally is found to be underused in existing literature. With the desire for optimal sink paths more prevalent.

3 Mobility aware duty cycling

3.1 Mobility pattern and network topology

In building upon the MADCAL algorithm we again utilise a circular mobility pattern around the static nodes in the network. 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 Fig. 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, there 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. This is illustrated in Fig. 2.

3.2 Overview of mobility aware duty cycling

This work builds upon the earlier MADCAL algorithm [12, 13] in utilising a predictable sink mobility pattern. Network parameters within a static node enable the calculation of the current sink position. Those being the sink starting position, speed and the time it has been travelling for. This calculation involves no expensive beaconing and

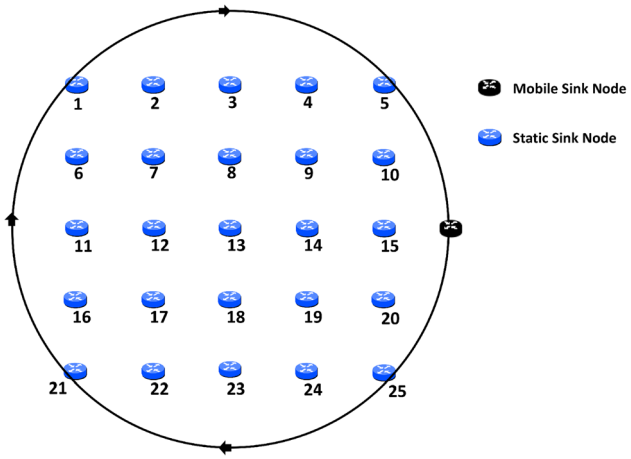


Fig. 1 Network topology - grid

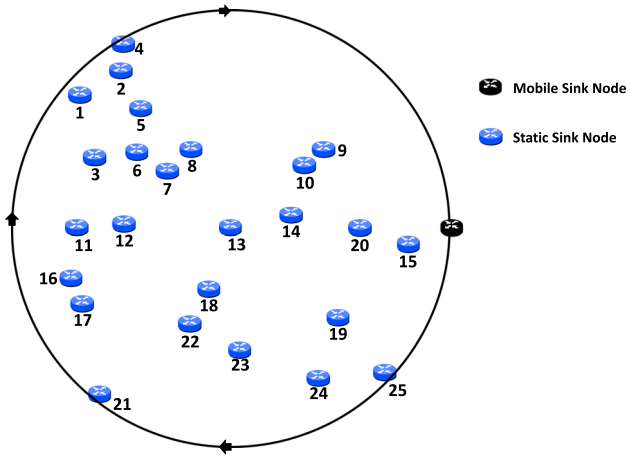


Fig. 2 Network topology - random

is made within each node, independently of others. A circular mobility pattern is utilised, with the sink travelling around the periphery of the network. Therefore, the *significant* nodes to be identified, are those which are one-hop from the path of the sink based on interference range.

Algorithm 1 Communication Threshold - MADCAL

procedure INITIALISATION
  set $sinkSpeed$
  $significantNode \leftarrow false$
  set $transDist$
  set $Circumference$
  set $firstSinkPos$
  set $firstSinkQuartile$
  set $distToCircle$
  if $distToCircle < transDist$ **then**
    $significantNode \leftarrow true$
  end if
  if $significantNode$ **then**
    set $circlePoint$
    set $nodeQuartile$
    set $distanceBetweenPoints$
    set $angleOfNode$
    $thresholdAfter \leftarrow true$
    $sinkThresholdAfter \leftarrow establishThreshold(sinkRadius, thresholdAfter)$
    $thresholdAfter \leftarrow false$
    $sinkThresholdBefore \leftarrow establishThreshold(sinkRadius, thresholdAfter)$
    set $thresholdDistance$
    set $beforeQuartile$
    set $thresholdOpposite$
  end if
end procedure
function ESTABLISHTHRESHOLD(radius, after)
  $nodeDist \leftarrow (radius - distToCircle)$
  $angleTemp \leftarrow \frac{(radius^2 + nodeDist^2 - transDist^2)}{(2 * radius * nodeDist)}$
  $angleRadians \leftarrow arccos(angleTemp)$
  $angle \leftarrow (angleRadians * (\frac{180}{PI}))$
  $factor \leftarrow \frac{distToCircle}{transDist}$
  if $sinkSpeed < 10$ **then**
    $factorCheck \leftarrow 0.5$
  else if $sinkSpeed < 20$ **then**
    $factorCheck \leftarrow 0.35$
  else if $sinkSpeed < 40$ **then**
    $factorCheck \leftarrow 0.25$
  end if
  if $factor < factorCheck$ **then**
    $factor \leftarrow factorCheck$
  end if
  $angle \leftarrow (angle * factor)$
  if after **then**

$threshAngleDegrees \leftarrow (angle + angleOfNode)$
  else

$threshAngleDegrees \leftarrow (angleOfNode - angle)$
  end if
  $threshAngleRadians \leftarrow \frac{threshAngleDegrees}{(180 * PI)}$
  $threshold.x \leftarrow circleCentre.x + (radius * cos(threshAngleRadians))$
  $threshold.y \leftarrow circleCentre.y + (radius * sin(threshAngleRadians))$
  return Coord $threshold$
end function

Mobility aware duty cycling involves two stages, the first to create a dynamic threshold between static node and mobile sink, as detailed in Algorithm 1. The point on the circular path the shortest distance from the significant node is named as the *circlePoint*. The maximum possible threshold may then be calculated based on this point and the node interference range, as seen in Fig. 3 from the point of view of Node 15. The size of this threshold is then controlled by a factor, in order to avoid extremes of size if the node is very close or far away from the sink path. Interference distance over distance to the path is applied to the size of the threshold, with sink speed then used to adjust further, ensuring the threshold size does increase or decrease too much. Smaller thresholds are found to perform better when the sink is faster, therefore the threshold is adjusted accordingly. Increasing in size as the sink speed decreases. The MADCAL algorithm only accounts for speeds of 2mps, 10mps, 20mps and 40mps. An adjustment the MADCaDPAL algorithm makes is to ensure this calculation is now completely dynamic for any speed in this range.

Secondly, the focus is on utilising the threshold created such that duty cycling within significant nodes may now be positively influenced. This is achieved within the SLEEP procedure of the MAC implementation, in particular at the point where the move from the SLEEP procedure to CCA takes place. If the sink node is calculated to not be in the node's threshold yet, then this move is delayed for the time it will take for the sink to reach the threshold start point. Still able to receive messages, but effectively asleep. This is detailed in Algorithm 2. This approach has already shown benefit of up to 15% in energy consumption, when compared to a standard duty cycling approach with CCA, preambles and check interval. With an additional benefit shown in frame delivery.

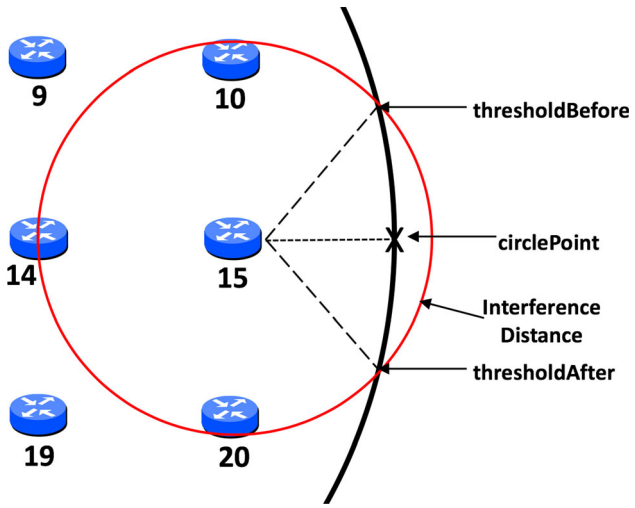


Fig. 3 Threshold Calculation

Algorithm 2 Threshold Interval - MADCAL

```
1: procedure SLEEP
2:     set checkInterval
3:     if significantNode then
4:         thresholdTime()
5:         if thresholdReached then
6:             interval ← checkInterval
7:         else
8:             interval ← timeToThreshold
9:         end if
10:    else
11:        interval ← checkInterval
12:    end if
13:    schedule CCA at time + interval
14: end procedure
15: function THRESHOLDTIME
16:    set sinkPos
17:    withinThreshold()
18:    if not thresholdreached then
19:        set arc to distance to
       sinkThresholdBefore
20:        set sinkQuartile
21:        timeToThreshold ← arc / sinkSpeed
22:    else
23:        timeToThreshold ← 0
24:    end if
25:    return timeToThreshold
26: end function
27: function WITHINTHRESHOLD
28:    if distance between sinkPos and
       thresholdAfter > thresholdDistance then
29:        thresholdReached ← false
30:    else if distance between sinkPos and
       thresholdBefore > thresholdDistance then
31:        thresholdReached ← false
32:    else
33:        thresholdReached ← true
34:    end if
35:    return thresholdReached
36: end function
```

4 The proposed MADCaDPAL algorithm

4.1 Problem statement

The dynamic threshold of communication between MSN and static significant node is shown to have benefit when considering the average energy consumption of all significant nodes. Once this threshold is determined it is utilised in determining when to move from SLEEP to CCA within a MAC layer implementation. However, on examining results further it can be observed that while average energy consumption is improved there remain energy consumption *spikes*. These being where certain significant nodes consume far more energy than others. These are not consistent across all tests and spike nodes may not even be the same across repeats of the same test with parameters unchanged. As such, despite the benefits of sink mobility it can be claimed that, to a certain degree, the hotspot issue remains as some nodes will still run out of energy faster than others. With the obvious negative resultant effect on network lifetime. As such, the first aim of the MADCaDPAL algorithm is to reduce these spikes in energy consumption, with an overall reduction in energy consumption also desirable. Again, without a detrimental effect on frame delivery to the sink.

4.2 Mobility aware approach to duty cycling and preambles

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, this has only been utilised to intercept the move from the SLEEP process to CCA [12, 13] at the MAC layer. As such, in examining this procedure, it could be observed that once the CCA process started, this would continue between the static node and sink beyond the end of the communication threshold. The result being that some nodes could then monopolise communication with the sink, whilst others must wait for the channel to clear, even if the sink is now within their threshold. This results in the aforementioned spikes of energy. Hence, in this study the aim is to *close* the communication threshold, in order that as soon as the sink passes the end of the threshold, communication is no longer possible.

As such, whereas the MADCaDPAL algorithm still influences the SLEEP function at the MAC layer, only moving to CCA when the sink is within the node's threshold, further factors are also take into account. The position of the MSN in relation to the threshold is now used to influence the CCA process as well as the sending of preambles. With these changes, we aim to further reduce energy consumption amongst significant nodes, while also improving frame delivery to the sink node. However, a more even distribution of energy consumption across significant nodes is of equal importance.

We utilise the grid and random network topologies as shown in Figs. 1 and 2. As can be seen in the test parameters in Table 1, the interference range of nodes is varied across four values. Within the grid topology this results in the same significant nodes each time, as can be seen in Fig. 4. 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 Fig. 5.

The lightweight carrier-sense multiple access (CSMA) MAC implementation in use can be seen in Fig. 6, with the first part of the original MADCAL algorithm shown to create the communication threshold, and MADCaDPAL

Table 1 Simulation Parameters

Parameters	Values
Number of nodes (static)	25
Grid topology size	$200m * 200m$
MSN path radius	150m
MSN start position	$x = 400m, y = 250m$
MSN speed (metres per second)	$2mps$, $10mps$, $20mps$, $40mps$
Simulation time	$942.47779607694s$
Interference distance * 4	$77.52m$, $69.13m$, 62.02, $55.94m$
Number of runs	5
Path-loss alpha * 4	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	$59400mWs$
Check interval	$0.01s$
Slot duration	$0.1s$

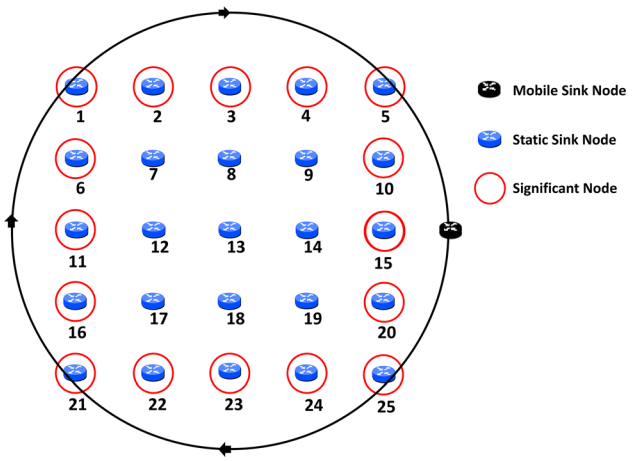


Fig. 4 Network topology - grid with significant nodes

highlighted to intercept SLEEP, CCA and preamble sending.

4.2.1 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 case, the optimized link state routing protocol (OLSR) [34] 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.

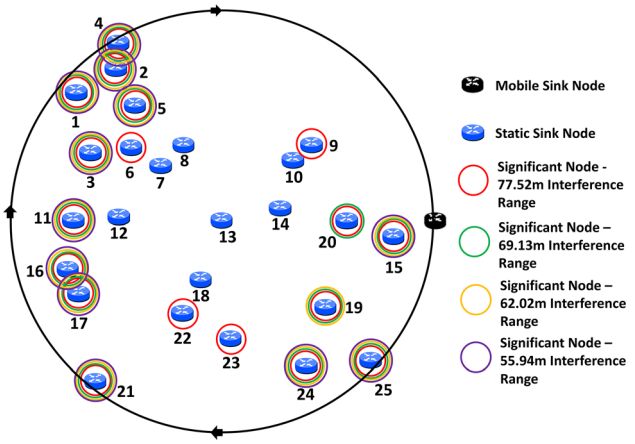


Fig. 5 Network topology - random with significant nodes

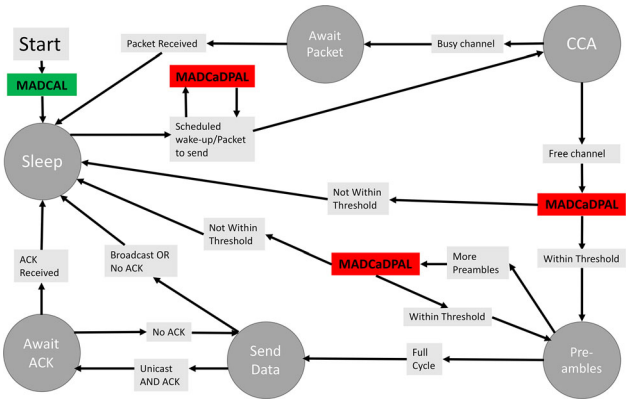


Fig. 6 Lightweight CSMA MAC Implementation with MADCaDPAL

4.3 New threshold calculation

An illustration of the development of a dynamic communication threshold between static significant node and MSN can be seen in Fig. 7, 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. When the MADCAL algorithm was developed, the initial threshold calculation was adjusted by a factor of the distance to the circle path divided by the node interference distance. This was then

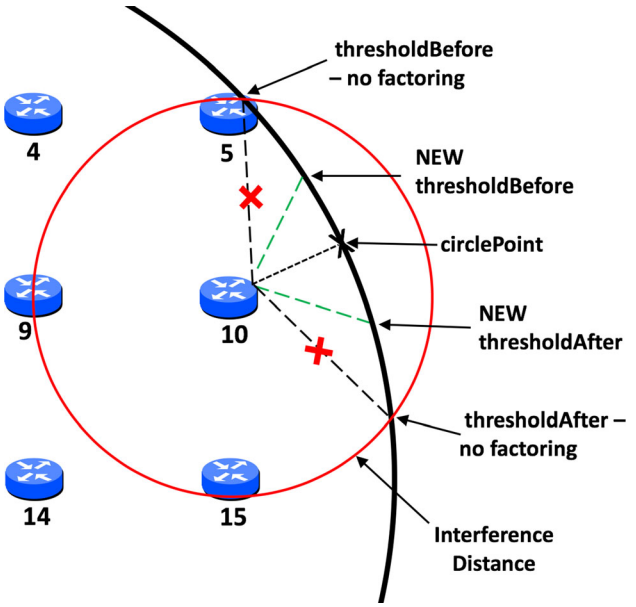


Fig. 7 Illustration of Threshold

adjusted based on sink speed, with faster speeds allowing for smaller thresholds. However, this was based around the specific speeds used, those being 2mps, 10mps, 20mps and 40mps. With MADCaDPAL, the calculation of this adjustment factor now becomes completely dynamic, with the only input required that of maximum and minimum sink speeds. In this case, 2mps and 40mps. Any speed within that range will have an adjustment factor calculated specifically. This is defined in Eq. (1). How this algorithm is applied to the building of the threshold can be seen in Algorithm 3, which details the initialisation and establishment of the communication threshold within each static, significant node.

$$factorCheck = \left(\frac{maxF - minF}{maxS - minS} \times speedDiff\right) \quad (1)$$

where *maxF* and *minF* represent the maximum and minimum factors. In the case of this study 0 and 0.5. *maxS* and *minS* represent the maximum and minimum sink node speeds. In the case of this study, 2mps and 40mps. With *speedDiff* the range between the highest and lowest speeds.

Algorithm 3 Communication threshold - MADCaDPAL

procedure INITIALISATION
  set $sinkSpeed$
  set $maxSpeed$
  set $minSpeed$
  set $maxFactor$
  set $minFactor$
  $speedDiff \leftarrow sinkSpeed - minSpeed$
end procedure
function ESTABLISHTHRESHOLD(radius, after)
  $nodeDist \leftarrow (radius - distToCircle)$
  $angleTemp \leftarrow \frac{(radius^2 + nodeDist^2 - interDist^2)}{(2 * radius * nodeDist)}$
  $angleRadians \leftarrow arccos(angleTemp)$
  $angle \leftarrow (angleRadians * (\frac{180}{PI})$
  $factor \leftarrow \frac{distToCircle}{interDist}$
  $factorCheck \leftarrow (\frac{(maxFactor - minFactor)}{(maxSpeed - minSpeed)} * speedDiff)$
  if $factor < factorCheck$ **then**
    $factor \leftarrow factorCheck$
  end if
  $angle \leftarrow (angle * factor)$
  if after **then**
    $threshAngleDegrees \leftarrow (angle + angleOfNode)$
  else
    $threshAngleDegrees \leftarrow (angleOfNode - angle)$
  end if
  $threshAngleRadians \leftarrow \frac{threshAngleDegrees}{(180 * PI)}$
  $threshold.x \leftarrow circleCentre.x + (radius * cos(threshAngleRadians))$
  $threshold.y \leftarrow circleCentre.y + (radius * sin(threshAngleRadians))$
  return Coord $threshold$
end function

Algorithm 3 Within the Initialisation procedure, the MSN speed is set from input (2, 10, 20, or 40 mps), the max and min speeds in this case are 40mps and 2mps. The maximum factor we use is 0.5, utilised for slower speeds. The minimum factor is 0 which means for faster speeds there is no limit to how much the threshold may be reduced by. The *speedDiff* variable sets a range value between highest and lowest speeds.

The *establishThreshold* function, as with MADCAL, establishes the coordinates of the communication

threshold, inputting the radius of the circular path and whether the coordinates to be calculated are before or after the *circlePoint*. The difference in MACDaDPAL is the new dynamic approach to determining the factor to apply to to threshold size. The distance from the centre of the circle to the static node is again calculated as *nodeDist*. As before, calculate the angle of the circle centre to the furthest point of communication, and the circle centre to the node,and to avoid excessively large thresholds, determine a factor by which this angle should be reduced based upon *distToCircle* divided by *interDist*.

A new dynamic factor check is based on the difference of the highest and lowest factor divided by the difference in maximum and minimum speeds, multiplied by the difference between sink speed and the minimum speed. This ensures that whatever the speed, the *factorCheck* is calculated uniquely for that speed. The faster the sink speed may be, the more the angle of the threshold may be reduced by. If the factor calculated is less than the factor check then the factor check value becomes the factor. The value of the angle of the threshold is multiplied by the factor in order to reduce it accordingly. The threshold is then calculated as in MADCAL.

4.4 Mobility aware duty cycling and dynamic preambling algorithm (MADCaDPAL)

The novel approach the MADCaDPAL algorithm takes is in how duty cycling at the MAC layer may be influenced by this threshold. Beyond existing work where only the SLEEP procedure is influenced [12, 13] in its move to CCA, this approach remains but is now extended to include a deeper examination of the actual CCA process and how the send of preambles may be influenced. As such, the move from the SLEEP function to CCA will still only occur once the MSN is calculated to be within the significant node's threshold. However, in order to ensure that communication between the sink and static node ends as soon as possible once the MSN has exited the node's threshold, further checks as now made. Once the CCA procedure is entered a further check to ensure the sink is within the threshold is made. However, this merely ensures that the small possibility that between the SLEEP procedure and CCA the sink has moved beyond the threshold is accounted for.

Of greater importance is a further check once a clear channel is confirmed and the sending of preambles commences. As such, the sending of preambles will be interrupted as soon as the MSN leaves the threshold, with any data in the cache of the static node stored until the next pass of the sink. Once interrupted, both the CCA and preamble sending procedures send the node to sleep for one slot duration. This is in line with the existing approach to sending the node to sleep within the MAC protocol. This ensures the SLEEP procedure will then again assume responsibility for calculating the position of the sink node.

$$pos.x = start.x + radius \times \cos\left(\left(\frac{startAngle}{180}\right) \times PI\right) + \left(\frac{speed}{radius}\right) \times simTime \quad (2)$$

$$pos.y = start.y + radius \times \sin\left(\left(\frac{startAngle}{180}\right) \times PI\right) + \left(\frac{speed}{radius}\right) \times simTime \quad (3)$$

This can be seen in Algorithm 4. It should be noted that the sending of data is not interrupted. Once this commences it is allowed to finish, wherever the sink may be located.

The control flow between Algorithms 3 and 4 can be seen in Fig. 8.

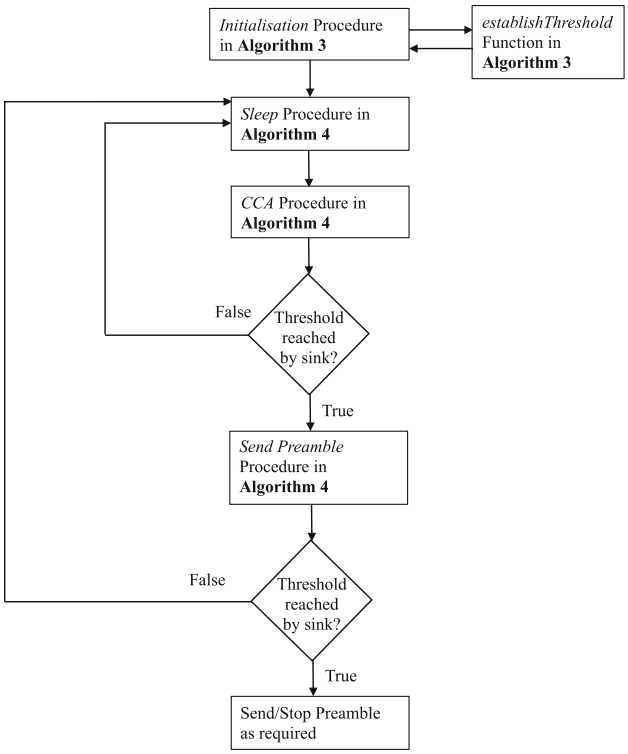


Fig. 8 MADCaDPAL Control Flow

Algorithm 4 Threshold Interval - MADCaDPAL

```
1: procedure SLEEP
2:     set checkInterval from input
3:     if significantNode then
4:         thresholdTime()
5:         if thresholdReached then
6:             interval ← checkInterval
7:         else
8:             interval ← timeToThreshold
9:         end if
10:    else
11:        interval ← checkInterval
12:    end if
13:    schedule CCA at simTime + interval
14: end procedure
15: procedure CCA
16:    if macQueue > 0 then
17:        withinThreshold()
18:        if thresholdReached then
19:            macState ← SEND PREAMBLE
20:            schedule STOP PREAMBLES at simTime +
    slotDuration
21:        else
22:            macState ← SLEEP
23:            schedule WAKEUP at simTime + slotDuration
24:        end if
25:    else
26:        macState ← SLEEP
27:        schedule WAKEUP at simTime + slotDuration
28:    end if
29: end procedure
30: procedure SEND PREAMBLE
31:    withinThreshold()
32:    if thresholdReached then
33:        if SEND PREAMBLE then
34:            sendPreamble()
35:            macState ← SEND PREAMBLE
36:            schedule SEND PREAMBLE at simTime +
    (0.5 ∗ checkInterval)
37:        else if STOP PREAMBLES then
38:            cancel SEND PREAMBLE
39:            macState ← SEND DATA #Preambles over,
    send data
40:        end if
41:    else
42:        cancel SEND PREAMBLE/STOP PREAMBLES
43:        macState ← SLEEP
44:        schedule WAKEUP at simTime + slotDuration
45:    end if
46: end procedure
```

Algorithm 4

In the Sleep procedure, the default sleep interval is set as *checkInterval*. Now, only applying to significant nodes; the *thresholdTime* function is called to establish the time it will take for the sink to reach the threshold. If the threshold has been reached, the interval to wake-up and move to CCA reverts to the *checkInterval* otherwise, it is set to the time it will take for the sink to reach the threshold. If this is not a significant node the wake-up schedule reverts to the *checkInterval* as normal. This ends the Sleep procedure.

The *thresholdTime* function is designed to establish how long it will take the sink to reach the threshold. The current sink position is calculated as *sinkPos* based on simulation time and the initial sink start position as in Eqs. (2) and (3).

Where *pos.x, pos.y* denotes the calculated sinkPos coordinates; *start.x, start.y* denotes the sink start position; *startAngle* denotes the starting angle of the sink, in this case 0; *speed* denotes the sink speed; *simTime* denotes the current time the simulation has been running.

The *withinThreshold* function is then called to establish if the sink is already within the communication threshold or not. If the threshold has not been reached yet, *arc* is set to the distance the sink must travel to reach the *sinkThresholdBefore* coordinates. Using the distance calculated between *sinkPos* and *sinkThresholdBefore*, the length of the *arc* is calculated as illustrated in Eq. (4):

$$arc = \left(\frac{180 - \left(\left(\arccos\left(\frac{\left(\frac{D}{2}\right)}{R} \right) \times \left(\frac{180}{PI}\right) \right) * 2 \right)}{360} \right) \times C \qquad (4)$$

Where *D* denotes the distance between *sinkPos* and *sinkThresholdBefore*; *C* denotes the circumference of the sink circular path and *R* it's radius. The current quartile in which the sink resides is established and the time left to reach the threshold is calculated as the size of the arc in metres divided by the speed of the MSN in mps. However, if the threshold has been reached, then the time left to reach the threshold is set as zero. The *thresholdTime* function returns the *timeToThreshold*.

The *withinThreshold* function will establish the position of the sink node in relation to the static node communication threshold. The distance between the current sink position and the *thresholdAfter* coordinates is established as per Equation ??. If this is greater than the size of the entire threshold then the threshold has not been reached and *thresholdReached* is set to false. However, if this is not true and the distance between the current sink position and the *thresholdAfter* coordinates is less than the size of the entire threshold, now establish if the sink is before or after the threshold. If the distance between the sink position and the *thresholdBefore* coordinates is greater than the *thresholdDistance* then the sink must be beyond the threshold and therefore *thresholdReached* is set to false. If the sink is also within *thresholdDistance* of *thresholdBefore* however, this means the sink is within the threshold. Therefore *thresholdReached* is set to true. With this, *thresholdReached* is returned and the *withinThreshold* function ends.

Within the CCA procedure, first there is a check to establish if there there is something in the queue to send. If so, the *withinThreshold* function is called to establish if the MSN is within the threshold currently. If the threshold has been reached move to sending preambles and schedule the stopping of preambles after a slot duration as normal. Otherwise, return to the Sleep procedure and awaken after a slot duration. If there is nothing in the queue, return to the Sleep procedure and awaken after a slot duration as normal. This ends the CCA procedure.

Within the Send Preamble procedure, the *withinThreshold* function is called to establish if the MSN is within the threshold currently. If the threshold has been reached then the process of sending preambles continues as normal unless Stop Preambles is reached. If the threshold has not been reached then the process of sending or stopping preambles is cancelled and the MAC protocol returns to the Sleep procedure. The node is scheduled to awaken after a slot duration. This ends the Send Preamble procedure.

5 Evaluation and results

5.1 Network parameters

Among both network implementations the assumption is made that static nodes will retain their positions throughout. Each static node implements MADCaDPAL 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. All other parameters can be found in Table 1.

The simulation time utilised guarantees the network is traversed an exact number of times by the MSN, this based on the network size and thus, the size of the circle path circumference. As such, at 40mps, the network shall be circled exactly 40 times, reducing down to twice for 2mps. This ensures each node is passed by the sink the same number of times, otherwise results would not be accurate for individual nodes.

The interference distance of each node is calculated using Eq. (5) [35]:

$$interferenceDistance = \left(\frac{\left(\frac{SoL}{Freq}\right)^2 \times Power}{16 \times PI^2 \times 10^{\frac{SAT}{10}}} \right)^{\frac{1.0}{Alpha}} \quad (5)$$

Where *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.

5.2 Simulation environment and parameters

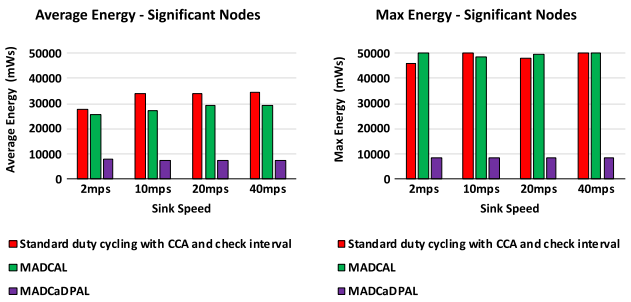
The grid and random topology networks are built using the OMNeT++ [36] simulation framework. In this case MiXiM [37] is utilised to build the actual network layout, whilst inetmanet [38] is used to define physical, MAC, network and transport layer parameters.

Results show the effect of an MSN on the significant nodes, these highlighted in Figs. 3 and 4. Each test scenario has been conducted using three different MAC implementations to establish the effect of a MSN at four different speeds and with four different interference ranges across the network. Firstly, the existing lightweight MAC implementation using standard duty cycling techniques with no allowance for sink mobility is shown. Secondly, results are shown using the MADCAL [12, 13] algorithm. Finally, we present results where MADCaDPAL is in use.

5.3 Results - grid topology

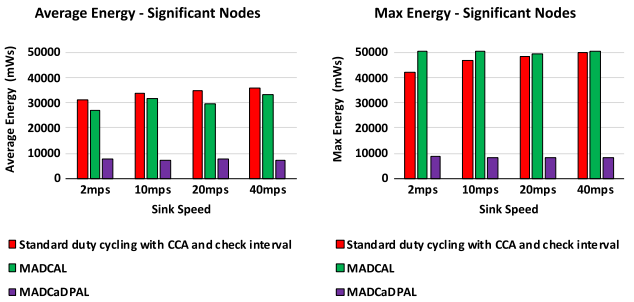
Figures 9, 10, 11, 12 show the average energy consumption of all significant nodes. This is presented alongside the maximum energy consumption of significant nodes, this being the single significant node to consume the most energy. In examining this comparison we may observe the difference between the overall average and the greatest spike in energy consumption in a single node.

In Fig. 9a the initial energy saving of up to 15% when MADCAL is in use can be seen versus the original MAC implementation. However, when MADCaDPAL is in use the resultant reduction in energy consumption is significant. Compared to original results with no allowance made for use of an MSN, we can observe up to 79% reduction in energy consumed. What can also be seen in Fig. 9b is that in the original MAC implementation and again when



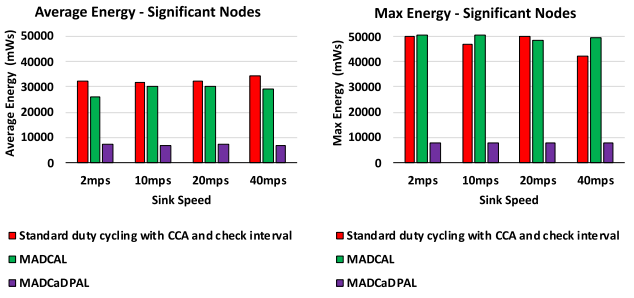
(a) Average Energy **(b)** Max Energy

Fig. 9 Average and maximum energy consumption of significant nodes (mWs), Grid topology. Transmission range 77.52m. MADCaDPAL



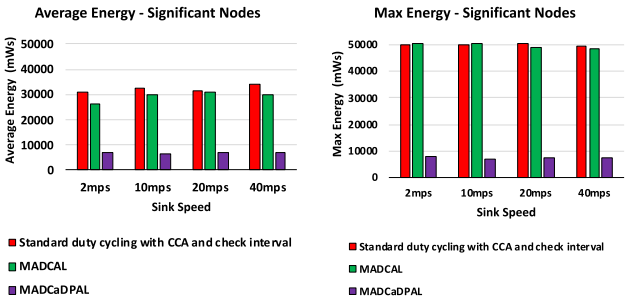
(a) Average Energy **(b)** Max Energy

Fig. 10 Average and maximum energy consumption of significant nodes (mWs), Grid topology. Transmission range 69.13m. MADCaDPAL



(a) Average Energy **(b)** Max Energy

Fig. 11 Average and maximum energy consumption of significant nodes (mWs), Grid topology. Transmission range 62.02m. MADCaDPAL



(a) Average Energy **(b)** Max Energy

Fig. 12 Average and maximum energy consumption of significant nodes (mWs), Grid topology. Transmission range 55.94m. MADCaDPAL

MADCAL is in use, that spikes in energy were considerable. Now with MADCaDPAL in use, the highest single node is only slightly above average on this particular scale.

Even though the interference range is reduced, a similar pattern of results can be seen in Fig. 10. Again there is a significant reduction in average energy consumption as seen in Fig. 10a, as well as with the maximum single node energy once MADCaDPAL is in use, as seen in Fig. 10b.

It can be seen in Fig. 11 that as the interference range becomes smaller, as do the benefits achieved by the MADCAL algorithm. However, with MADCaDPAL in use, the pattern of significant improvement in average energy as seen in Fig. 11a and maximum energy as seen in Fig. 11b, remains.

Even at the smallest interference range used as in Fig. 12, this pattern of improvements with the MADCaDPAL algorithm in use continues. Originally, as the interference range constricted to not much more then then distance between nodes, it became more difficult for the MADCAL algorithm to make improvements. Although they are still there to be seen. This, however, does not affect the MADCaDPAL algorithm. In ensuring the threshold is closed effectively the result is a reduction in energy consumption of over 80% amongst the significant nodes as seen in Fig. 12a. With energy consumed more evenly as the maximum figures show in Fig. 12b.

However, a benefit such as this would be less significant if it has a detrimental effect on other factors in the network. In Figs. 13,14, 15, 16 the number of MAC layer frames received by the MSN during each simulation can be seen.

In Fig. 13 and 14 it can be seen that as well as a significant improvement in energy consumption, the MADCaDPAL algorithm has also improved frame delivery in all but one scenario. With a 77.52m interference range and sink speed of 10mps frame reception decreases by around 20% from the MADCAL algorithm, although this is still an improvement on the original MAC implementation. However, in all other scenarios there is an improvement over both no change and the MADCAL algorithm. Significant improvement can be seen at 20mps, despite this being highlighted as of difficulty to achieve by the MADCAL algorithm.

Figures 15 and 16 show that even as interference range again reduces, sink frame reception improves with MADCaDPAL in use. Across all tests a speed of 10mps proves to be problematic in this regard. However, considering the

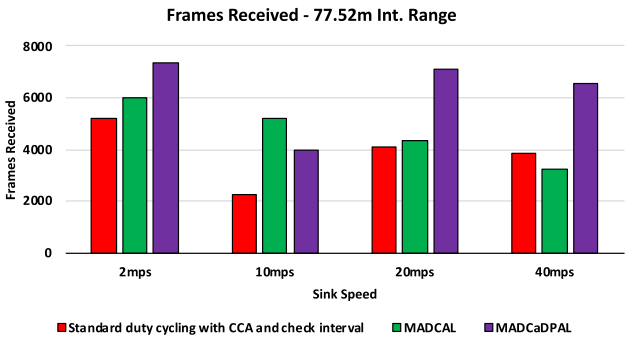


Fig. 13 Sink frame reception. Transmission range 77.52m, Grid topology. MADCaDPAL

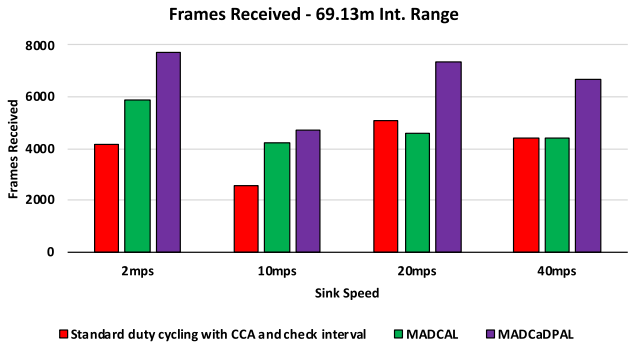


Fig. 14 Sink frame reception. Transmission range 69.13m, Grid topology. MADCaDPAL

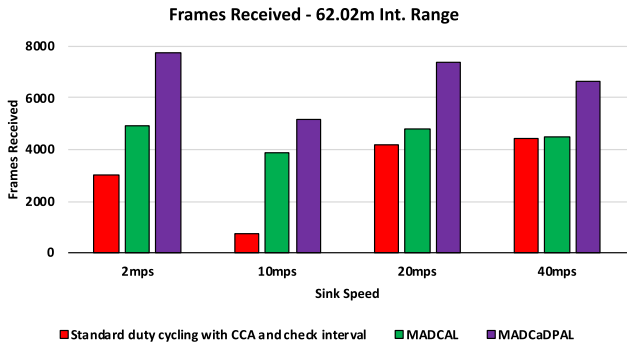


Fig. 15 Sink frame reception. Transmission range 62.02m, Grid topology. MADCaDPAL

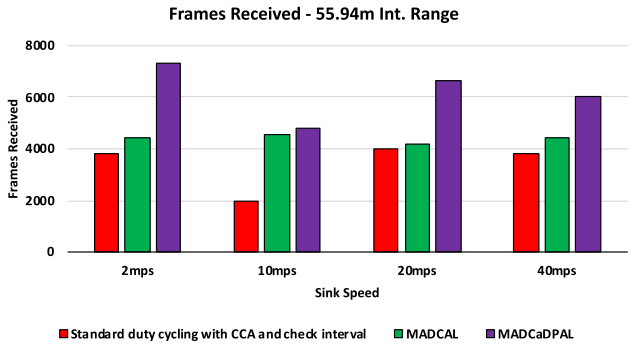


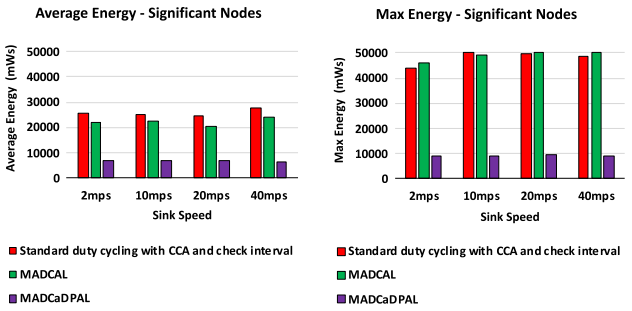
Fig. 16 Sink frame reception. Transmission range 55.94m, Grid topology. MADCaDPAL

benefit in energy consumption, sink reception even remaining at similar levels could be seen as acceptable.

5.4 Results - random topology

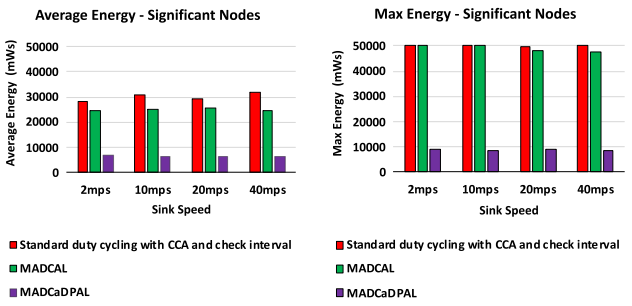
Figures 17, 18, 19, 20 show the average energy consumption of all significant nodes, again presented alongside the maximum energy consumption of significant nodes.

In Fig. 17 results are very much as with the grid topology, although it can be seen that the spike in energy consumption when standard duty cycling and MADCAL



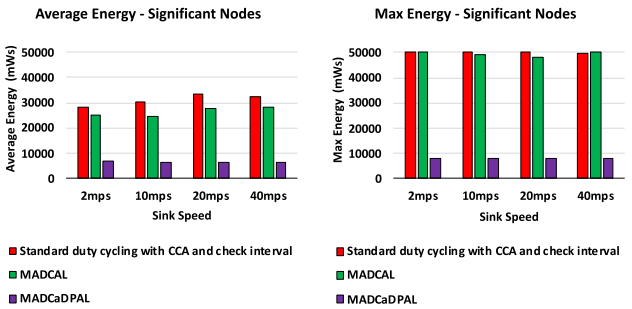
(a) Average Energy **(b)** Max Energy

Fig. 17 Average and maximum energy consumption of significant nodes (mWs). Transmission range 77.52m, Random topology. MADCaDPAL



(a) Average Energy **(b)** Max Energy

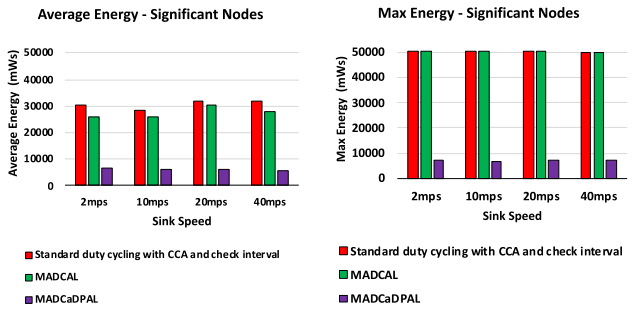
Fig. 18 Average and maximum energy consumption of significant nodes (mWs). Transmission range 69.13m, Random topology. MADCaDPAL



(a) Average Energy **(b)** Max Energy

Fig. 19 Average and maximum energy consumption of significant nodes (mWs). Transmission range 62.02m, Random topology. MADCaDPAL

are in use is even more distinct. Again energy consumption is reduced dramatically. What is off interest is that although the benefits are great, there is still a slight difference between average energy with MADCaDPAL in use and the maximum energy consumed. Showing that even though the scale of energy consumption is reduced and network lifetime increased, there is still an opportunity to further even energy consumption across significant nodes.



(a) Average Energy **(b)** Max Energy

Fig. 20 Average and maximum energy consumption of significant nodes (mWs). Transmission range 55.94m, Random topology. MADCaDPAL

This pattern repeats in Fig. 18, even as interference range reduces. MADCaDPAL can be seen to again significantly reduce energy consumption.

Even at the smallest interference ranges used as in Fig. 19 and 20, this pattern of improvements with the MADCaDPAL algorithm in use continues. The improvement in reducing energy spikes is now very clear. As a result, improving network lifetime before the first node expires by a factor of up to 7.

In Fig. 21, 22, 23, 24 the number of MAC layer frames received by the MSN during each simulation can be observed.

In Fig. 21 it can be seen that with the largest interference range in use there is generally an improvement in frame delivery to the sink. As the interference range lessens in Fig. 22, frame delivery generally stays the same aside from when the sink is moving at 10mps. Delivery now returns to the poor level of when the standard duty cycling was in use.

Figure 23 shows that in the lower interference ranges again frame delivery remains at similar levels. Again though, at the smallest interference range as seen in Fig. 24, there is a drop when the sink moves at 10mps.

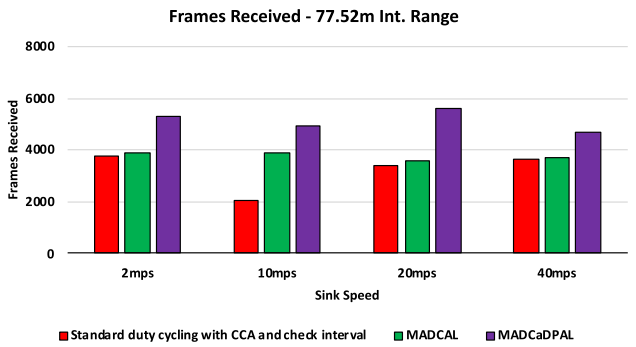


Fig. 21 Sink frame reception. Transmission range 77.52m, Random topology. MADCaDPAL

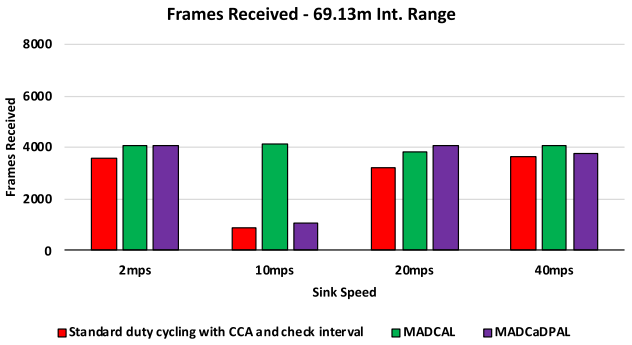


Fig. 22 Sink frame reception. Transmission range 69.13 m, Random topology. MADCaDPAL

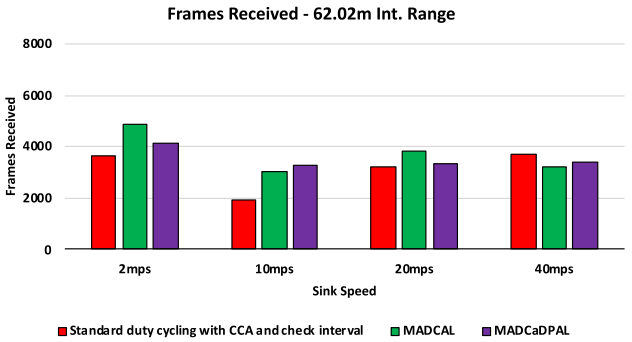


Fig. 23 Sink frame reception. Transmission range 62.02m, Random topology. MADCaDPAL

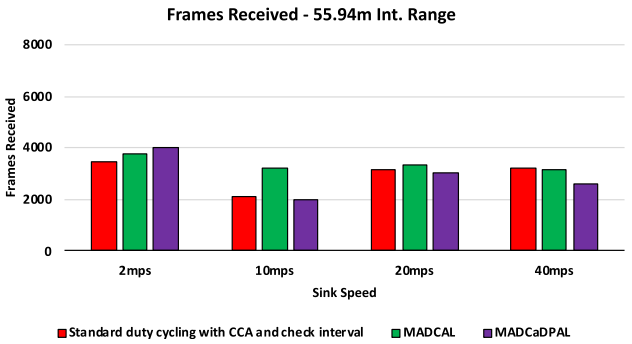


Fig. 24 Sink frame reception. Transmission range 55.94m, Random topology. MADCaDPAL

6 Conclusion

In this study we propose MADCaDPAL, for use with MSNs. This algorithm adds no network overhead, utilising predictable sink mobility and self-knowledge of location amongst static nodes to improve energy consumption and network performance when no allowance is made for a MSN, or when the MADCAL [12, 13] algorithm is in use. In utilising determination of the sink node position in relation to the threshold, within the CCA and preambling procedures of the lightweight CSMA MAC protocol used, and linking this to the already successful use of this

approach in the SLEEP procedure, improvements of between 70% and 80% are observed. As importantly from a network lifetime standpoint however, the issue of excessive energy spikes is somewhat negated. With the maximum energy consumption across significant nodes now much closer to the average consumption figures. This in contrast to the results for MADCAL.

Improvements are also shown in frame delivery when a controlled, grid network formation is used. When we utilise a more *strained*, random topology frame delivery is generally the same or improved aside from two scenarios. As such, whereas MADCAL represented a crucial first study in utilising predictable sink mobility to positively influence network behaviour, the extended MADCaDPAL algorithm demonstrates how this functionality may be used to have a significant effect on network behaviour. Whilst the initial aim was to reduce network spikes, the subsequent benefit in energy consumption is so great as to represent a major step forward in this area.

The MADCaDPAL algorithm shows significant reduction in average energy consumption whilst improving frame delivery. However, despite the improvement in energy consumption and bringing the maximum energy consumption more in line with the overall average, some differences remain. For future work we seek to develop a completely dynamic approach, altering the communication threshold as time passes to react to spikes in energy consumption and subsequently, improve network lifetime.

References

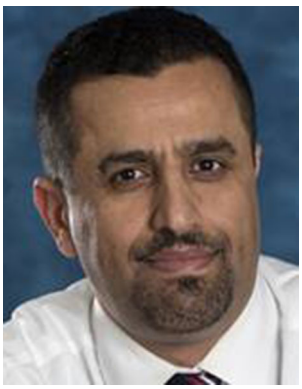
1. Ali, Salman, Ashraf, Adnan, Qaisar, Saad Bin, Afridi, Muhammad Kamran, Saeed, Husnain, Rashid, Sidra, et al. (2018). SimpliMote: A wireless sensor network monitoring platform for oil and gas pipelines. *IEEE Systems Journal, 12*(1), 778–789. https://doi.org/10.1109/JSYST.2016.2597171.
2. Ahmed, Adnan, Bakar, Kamalrulnizam Abu, Channa, Muhammad Ibrahim, Khan, Abdul Waheed, & Haseeb, Khalid. (2017). Energy-aware and secure routing with trust for disaster response wireless sensor network. *Peer-to-Peer Networking and Applications, 10*(1), 216–237. https://doi.org/10.1007/s12083-015-0421-4.
3. Uddin, Mohammad Ammad, Mansour, Ali, Le Jeune, Denis, Ayaz, Mohammad, El Hadi, M., & Aggoune., (2018). Uav-assisted dynamic clustering of wireless sensor networks for crop health monitoring. *Sensors (Switzerland), 18*(2). https://doi.org/10.3390/s18020555. ISSN 14248220.
4. Chen Lin, & Bian, Kaigui. (2016). Neighbor discovery in mobile sensing applications: A comprehensive survey. Ad Hoc Networks. ISSN 15708705. https://doi.org/10.1016/j.adhoc.2016.05.005.
5. IEEE 802.15.4, 2016. URL http://www.ieee802.org/15/pub/TG4.html.
6. Montenegro, G., Kushalnagar, N., Hui, J., & Culler, D. (2007). RFC4944: Transmission of IPv6 Packets over IEEE 802.15.4 Networks.
7. Cano, C., Bellalta, B., Sfairopoulou, A., & Oliver, M. (2011). Low energy operation in WSNs: A survey of preamble sampling MAC protocols. *Computer Networks, 55*(15), 3351–3363. https://doi.org/10.1016/j.comnet.2011.06.022.
8. Gao, Honghao, Yueshen, Xu., Yin, Yuyu, Zhang, Weipeng, Li, Rui, & Wang, Xinheng. (2020). Context-aware QoS prediction with neural collaborative filtering for internet-of-things services. *IEEE Internet of Things Journal, 7*(5), 4532–4542. https://doi.org/10.1109/JIOT.2019.2956827.
9. Zhou, Ao, Wang, Shangguang, Wan, Shaohua, & Qi, Lianyong. (2020). Lmm: latency-aware micro-service mashup in mobile edge computing environment. Neural Computing and Applications, pages 1–15. https://doi.org/10.1007/s00521-019-04693-w.
10. Wan, Shaohua, Gu, Renhao, Umer, Tariq, Salah, Khaled, & Xu, Xiaolong. (2020). Toward Offloading Internet of Vehicles Applications in 5G Networks. IEEE Transactions on Intelligent Transportation Systems, pages 1–9. ISSN 1524-9050. https://doi.org/10.1109/TITS.2020.3017596. URL https://ieeexplore.ieee.org/document/9184262/.
11. Tang, Xiaofeng, & Xie, Li. (2017). 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. https://doi.org/10.4236/ijcns.2017.105B023.
12. Thomson, Craig, Wadhaj, Isam, Tan, Zhiyuan, Al-dubai, Ahmed. (2019). 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. https://doi.org/10.1109/SmartIoT.2019.00037.
13. Thomson, Craig, Wadhaj, Isam, Tan, Zhiyuan, & Al-Dubai, Ahmed. (2019). Mobility Aware Duty Cycling Algorithm (MADCAL) a dynamic communication threshold for mobile sink in wireless sensor network. *Sensors, 19*(22), 4930. https://doi.org/10.3390/s19224930.
14. Tang, Tao, Hong, Tao, Hong, Haohui, Ji, Senyuan, Mumtaz, Shahid, & Cheriet, Mohamed. (2019). An improved UAV-PHD filter-based trajectory tracking algorithm for multi-UAVs in future 5G IoT scenarios. *Electronics, 8*(10), 1188. https://doi.org/10.3390/electronics8101188.
15. Gao, Honghao, Liu, Can, Li, Youhuizi, Yang, Xiaoxian. (2020). V2VR: Reliable Hybrid-Network-Oriented V2V Data Transmission and Routing Considering RSUs and Connectivity Probability. IEEE Transactions on Intelligent Transportation Systems.https://doi.org/10.1109/TITS.2020.2983835. ISSN 15580016.
16. Jamalabdollahi, M., & Zekavat, S. A. R. (2015). Joint neighbor discovery and time of arrival estimation in wireless sensor networks via OFDMA. *IEEE Sensors Journal, 15*(10), 5821–5833. https://doi.org/10.1109/JSEN.2015.2449079.
17. Wang, Keyu, Mao, Xufei, & Liu, Yunhao. (2015). BlindDate: A neighbor discovery protocol. *IEEE Transactions on Parallel and*

Distributed Systems, 26(4), 949–959. https://doi.org/10.1109/TPDS.2014.2316159.
18. Pozza, R., Nati, M., Georgoulas, S., Moessner, K., & Gluhak, A. (2015). Neighbor discovery for opportunistic networking in internet of things scenarios: A survey. *IEEE Access, 3,* 1101–1131. https://doi.org/10.1109/ACCESS.2015.2457031.
19. Yang, Shusen, Adeel, Usman, Tahir, Yad, & Mccann, Julie A. (2016). Practical opportunistic data collection in wireless sensor networks with mobile sinks. *IEEE Transactions on Mobile Computing, 99,* 1–14. https://doi.org/10.1109/TMC.2016.2595574.
20. Papadopoulos, Georgios Z., Kotsiou, Vasileios, Gallais, Antoine, Chatzimisios, Periklis, & Noel, Thomas. (2016). Low-power neighbor discovery for mobility-aware wireless sensor networks. Ad Hoc Networks. https://doi.org/10.1016/j.adhoc.2016.05.011. ISSN 15708705.
21. Hess, Andrea, Hyytia, Esa, & Ott, Jorg. (2014). Efficient neighbor discovery in mobile opportunistic networking using mobility awareness. In 2014 6th International Conference on Communication Systems and Networks, COMSNETS 2014. https://doi.org/10.1109/COMSNETS.2014.6734890. ISBN 9781479936359.
22. Thomson, Craig, Wadhaj, Isam, Al-dubai, Ahmed, Zhiyuan Tan, A., Cycling, New Mobility Aware Duty, Algorithm, Dynamic Preambling, et al. (2020). *IEEE 6th World Forum on The Internet of Things (WF-IoT 2020), New Orleans, USA, 2020*. In press: Institute of Electrical and Electronics Engineers Inc.
23. Tunca, Can, Sinan Isik, M., Donmez, Yunus, & Ersoy, Cem. (2014). Distributed mobile sink routing for wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials, 16*(2), 877–897. https://doi.org/10.1109/SURV.2013.100113.00293.
24. Luo, Haiyun, Ye, Fan, Cheng, Jerry, Songwu, Lu., & Zhang, Lixia. (2005). TTDD: Two-tier data dissemination in large-scale wireless sensor networks. *Wireless Networks, 11,* 161–175. https://doi.org/10.1007/s11276-004-4753-x.
25. Kweon, Kisuk, Ghim, Hojin, Hong, Jaeyoung, Yoon, Hyunsoo, Routing, Grid-Based Energy-Efficient, & from multiple sources to multiple mobile sinks in wireless sensor networks. In,. (2009). 4th International Symposium on Wireless and Pervasive Computing, ISWPC 2009, 2009. https://doi.org/10.1109/ISWPC.2009.4800585. ISBN, 9781424429660.
26. Lin, Ching-Ju, Chou, Po-Lin, & Chou, Cheng-Fu. (2006). HCDD: Hierarchical clusterbased data dissemination in wireless sensor networks with mobile sink. In Proceeding of the 2006 international conference on Communications and mobile computing - IWCMC '06, page 1189, New York, New York, USA. ACM Press. https://doi.org/10.1145/1143549.1143787. ISBN 1595933069.
27. Yuan Xun-Xin, Zhang, Rui-Hua (2011). An Energy-Efficient Mobile Sink Routing Algorithm for Wireless Sensor Networks. In 2011 7th International Conference on Wireless Communications, Networking and Mobile Computing, pages 1–4. IEEE, 09. https://doi.org/10.1109/wicom.2011.6040374. ISBN 978-1-4244-6250-6.
28. Nazir, Babar, Hasbullah, Halabi. (2010). Mobile Sink based Routing Protocol (MSRP) for Prolonging Network Lifetime in Clustered Wireless Sensor Network. In ICCAIE 2010 - 2010 International Conference on Computer Applications and Industrial Electronics, pages 624–629. https://doi.org/10.1109/ICCAIE.2010.5735010. ISBN 9781424490554.
29. Yarinezhad, Ramin, & Sarabi, Amir. (2018). 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. https://doi.org/10.1016/j.aeue.2017.11.026.
30. Redhu, Surender, & Hegde, Rajesh M. (2019). Network lifetime improvement using landmark-assisted mobile sink scheduling for cyber-physical system applications. *Ad Hoc Networks, 87,* 37–48. https://doi.org/10.1016/j.adhoc.2018.10.029.
31. Kumar Nitesh, Md., Azharuddin, Md., & Jana, Prasanta K. (2018). A novel approach for designing delay efficient path for mobile sink in wireless sensor networks. *Wireless Networks, 24*(7), 2337–2356. https://doi.org/10.1007/s11276-017-1477-2.
32. Vahabi, Shahrokh, Eslaminejad, Mohammadreza, & Dashti, Seyed Ebrahim. (2019). Integration of geographic and hierarchical routing protocols for energy saving in wireless sensor networks with mobile sink. *Wireless Networks, 25*(5), 2953–2961. https://doi.org/10.1007/s11276-019-02015-5.
33. Khan, Abdul Waheed, Bangash, Javed Iqbal, Ahmed, Adnan, & Abdullah, Abdul Hanan. (2019). QDVGDD: Query-driven virtual grid based data dissemination for wireless sensor networks using single mobile sink. *Wireless Networks, 25*(1), 241–253. https://doi.org/10.1007/s11276-017-1552-8.
34. Clausen, T. (2003). RFC 3626 - Optimized Link State Routing Protocol - OLSR. Technical report
35. IntRange, (2019). URL https://github.com/inetmanet/inetmanet/blob/master/src/underTest/wpan/linklayer/ ieee802154/phyLayer/Ieee802154Phy.cc.
36. OMNeT++ Discrete Event Simulator, (2015). URL https://omnetpp.org/.
37. MiXiM. URL http://mixim.sourceforge.net/.
38. Inetmanet Installation Guide. URL http://omnet-manual.com/inetmanet-installation/.

Craig Thomson is a lecturer at Edinburgh Napier University, United Kingdom. He received B.Eng. (First Class Honours with University Medal) in 2016 and passed his Ph.D. degree in applied computing in 2020, both from Edinburgh Napier University, and is an Associate Fellow of the Higher Education Academy (AFHEA). His research interests include Mobile Wireless Sensor Networks, Internet of Things and Low Power and Lossy Networks. He has published in leading journals and has been awarded at international conferences for his research.

Isam Wadhaj received B.Eng. degree in Computer Networks and Distributes Systems in 2010 and M.Sc. degree in Advanced Networking in 2013 from Edinburgh Napier University, United Kingdom. He is currently working as a Lecturer in the School of Computing at the Edinburgh Napier University (ENU). His research interests include Wireless Sensors Networks, Internet of Things, Security of IoT and Low Power and Lossy Networks.

Zhiyuan Tan received the Ph.D. degree in computer systems from the University of Technology Sydney, Ultimo, NSW, Australia, in 2014. He was a Postdoctoral Researcher of cybersecurity with the University of Twente, The Netherlands, from 2014 to 2016, and a Research Associate with the University of Technology Sydney, in 2014. He is currently a Lecturer of cybersecurity with the School of Computing, Edinburgh Napier University, U.K. His research interests include cybersecurity, machine learning, cognitive computing. He is an IEEE, ACM, EAI and BCS member, and was a recipient of the several academic awards and recognitions. Dr Tan is an Academic Editor of Security and Communication Networks and a Guest Editor of Special Issues for the ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), Journal of Information Security and Applications, Computers and Electrical Engineering and IEEE Access, etc. Dr Tan has also chaired and co-chaired 18 international conferences/workshops.



Ahmed Al-Dubai (Senior Member, IEEE) received the Ph.D. degree in computing from the University of Glasgow, Glasgow, U.K., in 2004. In 2004, he joined the University of West London, London, U.K. In 2005, he joined Edinburgh Napier University, Edinburgh, U.K., where he became a Professor and the Programme Leader of the Post-Graduate Research degrees with the School of Computing. He is currently the Head of the Networks Research Group. He has been published in world leading journals and in prestigious international conferences. He has also been involved with research in the area of group communication algorithms, smart spaces, and high-performance networks. He is a fellow of the Higher Academy, U.K. He was a recipient of the several academic awards and recognitions, and a member of several editorial boards of scholarly journals. He has served as a Guest Editor for more than 20 special issues in scholarly journals and chaired and co-chaired more than 30 international conferences/workshops.