Dependability of Wireless Sensor Networks

Mark Louis Fairbairn

PhD

University of York

Computer Science

September 2014

Abstract

As wireless sensor networks (WSNs) are becoming ever more prevalent, the runtime characteristics of these networks are becoming an increasing issue. Commonly, external sources of interference make WSNs behave in a different manner to that expected from within simplistic simulations, resulting in the need to use additional systems which monitor the state of the network. Despite dependability of WSNs being an increasingly important issue, there are still only a limited number of works within this specific field, with the majority of works focusing on ensuring that specific devices are operational, not the application as a whole. This work instead aims to look at the dependability of WSNs from an application-centric view, taking into account the possible ways in which the application may fail and using the application's requirements to focus on assuring dependability.

Contents

List of Figures

List of Tables

Acknowledgements

I wish to thank my supervisor, Iain Bate, for his guidance and feedback during the PhD. I would like to thank my family and friends for their support and encouragement over the years. Finally I would like to thank all those who gave up their time to help with proof-reading throughout the PhD.

Declaration

I declare that the research described in this thesis is original work, which I undertook at the University of York during 2010 - 2014. This work has not been previously presented for an award at this or any other university. Except where stated, all of the work contained within this thesis represents the original contribution of the author.

Some of the material in Chapters 3 and 4 has been previously published. Where material was published jointly with collaborators, the author of this thesis is responsible for the material presented here.

The majority of the Dependability Assurance material in Chapter 3 consists of work from the following paper.

- Mark Louis Fairbairn, Iain Bate, and John A. Stankovic. Improving the dependability of sensornets. In *International Conference on Distributed Computing in Sensor Systems*, pages 274–282. IEEE, 2013

The Dynamic Duty Control work presented in Chapter 4 includes the material from the following paper.

- Mark Louis Fairbairn and Iain Bate. Using feedback control within WSN's to meet application requirements. In *International Conference on Distributed Computing in Sensor Systems*, pages 415–422. IEEE, 2013

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) are networks of small, embedded, battery powered wireless devices, called motes. Motes consist of a low power, general purpose microcontroller, paired with a radio and a number of external sensors. These networks are commonly used for data collection over a large spatial area, ranging from a few metres to several kilometres [6]. Within a typical network there are a small number of hard-wired motes called sinks to receive the data, and a large number of strictly wireless motes called sources, generating data from sensor readings. When the distances involved are large, and to avoid the need for high powered radios, messages are typically relayed via intermediate motes towards a sink. This relaying of messages is referred to as multi-hop communication.

Due to WSNs general purpose nature, they are easily interfaced with a range of sensors allowing them to show promise in a wide variety of applications such as: The Ferriby Road Bridge deployment [56], where five devices were deployed, one to act as the sink, three to monitor the size of cracks in the road bridge, and one to measure the incline of the bearings; Humber Bridge [57], where one sink, ten humidity and temperature devices, and one inclinometer were deployed to monitor the status of the suspension cables; and monitoring occupants within assisted living facilities [162], where motion, pulse oximetry, and ECG readings have been obtained from a single WSN device attached to an occupant, with a number of relay nodes providing connectivity to a sink, amongst many others covered in Section 2.4.

WSNs are also being used in ever increasing numbers due to the recent trend towards Smart Homes [23,144], where many traditional home features, such as lighting [135], Heating [99], and even appliances [42] are gaining network functionality. Primarily, the goal of the Smart Home is to allow data from multiple devices to be aggregated, and the devices

themselves to be controlled, thereby enabling new applications and adding functionality to traditional applications. With a vast quantity of generated data, together with controllability of devices, users are able to precisely control and monitor their environment, such as detecting intruders [105], reducing lighting when there are no occupants present [109] or even just monitoring of consumed power [14].

One primary attraction of using WSNs over other embedded solutions, is the reduced cost of deployment in terms of both money and effort, as WSNs require no existing infrastructure due to the motes being battery powered and wireless. Other factors contributing to their use are ever-decreasing hardware costs and a reduction in power consumption compared to earlier devices [111]. This allows for larger numbers of devices to be deployed and for the deployment to remain operational for a greater network lifetime.

Some of the main concerns surrounding the deployment of Wireless Sensor Networks are as follows: the ad-hoc nature of the deployments, the reliability of cheap hardware, the use of batteries which may become depleted, and radio interference (possibly from external devices) on the communications [102]. In addition there have also been a number of documented cases where WSNs have failed to perform as expected [119]. All these issues raise concerns about the reliability of WSNs, especially when their proposed use includes the monitoring of humans. Here, failure of the system could, in the worst case, put lives at risk.

This thesis will attempt to take a systematic approach to reliability by primarily focusing on the overall topic of dependability. This includes reliability together with a number of additional factors, such as, availability, safety, integrity and maintainability which will be discussed in detail within Section 2.5.

1.1 Dependability

When looking at the dependability of WSNs it is important to acknowledge that failures of the system need to be acceptable under certain conditions, as ultimately ensuring no failures of any devices within a WSN cannot be achieved. In these circumstances it is necessary to investigate how these failures affect the network, and how they can be mitigated. Failures cannot be avoided as ultimately devices always fail once all batteries have been depleted, or whilst the batteries are being replaced, possibly leading to the failure of the system. For instance, applications such as carbon monoxide monitoring cannot function while the device is not powered. Ultimately, it is necessary to plan for failure, as in the

extreme case external interference in the form of radio jammers could cause system failures that cannot be avoided by operational procedures. For these reasons, the following definition of dependability by Avizienis et al [11] is used within this work.

> *The ability of a system to avoid service failures that are more frequent or more severe than is acceptable [11]*

This definition accepts that service failures may occur, however to remain dependable we need to have bounds on the frequency and the severity of the failures. This raises the first issue, that dependability must be analysed with a focus on a particular service or application. To focus this work, a common application will be used throughout all the chapters. The application chosen for this work is assisted living, where regular monitoring of the environment and the occupant is required [7, 162]. Figure 1.1 shows a typical deployment of the application. In this deployment there are a number of static, multi-function devices placed around the building. Each static device takes sensor readings for one or more environmental factors such as temperature or light, whilst additionally providing message-relaying for surrounding devices. These readings can be used for a number of functions, such as occupant tracking and fire detection. An additional static device, hard-wired to the mains and the internet, is also included as the sink node for the network. A small number of mobile devices are also present which allow for direct monitoring of occupants as they move about the building. These devices record movement through the use of an inertial measurement unit (IMU). Movement information together with specific sensor data, such as Electrocardiogram (ECG), Electromyography (EMG) and Galvanic Skin Response (GSR) readings, are then relayed from the device to a sink.

Having outlined the dependability in terms of network failure, and the context in which the application will be operating, it is important to consider how the dependability of a network might be systematically analysed. This can be achieved through the systematic evaluation of the network against predefined attributes. Avizienis et al [13] defines five dependability attributes which are as follows.

Availability Readiness for correct service. In the context of assisted living, will the network be able to support both the static and mobile monitoring applications?

Reliability Continuity of correct service. Will the monitoring system continue to support the applications and what could cause this to stop?

Figure 1.1: Typical WSN deployment in the home

Safety Absence of catastrophic consequences on the user(s) and the environment. Can the applications fail in such a way that people and / or the environment are at-risk?

Integrity Absence of improper system alterations. Will environmental factors such as interference cause the system to fail, and can attackers cause failures?

Maintainability Ability to undergo modifications and repairs. How can device failures be handled and systematic replacement be achieved?

Availability and reliability are similar, with both being equal in the case of no failures. In the presence of failures the availability can be viewed as the aggregate uptime of the network, whereas the reliability my be the longest period of uninterrupted uptime. From this definition it can be seen that a network with ms-duration failures every minute may be highly available, however the reliability is particularly low.

These five factors are looked at in more detail within Chapter 3, which looks in detail at the static motes and their operation when communications are event-triggered, for example, fire detection. Events are raised based upon sensor readings, however this may be after a long period of inactivity during which time some devices may have failed due to power loss or hardware failure. For this reason, it is important to assess the Maintainability of a network using systematic tests, a main component of Dependability Assurance (DA). By analysing these test results, an effective heath monitoring system is derived, which uses low network resources to provide assurance in the state of the network at run-time.

Static motes are further considered within the context of data-streaming based applications, such as climate control, within Chapter 4. In these applications, data is constantly sent to the sink for offline analysis, resulting in high levels of network traffic and depletion of battery life. By taking the application timing requirements into account, this thesis proposes Dynamic Duty Control (DDC), which dynamically adapts the WSN duty-cycle to the environmental characteristics surrounding the network. This has the effect of reducing power consumption whilst still achieving the desired timing requirements, increasing the availability and reliability of the network.

Finally, the work introduces a number of mobile devices to track the movements of occupants in Chapter 5. To keep power consumption as low as possible, a mode-based approach is used to adjust the volume of data generated and transmitted, depending on whether the occupant is stationary or moving. These modes have strict timeliness requirements, which are met by using DDC, however the introduction of possible changes between modes requires an additional system called Mode Change Windows (MCWs). MCWs allow the timeliness requirements to be met when the system changes between modes, whereas traditional approaches frequently miss these requirements, resulting in data which has exceeded its deadline arriving at the base node. These three components: event-triggered communications, streaming, and mode changes are shown in Figure 1.2.

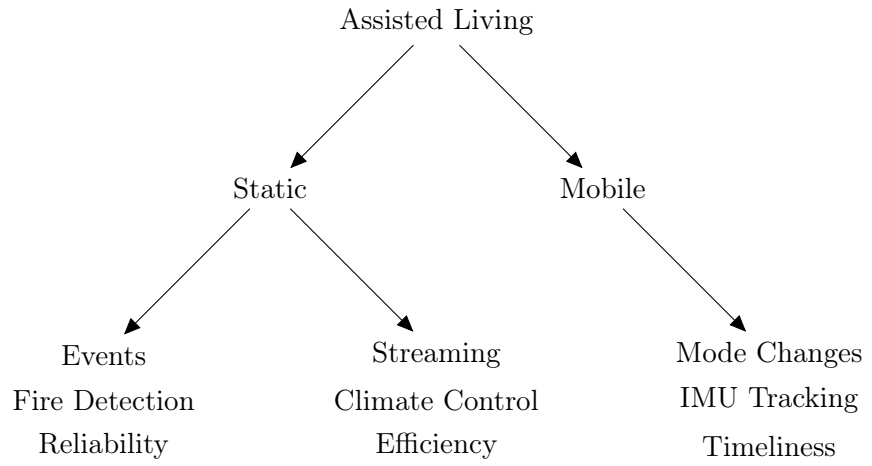

Figure 1.2: Overview of main contributions

1.2 Hypothesis

When taking a systematic approach to analysing the applications running upon a WSN the Dependability can be increased by considering the possible failure modes and the

application's requirements. Additionally by taking the application requirements and using feedback, reliability and efficiency can be improved whilst still ensuring application timing requirements are met.

1.3 Outline

This thesis will be structured as follows.

Chapter 2 provides a review of the current state-of-the-art. This covers the theory behind WSNs, including power consumption and radio communications. Following this, case studies on real-world deployments of WSNs will be reviewed and issues noted. Literature on dependability will be reviewed with particular emphasis given to those cases involving WSNs. The literature review continues, looking into ways of mitigating known issues through the use of approaches such as health-monitoring and adaptive network adjustment and how reactions to these issues can be performed. Finally a review of Operating Systems and Simulators is performed..

The first technical chapter, Chapter 3, will define Dependability Assurance (DA) in detail and how this can be applied to our example application. This section will outline how DA can be applied to meet the dependability attributes defined by Lepris et al, leading to the generation of Dependability Tests (DTs). The second part of this chapter looks in detail to the run-time checking of the DTs, with in-depth emphasis on Maintenance and Failure Rates of the devices and how this affects the dependability. This section concludes with requirements as to the failure times of WSN devices, showing how larger failure periods allow DA to perform more optimally, a requirement that will be met in the following chapters.

Chapter 4 aims to meet the failure requirements generated by DA, by taking the timing requirements extracted by DA and using these to reduce power requirements and thus increase the availability and reliability of the motes. This is achieved through the creation of Dynamic Duty Control (DDC), which takes the application timing requirements and information from the existing network traffic at run-time, and uses feedback to control the duty-cycle of motes within the network. DDC is analysed using numerical simulation, to compare it with current state-of-the-art approaches, and to estimate the power consumption. This raises questions about the effect of other MAC protocols and routing layers on the power and responsiveness of the application. For this reason analysis using Contiki is performed in order to analyse the performance of the network under Contiki's de-facto

standard MAC and routing layers.

Chapter 5 looks at data-driven applications which change timing requirements depending on the current mode of the network. When changing between modes, deadlines in either mode may be missed, invalidating the application requirements. To address this, the notion of Mode Change Windows (MCW) is introduced which allows the network to perform mode transitions sooner, and with careful scheduling allows deadlines to be met. This is demonstrated though the use of the UPPAAL model checker. The basic form of MCW requires the application developer to provide suitable network values to ensure deadlines are not missed, however this requirement is removed by introducing DDC to dynamically adapt to the network conditions, automatically providing the appropriate values. Finally, it is shown that by applying DA to MCW the requirement for additional data messages can be met, allowing for dependability and low power consumption to be provided within this scenario. Analysis of the application is performed on the current state-of-the-art MAC and routing implementations with further analysis to ensure the timing requirements are being met and power consumption is reduced.

Chapter 6 concludes with the results of the previous chapters, Dependability Assurance, Dynamic Duty Control, and Mode Change Windows. These are analysed with respect to the Dependability Attributes as detailed in Section 2.5.1, summarising how dependability is improved against the current state-of-the-art. Finally, suggestions as to possible further works are outlined.

Chapter 2

Literature Review

This literature review presents an overview as to the current state-of-the art in WSNs, with focus tending towards dependability by the end of the chapter. Initially an outline as to current hardware is provided with an overview as to the common software stack on these devices. A detailed overview of the communication stack, including the various protocols commonly used on these devices is then presented. This is followed by an analysis of WSN deployments and any issues that were identified such as power and unreliable communications. A formal definition of dependability is provided, along with a number of algorithms that cover various attributes of dependability. Towards the end of this chapter a brief overview of WSN operating systems and simulators are provided to inform any implementation which is required. In addition an overview of reactive approaches is provided to assist the work in Chapter 4. Finally the conclusions drawn from the literature review are presented, which outline the outstanding research challenges that this thesis aims to address.

2.1 WSN Overview

Wireless sensor networks are collections of small embedded devices called motes. The primary purpose of these devices is typically to sense the environment in which they are deployed using a large variety of possible sensors, and report these readings to a central location for analysis. More advanced systems may take action based upon this data, or may analyse it within localised areas before sending summarised data to a central location. The main differentiation between WSN motes and traditional monitoring techniques are that motes have been designed to be small, low cost, battery powered, and wirelessly com-

municate [149]. These factors allow for large numbers of devices to be deployed within a specific location, requiring no existing infrastructure to support the devices, whilst providing high levels of redundancy in the case of individual failures. In addition to redundancy the greater number of devices allows for large amounts of data to be generated over a wider area. This larger area allows previously expensive applications, such as forest fire detection [85], pollution monitoring [65], and precision crop monitoring [32, 86], to provide more utility as the locality of the readings has a large impact on the value of the data. For example, 3 high accuracy stations, whilst providing precise readings, may be less useful than 40 less precise sensors over a larger area. In addition large numbers of imprecise readings could also be used to obtain high precision values [65].

Communication is a fundamental difference between WSNs and traditional sensors, as the small physical size of the devices, restricted availability of power, and possibly large geographic area, makes direct communication over large distances relatively expensive, or in some topologies such as through the ground, impossible [82]. For this reason WSN devices use multi-hop communication to communicate over large areas by relaying messages from the source of the data, though intermediate devices to the destination, commonly called the sink. This not only allows the network to be deployed over a larger range than the transmission distance of individual devices, in addition the relaying of messages adversely effects the relaying nodes, as power and bandwidth is consumed on the relaying nodes. In the case of a single sink this means that the nodes closer to the sink experience more traffic [171], leading to a higher chance of failure. This increase in traffic and decrease in power as the sink is approached is referred to as the energy hole problem. The most promising standard for WSNs is the 802.15.4 specification, due to it having the lowest power requirements out of the three most popular alternatives, Bluetooth Low Energy (BLE), ANT, and 802.15.4 [20, 33]. In addition 802.15.4 has been proposed as part of the 6loWPAN protocol [101], allowing devices to be addressed using IPv6 [147], making it suitable for our assisted living scenario, and therefore has been chosen for this thesis.

These features make these devices attractive for a large number of scenarios where installing infrastructure would be expensive in either cost, such as freight monitoring [126, 127], or time such as in time-critical environmental disaster response [25, 125]. Another attraction of these devices is the low impact on the surrounding environment, for example in wildlife habitat monitoring [92] or forest fire detection [85,139]. A more detailed overview of previous WSN deployments is given in Section 2.4.

Whilst WSNs can include actuators to physically react to their surroundings, these types of WSNs are deemed outside the scope of this thesis due to the additional safety related effects that this causes, however the work presented within this thesis could be extended to support this type of application.

2.1.1 Devices

Wireless sensor networks are heavily constrained by the hardware upon which they operate. These constraints cover a variety of factors from power consumption, processing power, size, and weight. To better understand theses constraints an overview of common WSN platforms is presented, along with the current state-of the art. The end of this section will present a summary of the most important constraints that need to be considered.

The de-facto standard hardware in the majority of the WSN literature is the TelosB mote [118], which is also known under the SkyMote name. This device replaced the previously popular MicaZ [29], with both devices having the same architecture consisting of separate Radio and Microcontroller. Newer device such as the OpenMote-CC2538 [151] have begun to integrate the radio and microcontroller onto the same chip. Table 2.1 gives an overview of the three devices.

Device	**Year**	**Processor**	**Speed**	**Flash**	**RAM**
MicaZ	2001	8-bit ATmega128L	8Mhz	128KB	4KB
TelosB	2004	16-bit MSP430	8Mhz	1000KB	10KB
OpenMote-CC2538	2014	32-bit ARM	32Mhz	512KB	32KB

Table 2.1: Performance for the three mote types

From this table it can be seen that the word size of the processors is increasing with the newer processors, including the size of the RAM, and the clock speed on the newest processor. This implies that more intensive processing can be performed on these devices, lowering the performance restriction on WSN-class devices, and is especially true on the ARM-based variants due to higher performance of the processor [103]. This increase in compute power may indicate that more computationally expensive pre-processing of the data on-device may be a good approach to save power. Care must be taken however as none of these devices contain floating point units, meaning that any computation using larger size integers than the word size or floating point arithmetic must be re-written by the compiler, making computation slow, and therefore should be avoided. Another feature of these newer devices is the greatly increased size of the ROM. This allows for computation

/ space tradeoffs as lookup tables may be advantageous over computationally expensive mathematical functions such as exp().

All three devices are powered by two AA batteries located on the underside of the device. Table 2.2 shows the power draw for each of the devices.

Device	TX Power	RX Power	Active	Sleep
MicaZ	25mA	27mA	8mA	20uA
TelosB	19mA	22mA	2mA	5uA
OpenMote-CC2538	24mA	20mA	7mA	1.3uA

Table 2.2: Power consumption of the three mote types

This table shows that radio communications are not showing much change in power consumption between the generations of devices, an observation that has been supported in the literature [69, 165]. This issue means that the radio-on time is still the most important factor in device lifetime, as it has the highest power consumption, and therefore should be minimised as much as possible.

2.1.2 WSN Stack

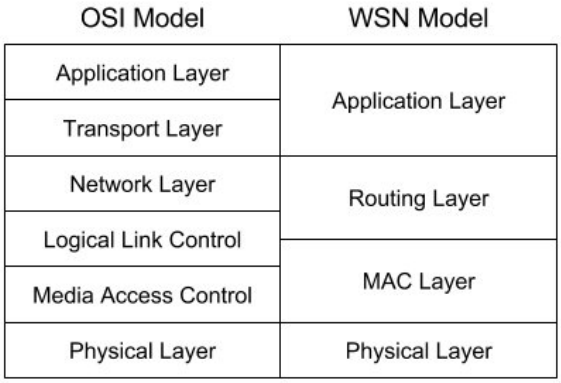

Figure 2.1: Seven Layer OSI vs Four Layer WSN

Typically networks are described in terms of the seven layer OSI model [173] as shown in Figure 2.1. This model is suitable for the desktop environment where applications are more abstracted from the underlying hardware, however for WSNs this model is overly complex. The 802.15.4 specification defines a four layer model as shown alongside the OSI model, and defines the implementation of the Physical and MAC layers. These four layers are defined as follows:

2.1.2.1 Physical

The physical layer in the 802.15.4 specification is the same as that used in the 7 layer OSI model. This layer deals with the management of the physical RF transceiver, including modulation of the physical signal, management of the channel selection, transmission energy levels, and signal management. Within this thesis the original 802.15.4 specification will be used, in particular the 2.4Ghz PHY. Since 2003, numerous additions have been made to add further PHY frequencies, channel hopping schemes, and other modulation schemes, however as the 2.4Ghz PHY is the most common it will be assumed for the rest of this thesis.

2.1.2.2 MAC

The MAC layer is responsible for the transmission of MAC frames through the physical channel. In the 802.15.4 specification for the largest size of a MAC frame is specified as 127 bytes [106] which must also include the appropriate headers as defined by the standard. For 802.15.4 devices Carrier Sense Multiple Access (CSMA) with Collision Avoidance (CA) is used, specifying that devices should sample the radio channel before attempting to transmit, with backoffs being applied if the channel appears busy [15, 75]. 802.15.4 specifies two modes of communication, beacon enabled and non-beacon enabled modes [172]. In non-beacon enabled modes nodes have no restrictions on when they can transmit, relying on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with the backoff periods to handle contention. The beacon enabled mode however used the concept of WPAN coordinators to organise transmissions into superframes, which are delimited by beacons broadcast by the WPAN coordinator [24]. These superframes contain active and inactive regions, which define the periods in which devices can enter lower power sleep modes. The active region is split into two further regions, the contention access period, where nodes must complete to transmit as with non-beacon mode, and the contention free period, which uses guaranteed time slots allocated to individual nodes for transmission. Further details of other MAC protocols are covered in more detail within Section 2.2. Another important classification is between contention-based protocols which employ CSMA/CA, which compete over the right to transmit, causing latency issues [174], and reservation-based TDMA protocols, which use pre-calculated time slots to communicate.

2.1.2.3 Routing

The MAC layer deals with single link communication, however one of the advantages of WSNs is that communication can occur over multiple hops to get from the source to the sink. To support multi-hop communication the routing layer is used. There are a number of different classifications of routing protocol [5, 134], however there are two main classes, static protocols and ad-hoc protocols. Static protocols have a fixed topology, typically specified by the system designer, whereas ad-hoc protocols adapt to the surrounding environment as required. Static protocols are useful within networks when the topology is well-known, and the designer does not want the overheads associated with Ad-hoc protocols. Ad-hoc protocols add additional complexity, however this allows them to be used in environments where the correct topology may not be known at design time, or where the topology may change depending on factors from the surrounding environment, such as interference. Ad-hoc protocols can be further classified into proactive and reactive protocols. Proactive protocols constantly maintain routing topologies regardless of the behaviour at the application layer. This allows for low latency initiation of communications as routes are already established, however at the expense of maintaining the routes even when they may not be required. Reactive protocols build routes on-demand, as required to support the application, which may require the route to be discovered if one does not exist, or repairing of the previous route if it has degraded since last use. Another class of protocol which is sometimes mentioned in routing protocols are Time-Division-Multiple-Access (TDMA) based protocols, which calculate global fixed schedule for nodes to transmit messages. Whilst TDMA protocols are strictly MAC protocols, they are commonly implemented as cross-layer solutions, and therefore have impact at the routing layer. There are also location-based protocols are a special class of static protocol where messages are sent to locations, with all nodes knowing their current location (possibly calculated during initialisation) making routing more efficient [157]. A brief overview of these will be given in Section 2.3.

In addition to the underlying routing protocols there are a number of additional features that can be added to further change the behaviour of the network, the first of which is clustering [1, 148]. Clustering approaches use logical groupings of nodes to create a singular virtual node, which can then communicate using standard routing protocols. These clusters use the notion of elected cluster heads which take responsibility for communicating on behalf of the cluster. The second feature that is frequently used is the addition of dynamic

information when discovering or updating routes [28]. The information used may include a number of QOS factors such as battery power levels, interference, and number of alive nodes, and can be used to add positive or negative pressure to certain nodes [58, 62, 169]. These additions are outside the scope of this work as we are focusing on dependability, and therefore are not discussed further.

2.1.2.4 Application

The application layer is the level at which the end application is located, with the support of the operating system to manage the underlying layers. At this level the raw information is collected from attached sensors, processed, and transmission of the data requested to a specific destination. The destination may be a single node as in the case of a sink, or the entire network as in the case of a broadcast. At this level data aggregation, compression, and analysis can be performed depending on the individual application [51]. This is especially true of data collection where the application does not require all the data, but a summary of the data in terms of a sum, average, min / max, as these operations can be performed easily in-network.

Support frameworks can also be provided at this level to make collection and analysis of data easier. These frameworks may support the execution of queries upon the network [49, 59, 90, 91], where the operator specifies a SQL-style query which is distributed across the network accordingly. Results from these queries are then collected in an energy-efficient manner, and aggregated back at the base node [83]. Other queries may allow events to be be detected and alarms raised at the appropriate time.

2.2 MAC

In addition to the 802.15.4 MAC specification there are also a large number of alternative 802.15.4 MAC protocols that have been developed within the literature. Most of these protocols are compatible with the official specification, using the same header structure in the MAC frames and the non-beacon enabled mode. As the non-beacon enabled mode fails to specify any restrictions on the radio states of the devices, specific MAC protocols are free to apply additional duty cycling features. For these reasons popular MAC protocols such as Low Power Listening (LPL) [100] and ContikiMAC [36] are both 802.15.4 compatible, but have issues interoperating due to their differing duty cycling behaviour [70]. This

section will describe a number of MAC protocols, however TDMA protocols will not be described in this section due to their strong relationship with the routing layer, instead both the MAC and Routing layers of TDMA protocols will be described in Section 2.3.1.

The most basic approach, commonly referred to as nullMAC [18], keeps the radio in a constant listening state, waiting for the start of an incoming data packet. If a node wishes to send a packet it switches into the transmit state, sends the packet, and then returns to the listening state. For the majority of the time with this approach the nodes are idle listening. Idle listening is when devices are listening to the radio with no devices transmitting at the same time, in effect wasting time in the relatively power expensive radio-on state.

The advantages of this mode is that sent messages are never missed by the receiver being in the sleep state, and this is the method used for 802.11 base stations [31]. This mode however consumes a high amount of power, therefore the radio should be in the off-state by default, with the radio periodically woken up to perform communications, referred to as duty-cycling. Whilst duty-cycling typically reduces the power consumption of mobile devices it introduces the need for larger buffers to hold packets generated whilst the device is offline, ready to be sent at the next awake period. In addition to the required buffers, any time that a packet has been generated, but can't be sent due to the radio being in the offline state, leads to increased delay in the packet reaching the destination [93]. This delay could be unacceptable for certain applications if it became too large.

In Sensor MAC [168] (S-MAC) all nodes share a common duty cycle between neighbours, defining the period of the duty cycle and the amount of time the device remains alive in the awake period (active period). The active period is further split into two sections, the time synchronization period and the data transfer period. The time synchronization period is used for nodes to exchange duty cycle information, allowing neighbouring nodes to use the same duty for communication. This synchronization is performed at the start of every active period so that clock drift does not become a significant issue. The active period can be extended in the case that transfers have not completed by the end of the active period, however the active period cannot be reduced in size. This MAC protocol reduces the idle listening time, but due to the minimum active period size it is not fully minimised, and with the addition of a synchronization period some overheads are introduced.

Timeout MAC [154] (T-MAC) aims to remove the limitations of S-MAC, mainly the requirement for a minimum awake period. This is achieved by defining a fixed period at the

start of which the radio is turned on, accepting the transmission of all data packets. When the radio is activated a timer is enabled which counts down for some time T, whereby when the timer expires the radio is turned off until the start of the next period. To provide the dynamic sizing of the transmission window the timeout is reset every time a packet is sent or received, effectively extending the active radio time. Care must be taken to ensure that T is large enough to ensure that a single packet sent by any other node has time to be received, however a large value adds to the idle listening time at the end of the transmission window.

Berkeley Media Access Control [116] (B-MAC) is one of the first algorithms to reduce the idle listening time of all the nodes by making the sender node consume more time transmitting. B-MAC operates with receivers periodically probing the channel, assessing if there are any messages being sent, and if this is not the case, quickly returning to the sleep mode. If a message transmission from another node is detected the receiver stays awake and attempts to receive the message. Unlike previous MAC protocols the sender cannot assume that the receiver will be awake when the message starts transmitting, and instead a preamble is initially broadcast. This preamble allows receiver nodes to wake, detect a transmission, listen to the radio, and at the end of the preamble, receive the message. This protocol ensures that at no point are the sender receiver pair both awake and idle listening, whilst waiting for the end of a transmission period as in the previous algorithms. Instead the idle listening is reduced to the time to check for channel activity, with wasted radio time being relocated to the time to transmit the preamble and the length of time receivers listen to the preamble. It has been shown to be more efficient than T-MAC in the majority of scenarios [133], and forms the basis of the remaining MAC protocols.

X-MAC [22] improves on B-MAC by reducing the amount of time that the preamble is both transmitted and received by the sender and receiver respectively. This is performed by replacing the long preamble with a repeating short preamble containing the destination node and the number of remaining preambles, with a small break between each repeat. The size of this break is calculated to be large enough to receive a simple acknowledgement from any receiver nodes, halting the transmission of the preamble and beginning the transmission of the data. Receiver nodes periodically wake, sample the channel for longer than the acknowledgement window size. If a preamble is partially received the node stays awake to receive the next complete preamble, compares the destination with its own ID, and in the case of a match, sends an acknowledgement. This protocol reduces the preamble time in

addition to providing a rapid way for listening nodes who do not match the destination ID to return to sleep. This method does however increase the size of the listening period for the receivers, but drastically reduces the preamble time.

Both TinyOS and Contiki operating systems use similar methods as their default MAC protocols (TinyOS LPL [100] & ContikiMAC [36]). These protocols operate in the same way as X-MAC, however the data packet is used as the preamble instead of a specially formatted preamble packet. This allows the received acknowledgements to signal the full reception of the data message, and the end of the sender's transmission. In this mode the data packet is repeatedly transmitted by the sender, with a pause between each transmission that is sized to be large enough for a receiver to acknowledge the receipt of a message. Listening nodes periodically wake up and sample the radio for a duration longer than the pause interval between two packets in which they sample the radio. If a message is detected the receiver stays awake to receive the next complete message. When the whole message has been received an acknowledgement is sent to inform the sender that message has been received and that transmissions may stop. In the case of ContikiMAC further sleep is performed by the receiver around the pause, instead of sampling for the entire pause duration. This is done by sampling for two short periods, separated by the size of the acknowledgement pause, ensuring that if one sample was to fall inside the pause, the other would hear the message if one exists.

ContikiMAC also provides another couple of optimisations to reduce the time spent waiting for communication. The first is the concept of phase synchronisation between the sender and the receiver node, whereby upon a successful transmission between a pair of nodes each node remembers the transmission time of the other device. When these two devices wish to communicate for a second time they wait for two preamble periods before the receiver is due to awake before starting transmission, reducing wasted communications. The second optimisation is when the sender has multiple messages to send. In this instance a special flag is set on each message that is not last in the queue, informing the receiver that more messages are pending. When the receiver notices this flag on the first received packet it does not sleep for another period, instead waiting for the next data message to arrive. Once the queue has been reduced to a size of one the flag is omitted, allowing the receiver to sleep once communications are complete.

Both TinyOS and ContikiMAC allow the operator to specify the duty cycle that should be used by the nodes in the network, which in turn specify the responsiveness of the network

and the upper limit on the power savings. Further guidance is not given on how to specify these duty cycles, with TinyOS defaulting to 10Hz and ContikiMAC defaulting to 8Hz.

2.3 Routing

As stated earlier there are a number of Dynamic and Static Routing protocols. Dynamic protocols include Proactive and Reactive protocols, with Static including TDMA based approaches. A sample of representative protocols for each of these three categories is provided, with the original work being reviewed instead of any enhanced versions, which themselves will be summarised.

Flooding [84] is the most basic form of communication within WSNs. Flooding is performed when a node broadcasts a message to all neighbouring devices, which in turn broadcast the message. Nodes remember the last broadcast message, and therefore will not re-broadcast the message. This ensures that the message covers the entire network, however at the expense of power on all devices. Time To Live (TTL) values may be used to restrict the number of hops which the message may take [52]. Each device reduces the TTL by one, and should an incoming message be received with a TTL of 0, the message is no longer forwarded. Whilst this reduces the impact on the network it shortens the range which the message travels, which may be useful for localised activities [110].

2.3.1 Dynamic - Proactive Protocols

There are a number of routing protocols which constantly maintain routing information for the network. These protocols require constant maintenance to compensate for the dynamic environment in which the WSN operates, with some protocols requiring maintenance to ensure various QOS factors are maintained.

Destination-Sequenced Distance Vector Routing [115] (DSDV) uses routing information gathered periodically by all nodes to ensure that a message reaches its destination. When a message is generated the destination is looked up in the local routing table on the sending device to identify the next node in the route to the destination. Once the next hop has been identified the message is forwarded and the process repeated until the destination is reached. This protocol requires that all devices know all routes to all valid destinations within the network, with these routes being contained in each nodes' routing tables. To ensure constant information between devices each device periodically broadcasts its routing

table to its neighbours, which in turn increment the distance to the destination, and if the distance is smaller than any existing record to the same destination, the old record is replaced with the sender being recorded as the next hop in the route. This ensures that once all tables have been fully propagated, all nodes know the next hop on the shortest path to all possible destinations. There are many optimisations that can be used to reduce the overheads of updating this information, such as sending only updates to the local tables or adjusting the frequency of the updates over time [67], however these are outside the scope of this review.

There are also a number of protocols which operate in a similar manner to DSDV, however using different cost metrics. Where DSDV uses hop count as the deciding factor as to which route should be prioritised, other protocols may incorporate factors such as battery levels, age of records, communication quality and other QOS factors.

Routing Protocol for Low Power and Lossy Networks [2] (RPL) is a relatively new routing protocol (2011) which uses the concept of Directed Acyclic Graphs (DAGs) rooted at a single node called Destination Oriented DAGs (DoDAGs) to route packets around the network. These graphs are created in an iterative nature similar to DSDV, with the route node sending out DoDAG Information Objects (DIOs) identifying the sender and their current rank. When DIOs are received the sender's rank is compared to the current parent of the receiver, and if it is a higher rank it is selected as the new parent, its rank is calculated and a new new DIO message is broadcast. This method ensures construction of loop-free DoDAGs, with messages being sent up the tree to the route node.

RPL differs from similar tree-based algorithms by supporting downward links through the use of Destination Advertisement Objects (DAOs). DAOs allow nodes to broadcast their presence and that of all known child nodes to parent nodes in the tree. When a DAO message is received, information about the child and any grandchildren are stored. DAOs allow point-to-point communication by sending messages up the tree until a route to the destination exists, in the worst case at the route node, before traversing back down the tree to the children. Point-to-point communication however is relatively expensive in terms of hop count due to routes navigating the DoDAG [164]. Other differences include the support for arbitrary ranking functions, with support for multiple simultaneous DoDAGs routed at different nodes, with each possibly running different ranking functions. However RPL as a whole, whilst supporting many different features takes a large amount of Flash and RAM space making it unsuitable for devices with limited resources.

2.3.2 Dynamic - Reactive Protocols

There are another class of protocols which do not attempt to maintain any routing information until transmission is requested by a node in the network. When a transmission is requested routes are discovered before the data is sent to the destination, thereby removing the need to constantly maintain routing information.

Destination Sequenced Routing [64] (DSR) is a routing protocol which discovers routes between the source and the sink when transmissions are attempted and there are no pre-existing routes. If a source has never attempted to send a message then no routes will currently exist for this node. Routes may also no longer exist if previous routes between the source and the sink have expired due to inactivity over a period of time, or if the previous route was detected as failed. New routes are discovered by broadcasting a route request (RREQ) across the network, with each hop appending the ID of the relaying node to the packet. Once the RREQ reaches the sink a unicast reply (RREP) is sent back along the path contained within the packet header. Once the source receives the RREP the route is extracted from the packet header and inserted into all future transmissions to the same destinations so that the message can be relayed appropriately. Should any hops in the transmisssion path fail, a route error (RERR) is sent back to the sending node, which in turn removes the path from cache and re-initiates route discovery accordingly.

Ad-Hoc On-Demand Distance Routing [26] (AODV) is similar to DSR in the use of RREQ, RREP and RERR messages, however instead of storing the route to be taken inside the packet, which may consume a large amount of space if the route has many hops, the route is cached in the relaying nodes. The caching of routes is performed when the RREQ message is relayed by intermediary nodes, with the RREQ creating the backwards path and the RREP creating the forward path (as only the sender ID's are known). When receiving a RREQ message the source is recorded with the previous hop ID, and the message re-broadcast. At the sink the RREQ is sent back along the same route using the cached source and previous hop ID at each node in the route, with each hop also storing the source of the RREP and the ID of the hop ID.

All AODV implementations use the same method of route discovery, however implementations differ in their handling of both the timeout and route error cases. In some implementations if a route has not been used for a set period of time it is removed from the routing table, some other implementations however do not use a timeout and instead use an evicted policy where the least recently used route is evicted from the routing table

should a new route need to be established and the table is full. Other variations exist whereby routes are never evicted, instead waiting for an error to occur on one of the hops before attempting recovery. Some implementations attempt local repair using a localised broadcast [110], returning a route error (RERR) if the repair fails, whereas other implementations return the RERR immediately [77].

2.3.3 Static Protocols

Whilst Time Division Multiple Access (TDMA) protocols are strictly MAC level protocols, they are typically used with a different set of routing layer protocols, and therefore are both described in this section. TDMA protocols use the concept of a superframe, which is a periodic schedule split into a number of slots, with each slot defining which devices are allowed to transmit at that point in time. As a strict transmission schedule is defined collisions between devices in the network are mitigated, with the behaviour of the network being strictly defined. Whilst such approaches can remove intra-network interference, external interference may still exist, and as such a device may still fail to transmit in its allocated time slot. Initially this scheme looks like only one node can transmit within the entire network at one time, however some protocols may allow multiple nodes to be assigned to the same time slot which, should the nodes be sufficiently spaced apart within the network topology, should not interfere with others.

There are a number of TDMA protocols that could be analysed, however we will only analyse WirelessHART in the interests of conciseness as it is the industry De-Facto TDMA approach. Whilst WirelessHART [140] uses the same concepts as standard TDMA schemes, with superframes being split into 10ms transmission slots, it also allows more freedom when specifying the schedules of the devices in the network. Primarily the number of superframes can be greater than one, allowing different superframes to co-exist for different purposes. Another enhancement that WirelessHART introduces is the concept of each slot having an associated channel, allowing network schedules to implement channel hopping. Finally the method used to populate the schedules is left undefined, allowing the network managers to schedule communications as desired [78]. This allows for tree-based communications to be scheduled easily, or any other approach.

2.3.4 Summary

There are a much wider variety of routing protocols within WSNs [4], including combinations of existing protocols [27], and frameworks to support multiple routing protocols [117]. This section has reviewed a number of the more common protocols, and from this review AODV has shown itself to be the most suitable solution for the broad array of applications that this thesis targets, and therefore will be used within this thesis. This suitability is due to its good average-case performance [98, 152] and minimal management overhead when compared to static routing topologies or TDMA-based approaches [21].

2.4 Deployments

Wireless sensor networks have been used in a large number of different scenarios, with the majority of deployments being monitoring applications without the inclusion of actuators, serving as purely information systems. This section will categorise these applications, firstly into event driven systems where the devices only inform the sink when an event has been detected from the sensor readings, and secondly into periodic systems where the applications sense the environment and report it to the sink at regular intervals.

2.4.1 Event Driven

The first class of systems to be reviewed are event driven systems. These systems typically monitor the environment waiting for a specific external event to occur, detected through some form of classification on the device itself or a cluster head in cluster based systems. Once an event has been detected the sink is informed, typically consisting of just the event, or in some cases including the data that caused the event to occur [104]. These offer higher power savings than periodic systems, and with the inclusion of hardware support to wake on events [53], power can be reduced further.

Events may be generated in a variety of ways within the WSN mote. These could be raised directly due to some external event such as a PIR sensor sending a "detected movement" signal to the mote, or they may be the product of more complicated processing of raw data. Processing of the raw data can be performed using a variety of methods which can be viewed as having some model from input signals to event classifications [34]. In the basic case this could be simple thresholding of data, where the data being over a set value indicates the raising of an event. More complicated models may also be used, taking into

account time, multiple signals, and other factors, These complicated models may cause less false positives for the same level of accuracy, however they introduce the issue of creating and updating the internal models on the devices. Some specific examples are discussed in greater detail below.

Solutions such as CollECT [132] perform event detection through the use of multiple types of sensors in a distributed and heterogeneous network. CollECT aims to locate the source of the event through triangulation with other sensors, and continues to track the event as it moves. To do so it assumes that the location of all devices is known, and therefore can be used to compute the location of the event. Unfortunately evaluation was performed within a custom simulator, with no regard to anything beneath the application layer such as routing, making analysis limited, since factors such as collisions and timeliness are omitted.

Application-specific event monitoring applications have been proposed, such as automatic monitoring of car parking [76]. SPARK [141] is one of the approaches that use real WSN devices within a car park test bed. In this application car parking spaces are monitored for occupancy using light sensors, with the occupancy information being used to direct new cars to free spaces. Whilst this application was performed on a small test bed it demonstrates that useful information can be derived from small numbers of event-detecting devices. Further tests have been performed by Lee et al [76] using range sensors to detect vehicles entering and leaving a multi-storey car park. As with the previous example event information is only sent once a car is detected, dramatically reducing the amount of radio traffic considering the high sample rate of the sensor which would require a message being sent every few seconds.

Other methods may change modes depending on events being detected. Mercury [87] identifies that the rate of data from sensors can be read faster than they can be transmitted to the sink. For this reason the Mercury protocol has a classifier on-device to assess the readings, reporting events to the sink should they occur. An additional feature of Mercury is the ranking of sensor data, with only data of high interest being sent to the sink to conserve power. This identifies that there may be some situations where the application may change what is reported to the sink depending on the apparent utility of the data and the current operating conditions.

There are also additional methods that may be used to control the flow of data within the WSN to decrease power consumption. Simple methods may indiscriminately remove

parts of the data, such as down-sampling performed by removing every Xth reading. This is the most basic approach, however depending on the application it may also be degrading the quality of the data, especially where the data contain features of interest that only occur for a small number of samples. Other more complex schemes may attempt to compress the data, either through lossless or lossy schemes [136]. Lossless schemes involve using more efficient encodings of the raw data, I.e. compression algorithms such as Lempel-Ziv-Welch (LZW) and derivatives [50].

2.4.2 Periodic

There are a number of applications where periodic sampling of the environment is performed, with the raw information being relayed back to the sink. Raw data may be useful in a number of scenarios: when cross-correlation with other devices is required; analysis of the data is computationally expensive to perform on the devices themselves; or off-line analysis is to be performed.

One popular example of this periodic reporting is in structural health-monitoring, due to the lack of infrastructure required to deploy a network. This monitoring typically uses microelectromechanical sensors (MEMS) to detect vibrations in the structure being monitored, with the readings being sent to the sink for analysis. The sample rates of these sensors can vary from 100Hz up to the kHz range [39, 89], indicating a large variation in the amount of data that may be relayed to the sink. Other typical periodic systems can be found within assisted living scenarios [53, 155], where a number of factors such as motion, location, and vital signs may be monitored, with single sensors undertaking multiple roles depending on their location. There are also a large number of opportunities within the agriculture and food industries [128, 145, 158], where applications such as monitoring greenhouses, animals, and food, can all benefit from the low installation costs due to the lack of required wiring. Finally any traditional scientific monitoring use cases such as monitoring glacier movements [108] also show promise for WSNs due to the higher number of readings that can be collected compared to manual measurement.

Due to the large amount of data periodic sampling provides there are a number of approaches that can be used to reduce the volume of data within the network, and thus increase the lifetime of the network. These can either either be simple approaches which aim to reduce the amount of data by dropping "less important" data, or in-network aggregation or processing of data. Simple rate-limiting can be implemented by ranking the incoming

data and then prioritising the delivery of the data based upon this ranking [94]. This requires that some metric of data importance is used, which may be difficult depending on the use of the data. In-network aggregation and processing requires that the data must be exposed to the application layer at each hop, with the application performing some actions upon the data before forwarding to data to the next hop in the network. One method that supports automated aggregation of data is Maximum Lifetime Data Aggregation algorithm [66] (MLDA), where queries are constructed and broadcast across the network. Based upon the type of query and the routing topology, specific operators can be processed as the data moves towards the sink. An example of such operators are MAX and SUM, where it is trivial for a routing node with a number of messages to aggregate these into one message. More complex schemes can also be used, for example if an application must detect an event based upon the data from a number of nodes, once the data being sent from the nodes to the sink has can be used to make a decision, the raw data can be dropped and the event message substituted for the rest of the hops [68].

Where lossy data compression is used there is an explicit trade-off between the quality of the data and the lifetime of the network, leading to a clear accuracy-dependability trade-off. This work focuses on improvements to dependability with no information on the contents of the application data, instead focussing on delivering the data in a timely manner whilst improving network lifetime. This however does not reject in-network data aggregation, as a reduction in the network traffic allows DDC to reduce the power consumption further, making data aggregation a complementary approach to this work.

2.5 Dependability of WSNs

Within the WSN literature there have been a large number of works which look at the reliability of WSNs, specifically the communications between devices. To mitigate communications issues various mechanisms have been used, such as coordination between nodes to identify and avoid faulty nodes, and channel hopping to avoid specific sources of interference. The majority of these works focus on reacting to faults as they occur, however a small subset of approaches use fault injection to simulate errors before they occur, taking appropriate action to reduce the likelihood of network errors should a real error of the same type occur [129].

The issue with these works is that focus is only given to a single type of failure, known as failure modes, typically the case where messages that were sent are not received (omis-

sion) [8]. There are however a number of other failures that could occur, with current works selecting a subset of possible failures [30, 130], however systematic analysis should be performed to ensure that all failure modes have been accounted for [88].

2.5.1 Failures, Hazards, and Dependability

Historically Failure Mode and Effect Analysis (FMEA) [114] provided a systematic method for performing failure analysis. This is done by analysing the target system to identify the various failure modes, which in turn are used to identify the associated causes and effects. These causes can then be addressed to ensure that the chance of their occurrence is as low as reasonably practicable (ALARP) [97]. ALARP acknowledges that the complete removal of a specific failure mode is extremely difficult, instead the chance of the failure mode occurring should be reduced such that it is either of lower probability of occurring than some other equally damaging failure mode (which can be analysed using fault trees), or it has been reduced such that it is highly unlikely to occur. FMEA however provides little guidance on the discovery of failure modes, and therefore a process called hazard and operability study (HAZOP) can be performed under certain circumstances to aid in the identification of failure modes [35]. HAZOP requires the decomposition of the target system into a set of distinct subsystems or functions, which are then analysed for deviations from their normal behaviour using a set of keywords such as More, Less, Part of, Reverse. The effects of the deviations are recorded and any hazardous states are identified to be reasoned about in terms of probability of occurrence, or methods of mitigation. Whilst this approach provides more guidance than FMEA, its application to the software domain still requires guidance. For this reason Software Hazard Analysis and Resolution in Design (SHARD) was developed to provide a systematic approach, not only to the application of a HAZOP approach to the software domain, but to analyse the results and suggest actions to rectify the identified faults [96]. HAZOP requires that the system be decomposed into a number of individual components connected with flows of information. These flows can then be analysed by applying a set of keywords such as Omission, Commission, Early, Late, to obtain the possible failures to be analysed further.

Whilst HAZOP ensures that suitable exploration of the failures modes is performed, it typically only concerns the reliability and availability of the network. To encompass reliability, including other factors that are deemed by some to make a system reliable, Avizienis chose the term dependability to serve as the encompassing term [12]. By using the term

'reliability' the vague limits of traditional 'reliability' can be categorised. Dependability is defined by the following dependability attributes.

- Availability - Readiness for correct service
- Reliability - Continuity of correct service
- Safety - Absence of catastrophic consequences on the users and the environment
- Integrity - Absence of improper system alteration
- Maintenance - Ability for a process to undergo modifications and repairs

The items that can affect the dependability of the system are as follows:

- Faults are the defects that can occur within the system. Faults may not lead to failures, however this would depend upon how the faulty part of the system is used and the way in which is it used.
- Errors are differences between the intended behaviour of the system and its current behaviour and they occur due to faults. Errors, much like faults, may not cause failures, as the error could be detected and corrected before it had a chance to propagate into a failure.
- Failures are when the system exhibits external behaviour which is different to that originally intended, and can be caused by unhandled errors within the system.

To reduce the chance of failures there are four methods that can be used to increase the dependability of a system, these are as follows: Prevention, Removal, Forecasting, and Tolerance.

At the time of writing there is limited literature on dependability with respect to wireless sensor networks, with many works focusing on small subsets of availability and reliability. These works are categorized into three sections, health-monitoring, fault injection, and power conservation, and are summarised in the following sections.

2.5.2 Availability & Reliability - Health-Monitoring

With most of the literature it is assumed that node replacements cannot be performed, and so once a node dies, the service that it provides is permanently lost. In some circumstances knowing when the system has degraded beyond being useful is an important factor, as the

system can no longer be relied upon, leading to the requirement for health-monitoring of WSNs. In some circumstances however it may be possible to replace nodes once they have failed, either by repairing the node (replacement of batteries or hardware), or the equivalent by deploying another fully functional node at the same location. This replacement of nodes is referred to as maintenance and also requires that the current state of the network can be measured using health-monitoring.

2.5.2.1 Heartbeat

Heartbeat [3] (HB) is one of the most basic forms of health-monitoring systems. The main component behind HB is the periodic broadcast from a node to its neighbours, informing them that it is still operational (a heartbeat message). When a heartbeat is detected by a neighboring node it is recorded with the current time as a watched neighbour. If the neighbour is already present then the time is simply updated. Periodically all the records are checked to ensure that no neighbour has failed to be detected, and if so, an alert is sent to the sink node with the ID of the node in question. This method ensures that no ahead-of-time information is required about the positioning of neighbours, with the broadcasts being single-hop so as to minimise network traffic. The timeout for the heartbeat can also be set to greater than twice the sending period of the heartbeats to provide tolerance to network fluctuations, however this is not analysed within this paper. Not only does this method check that a node is still powered, but also that the communication links remain stable, with an alert being raised if this is not the case.

2.5.2.2 Run-time Assurance

Run-time Assurance (RTA) [163] is one of the first works that looks at ensuring the availability of the application, as opposed to ensuring the availability of all devices and communication links. The authors note that in the vast majority of cases a number of failures may be permitted as long as the overall application can continue to provide service. The example provided is that of a fire detection system, where the temperature is used to detect fires within a building. In this example should there be multiple nodes within a room then failures can be tolerated until no nodes are operating within the room, leaving fires undetected.

To detect when the application fails to meet its requirements RTA first requires that the application is modelled using a Sensor Network Event Description Language (SNEDL)

diagram. This diagram models the data flow between devices, capturing the dependencies between readings [63]. Using this model the application is automatically analyzed, producing a series of online tests which must be periodically performed. These tests ensure that the data flows captured in the model are still achievable, and when they are not an alert is raised. RTA ensures that only the minimal number of tests to meet the application requirements are performed, reducing unnecessary overheads.

In effect this system runs simulations of the application, and upon the failure of a simulation, an alert is raised. This system however only raises alerts once the system has failed, unlike HB which informs the user when the system has degraded, and therefore requires that maintenance to solve the failure is undertaken in order to quickly minimise the time-at-risk.

2.5.3 Fault Injection

There have been a number of works that use fault injection to check if the network can tolerate specific types of failure [8]. These works aim to simulate a specific failure case and check if the networks' normal operations are disrupted by the injected failures. Unlike RTA, which generates failures from a model of the system, many of these works are developed to target a specific failure case [112]. Whilst this makes the approaches application-specific it does however allow degradation in service to be measured, allowing maintenance to be requested before failure of the system occurs.

These approaches provide an attractive way to measure the degradation of the system, however the features to test are not derived in a systematic manner such as that used by RTA to devise its tests. For this reason it may be prudent to use fault injection as a method to ensure any dependability requirements are met at run-time, and to take preventative action before the system fails.

2.5.4 Availability & Reliability - Power Conservation

In addition to monitoring the availability of the network, the overall reliability of the system can be increased by decreasing the power consumption of the network, and therefore increasing the amount of time nodes can operate without power failure. Energy consumption in WSNs has a large amount of literature due to the increase in power consumption growing faster than the available battery power, the so called "battery gap" [73]. The majority of works to date have focused on power savings at the MAC and Routing levels

of the network stack. As this field is under heavy active research, and has a number of de-facto solutions such as ContikiMAC and AODV / RPL, and with active research into power reduction at the lower levels, for example varying transmission power [123], it has been decided to focus on the conservation of power at the application level.

The application level can only save power by reducing the number of messages that it requests to be sent, and therefore requires that it can interpret the application data to realise appropriate power saving techniques whilst ensuring that the application can continue to operate uninterrupted. An example of this would be data compression, where the application has been analysed to deduce that only the maximum value of all sensor readings is important for the application. Using this example each node on the route from the sources to the sink could relay only the maximal value from all child nodes, reducing the number of messages sent whilst ensuring the application operates as intended. Further data reduction examples can be found in Section 2.4.2. Another factor that can be used at this level is the timing requirements for messages to reach the sink, as higher latency tolerances allow for data to remain in-network for longer, possibly allowing higher levels of data compression to be achieved.

In addition to power conservation there are also a number of additional techniques that may be used to scavenge energy from the surrounding environment, commonly referred to as energy harvesting [61]. These techniques range from using solar panels to obtain power, even within indoor environments, to harvesting power from vibrations in the surrounding environment. These typically only provide a small amount of power and therefore are unsuitable when communications are frequently required.

2.5.5 Safety, Integrity, and Maintenance

Whilst other works exist on factors such as integrity, the two factors with little literature are safety and maintenance. Safety focuses on the safety of humans, specifically how the WSN may directly harm humans. The core WSN devices are safe, due to their low power and lack of actuators. In some circumstances the data provided by the WSN may be used in safety related scenarios, however in these cases the WSN itself cannot harm humans, instead it is inappropriate reliance on the WSN data that is the concern. The only way a WSN can cause direct harm is with the inclusion of actuators, for example controlling high power loads, operating machinery, or other such scenarios. Within this thesis it is assumed that the WSN is only used to gather information and therefore does not have any physical

actuators which could harm humans, and therefore safety is not a concern for this work. The other attribute that is commonly omitted is Maintenance, with the common view that once the WSN has been deployed the network is fixed, and will continue operating until power is exhausted. This work assumes that this is not the case, as more nodes can be deployed inexpensively, and in some circumstances the replacement of batteries within devices, as long as it is not too frequent, is acceptable. Where the data produced by the network is used in a safety-related context the use of the data is safety relevant, however the network itself is still not safety relevant as it cannot directly cause harm.

2.6 Reactivity

Wireless sensor networks are deeply affected by their surrounding environment due to a number of external factors such as communications interference, changes in topology, and even temperature (which can affect battery performance). For this reason a number of applications require that WSNs react to the changes in their environment in order to handle these fluctuations, whilst ensuring that these reactions are stable (free from oscillations), so that battery power can be conserved. A number of approaches aim to analyse these factors before the deployment of the network, using simulations to assess the robustness of the network to possible environmental changes, to optimise the deployment of WSN nodes. These can involve genetic algorithms [46, 74], simulated annealing [16, 137], reinforcement learning [131], and other search-based methods. As these approaches require candidate solutions to be evaluated, simulations are required, which are computationally expensive to perform on-line.

As battery power is an issue, a method needs to be used that can rapidly decide what action to take in a specific situation. For example if packets are taking longer to reach the sink than desired, how should some internal variable such as transmission rate be modified? Simple conditional control with predefined rules leads to undesirable oscillations, and therefore an approach that reduces the change in a variable as the output gets closer to the desired value is required. These approaches can be summarised by proportional, integral, differential (PID) controllers which can be configured to provide the same behaviour and can be further configured to provide additional control. More advanced techniques may use learning-based protocols to either classify values into discrete groups, or to provide continuous output values. Whilst there are many approaches that may be used, we will focus on two distinct methods, PID loops and neural networks (NNs).

2.6.1 Control Theory - PID Loops

PID loops are commonly used within industrial control systems as a method of closed-loop control, a process where the output value of the process is used as an input to the control loop, allowing the system to modify its behaviour until the output value is as desired [10]. PID loops operate on the principle of error to a target value, known as the setpoint. This error is calculated as the difference between the input value and the setpoint, before being used within the three internal components of the loop, in order to generate a new output value. The three internal components are as follows:

Proportional The proportional component takes the error value, multiplies it by the P component, and uses this value as the output. This component ensures that as errors increase, the output value is varied such that the error should be decreased. As this value has no internal state, and so is always at a fixed ratio of the input, it may stabilise away from the desired setpoint, and therefore other components must also be used.

Integral The integral component uses the input error and adds it to the cumulative error over time. This cumulative error is then multiplied by the I component before being sent as the output. This component ensures that whilst the output is not equal to the setpoint, the output will be constantly varied so that it approaches the setpoint. Due to the setpoint being the cumulative error any systems that do not have instantaneous evaluation of the output may overshoot, due to the increase in pressure whilst the setpoint has not been reached. The buildup of the integral value whilst the controller is far from the desired setpoint is called windup [113]. Windup can be mitigated with various techniques such as varying PID values, limiting the maximum accumulated integral factor, and resetting the I component at specific points.

Derivative The derivative component is used to dampen the responsiveness of the PID loop and operates by taking the error and subtracting it from the previous round to compute the change in error. This change is multiplied by the D component before being output. This ensures that a sudden change in the input does not create large fluctuations in the output values, helping smooth the behaviour of the controller.

These three components are finally summed and output as the result of the PID loop for the respective input value. The selection of appropriate P, I, and D values is a widely

researched area [146], with a number of techniques ranging in complexity that are suitable under different conditions. When simulations can be performed it has been shown that search based methods for identifying good PID values can be used [47, 120], however when compared to traditional tuning methods these take substantially more time to compute. Whilst there are more advanced methods of industrial control, such as model predictive control, these methods typically produce worse results than a naive PID approach for poorly understood models [19]. For this reason they will be omitted from this review.

2.6.2 Learning - NNs

In addition to reacting to network state it may be required to assess the current state of an individual mote, or multiple motes, and take action based on the result. A simple example of this would be location tracking of WSN devices based upon the known location of a number of other WSN devices, as envisaged for the assisted living scenario. This example requires a complex relationship between the messages that a node receives, the strength of the signals, and the location of the current device [9]. Basic linear classification could be performed to deduce a simple metric, however with a large amount of error in the results [71]. A more accurate method is to use neural networks [124] as these algorithms have the ability to approximate any smooth function between the input and output values, given a suitable number of neurons [55]. In this example the input values would be the RSSI messages sent from fixed-location nodes that have been received at the mobile node. The output of the NN could either be a location, such as a room, or more precise values such as coordinates.

For all learning-based algorithms an amount of training data is required to set up the algorithms, with greater amounts of data typically yielding greater accuracy from the algorithms. Not only is a large amount of data required, but training algorithms such as neural networks requires a large amount of computational power, making training of these algorithms expensive in terms of both memory space and computational power [161]. For these reasons it is envisaged that readings will be reported to the sink, which as specified earlier is typically a higher power device, which in turn can train the appropriate classifiers. Once the classifiers have been trained the configuration values can be transmitted to the leaf nodes allowing classification to be performed.

2.7 Time Synchronisation

For a number of protocols time synchronisation is required, this may be either global time [95], or a relative point in time common to a subset of devices [72]. Global time synchronisation requires that some definitive time source is defined as the reference start time. These are commonly measured either from the start time of a specific node, or from some external time source such as a networked sink node. The issue with WSNs is that there can be large latencies with communication, making time synchronisation difficult [122]. Another issue is the clock drift between devices due to the small manufacturing differences between crystals on each device, in the extreme case leading to time differences of 54ms every hours [153]. Other external factors such as power and temperature can also affect the speed of the clock [48, 167], however by much smaller margins than the original skew.

These differences can be compensated for by using clock skew compensation [138], measuring the relative differences and then compensating accordingly. This method however only prolongs the time to become desynchronised, with periodic re-synchronisation ultimately being required [143]. Depending on the complexity of the synchronisation algorithm, times accurate down to the microsecond can be ensured between devices [40].

2.8 Operating Systems

To support development for these WSN microcontrollers there are a number of WSN specific OSs that can be used. These OS' have a strong focus on minimising the size in both RAM and ROM of the OS to support the limited resources of the target architectures, whilst also focusing on communications, and the ability to interface with other hardware devices [45]. Within WSNs there are two commonly used operating systems which will be described in detail.

2.8.1 TinyOS

TinyOS [54] is an operating system developed in 2000, with a kernel which fits into 400 bytes of program space, and uses a variation of C called NesC to provide a modular approach to constructing applications. TinyOS provides a large number of components, which must be explicitly used by the end user's application, with interfaces between these components being connected together ('wired' in NesC terms) to achieve the desired behaviour. These connections provide the links between events which may occur, such as timers being fired

or message receive events, and the logic to process these events. This abstraction into components, along with the supporting NesC language allows the resulting binaries to be as small as possible in both code space and data space, as unused components can be omitted from the build unlike alternative operating systems [80]. TinyOS comes with support for a large number of common WSN platforms like Shimmer, TelosB and MicaZ devices, however it lacks support for newer devices such as the ARM-based OpenMote-CC2538. TinyOS also includes a number of common WSN MAC and routing protocols, such as LPL at the MAC level and DSDV, AODV, Directed Diffusion, and more for the routing protocols [81].

2.8.2 Contiki

Contiki [37] is another popular operating system used for mote-class devices which was released in 2002, with the core kernel taking 810 bytes of program space. Contiki differs from TinyOS in that standard C is used for programming the devices, lowering the learning curve required to begin using the platform [142]. Contiki uses the notion of protothreads which encapsulates the ideas of both multithreading and events. Multiple Applications can be defined, each of which is waiting on a set of events. When an event which is being waited upon occurs, one of the tasks waiting on the event is executed until it either returns to wait on events, or it actively yields, at which point the next waiting task, if any, is executed. Should a task have yielded instead of waiting on an event, it is repeatedly called in a round-robin manner with any other yielded tasks until all tasks wait upon an event. In this way cooperative multischeduling can occur, without the overheads of preemption [38].

Contiki also supports a large number of devices, like TinyOS, however it also has support for more recent WSN platforms like the ARM-Based CC2538. In addition to the large number of supported devices, Contiki also supports a variety of MAC protocols such as ContikiMAC, X-MAC, LPP, in addition to a number of Routing protocols such as Rime-Mesh (an AODV-like multihop protocol) and ContikiRPL [150] (IPv6 implementation).

For this work initial experiments will be conducted with TinyOS as there is existing expertise with this OS, however later experiments will be conducted with Contiki in order to use their implementation of the ContikiMAC protocol, and support of additional hardware platforms.

2.9 Simulators

There are a large variety of simulators that are either specifically developed for WSNs or simulators that are used for traditional wireless networks which have been extended to support WSNs. These can be summarised as follows [60].

Accurate simulators: These simulators aim to either simulate the internal workings of the devices as accurately as possible, and or the communications as accurately as possible [156]. Accurate execution can be provided by emulating the physical hardware and executing the actual binaries, or through targeted compilation of the original source code to a simulation target . Accurate radio communication can be simulated through the use of various radio-propagation models, which model the interactions between radio messages at different strengths and phases to calculate the effective message received at the destination node. These simulators tend to be computationally expensive and therefore only a limited number of devices can be analysed in a reasonable time frame [160].

Fast simulators: Another class of simulators exist for analysing the large-scale properties of WSNs, such as emergent behaviour, and therefore require that large numbers of devices can be simulated [170]. For this reason these simulators either abstract away some of the execution logic and/or simplify the radio communications model to reduce the computational complexity, allowing more devices to be simulated. A selection of the most popular simulators is presented below.

2.9.1 NS-2 & NS-3

Network Simulator 2 (NS-2) was traditionally developed to simulate wired networks, with the first release being in 1996, with the later support of 802.11 and 802.15.4 [166]. This simulator provides an event-based programming environment using a combination of TCL and C++ and is specifically designed for communications research. This focus allows the physical, MAC, Routing and application level to be defined as part of the simulation script, providing the ability to easily swap layer implementations. As part of this simulator an accurate two-ray-ground physical model is provided, which can be used in conjunction with the 802.15.4 MAC and AODV / DSDV routing layers. As the simulator is relatively old much documentation exists detailing how to set up experiments and how to analyse the results. This simulator also provides the Network Animator (NAM) which allows the simulation logs to be visualised. This allows rapid inspection of overall network behaviour, allowing easy identification and diagnosis of dropped packets.

2.9.2 TOSSIM

TinyOS has an associated simulator called TOSSIM [79] which allows TinyOS applications to be compiled for the simulated MicaZ platform. This compilation replaces some of the TinyOS components, such as the clock, with simulator implementations allowing the majority of the application to remain unchanged. Within TOSSIM the radio model is relatively simplistic and has two modes, simple and lossy. Simple radio assumes that the the radio PHY can be modelled as a logical OR of all signals within range, with lossy communications modelling the network as a graph, with edges between nodes containing a packet success rate. As with NS-2, simulations are controlled with scripts, in this case written in Python or C++, which allows the simulation to be controlled including the connectivity graph, injecting messages, and physical device interaction.

2.9.3 Cooja

The default installation of Contiki comes with its own simulator called Cooja [107] which is intended to be used for evaluating WSN applications. Cooja is a graphically controlled simulator (with some limited support for non-gui scripting) which allows the simulation to be easily set up and configured. The logic of the devices cannot be abstracted to the same levels as provided in NS-2, instead the program must be written either within Contiki and compiled for the simulation target, or a binary provided for the real WSN hardware, for example a TelosB binary. When the target is the Cooja simulator an instance of the binary is created for each of the nodes, with the simulator receiving or sending events to the instance depending on the appropriate radio traffic. In the case of a TelosB binary MPSIM is used to emulate the WSN processor, with the WSN radio being emulated as a simple state machine which in turn sends and receives data over the Cooja PHY layer [41]. The Cooja target ensures that the logic of the program remains the same on both the simulator and the physical devices, where the TelosB binaries also ensure that any target specific issues, such as overflow conditions, remain consistent in both the simulator and the physical devices. A number of PHY models are supported, such as the detailed two-ray-ground model, allowing the computational complexity of the simulation to be controlled as in NS-2.

2.10 Summary

The literature review has identified that WSNs are being considered for a variety of applications, however the issue of reliability is an ongoing concern. By viewing reliability as one component of dependability and analysing the remaining dependability attributes it has been identified that there is a lack of research into dependability within WSNs. When assessing the dependability attributes it can be seen that Safety, Integrity and Maintenance have minimal levels of literature in WSNs. Current works focus on Availability and Reliability, with this literature review identifying that there is a substantial amount of research into power saving within wireless sensor networks. These power savings are through many levels of the network stack, with the majority of works focusing on the lower MAC and Routing levels. There has been some level of work at the Application level, however there have only been minimal works looking into using application's soft real-time requirements to realise further power savings.

This thesis proposes taking an application-centric view to WSNs, with an application being defined in terms of both functional requirements and also timing requirements. By using this specification of an application this thesis proposes the following objectives:

Obj 1 - From an application-specification, a systematic method for ensuring the dependability of a specific WSN application should be derived. As the environment is dynamic and the devices themselves vary in performance over time, a method for monitoring dependability throughout the runtime of the system should also be defined.

Obj 2 - From the same application-specification, a protocol should be defined for improving the power savings of the WSN by taking an application-centric view of the network. This protocol should work with the current state-of-the art in MAC and Routing level protocols to ensure that savings are in addition to that provided by these protocols.

Obj 3 - To ensure that the previous objectives can be met not only for a particular application-specification, but for applications which may change their timing requirements at run-time in response to changes in the environment. This should also support the coexistence of multiple applications within the same network, each with possible variances in timing requirements.

Chapter 3

Dependability Assurance

As identified within the literature, there are many ways in which a WSN may fail due to a large number of possible factors which include external interference, environmental conditions such as temperature, or actions undertaken by external actors. These failures are difficult to predict off-line, with any such prediction being extremely pessimistic due to the inherent variability in the environment. To allow reliance on the information provided by these networks, online monitoring of the network must take place to identify and report, within reason, all possible forms of failure so that appropriate corrective actions can be taken. Not only must failures be detected, but also replacement of devices must be considered. All these issues are covered in the topic of Dependability, and therefore the Dependability of the WSN will be analysed.

3.1 Overview

As identified within the literature review there is minimal literature on analysis of WSN applications with respect to Dependability. The vast amount of literature focuses on providing generic solutions to availability and reliability that can support a wide variety of applications. Objective 1 of the literature review states that through the use of an application-specification which defines the functional specification in addition to the timing requirements of the application, focused analysis of the dependability of the application should be performed. Once this analysis has been performed, Objective 1 states that a method for ensuring that the dependability is met at run-time should be derived.

It is proposed that to ensure maximum coverage of possible failures, a systematic approach to analysing the dependability of the application is undertaken. This need for a

systematic approach is evident from the literature, as the few works which are relevant to dependability typically select only a single aspect of dependability upon which to focus. From the literature objective, in combination with the need for a method which ensures a good coverage of possible failures, the goals for this chapter are as follows.

DaObj 1 - Demonstrate a methodology for systematically analysing an application from its functional and timing requirements in order to derive a number of dependability requirements that must be met at run-time to ensure the application is dependable.

DaObj 2 - Demonstrate a method for transforming the run-time requirements into a number of tests which must be constantly evaluated at run-time, to ensure that the application is dependable even in the presence of changing environmental factors.

DaObj 3 - Ensure that the run-time tests do not adversely affect the performance of the network in terms of both network efficiency and power.

To perform the analysis this chapter will define a methodology for analysing applications which can be applied to a large variety of scenarios. This analysis can be used to systematically derive a number of requirements that must be checked at run-time to ensure that the dependability of the application is being met. The second half of this chapter will then follow this methodology and perform the analysis upon an example application, showing the derivation of the run-time requirements, leading to a worked example running on both simulation and physical platforms.

The target applications for this chapter are multiple source - single sink scenarios, as these represent the most common form of WSN deployment. Whilst this chapter focuses on this specific type of application, the methodology is generic enough to support other types of scenario. Secondly this chapter assumes that the application is event-triggered, and therefore does not send any data until an event of the specified type is identified. Similarly the methodology is not restricted to this type of application, as streaming-based applications which constantly report readings to the sink can be visualised as similar to event-triggered deployments when events are being simulated.

Dependability assurance will however only ensure that the application requirements can be met, or that a failure to meet these requirements can be successfully detected (in effect ensuring the system never fails-silent). Whilst DA shall ensure that the failures can

be detected, it is not required to provide enough information to diagnose the cause of the fault, which would require extra messages to be generated and cooperation between devices to take appropriate action. As failure identification is a related, but separate topic it will not be covered within this work, however further work may investigate extensions to support failure identification.

3.2 Method

3.2.1 Problem Definition

As the application in question is being evaluated with regards to dependability, it is important to address the individual dependability attributes, which are Availability, Reliability, Safety, Integrity, and Maintenance. To reduce the complexity of the problem, the devices being considered are assumed to be safe. This is firstly because they have no physical actuators and thus cannot affect the external environment in which they operate, eliminating the possibility of catastrophic consequences occurring as a direct consequence of the devices themselves. Secondly, the motes will be certified to the appropriate safety standards to ensure that they themselves cannot unintentionally cause harm. To further reduce complexity, it is assumed that radio communication integrity will be handled separately by other layers of the protocol stack such as the MAC layer, thereby simplifying the analysis to either successful or failed communication. This simplification is reasonable, as there is currently extensive active research in this area, with much work focusing on routing and MAC layer approaches. Finally, the integrity of the network will be assumed to be handled by other approaches as failures due to malicious activity are another active research topic.

This reduces the analysis to consideration of only the remaining three attributes, Availability, Reliability, and Maintainability. As these remaining attributes depend on the specific application they must be systematically analysed to derive the safety requirements that the resulting system must meet in order to be deemed dependable. In the case of DA the safety requirements do not only cover the safety of the system, but the dependability of the system. The term safety requirements has been kept to be consistent with the literature.

3.2.2 Deriving Safety Requirements

The first stage of Dependability Assurance (DA) is to identify, in a systematic manner, what parts of the application need monitoring in order to identify failures and schedule Maintenance accordingly. As identified within the literature review Section 2.5.1 there are a number of existing techniques that can be used to systematically analyse software, with the most suitable being Software Analysis and Resolution in Design (SHARD). As SHARD is being used the application must be decomposed into a number of communicating components, with an information flow connecting each component. As DA is not evaluating the software within each WSN, but the interactions between the software and the surrounding devices or actors (i.e. the operators of the network or the people being monitored), the smallest level of decomposition is the individual WSN, Sensor, or Actor. In this scenario the information flows are the exchange of data between nodes, any exchange between the nodes and external sensors, and exchange between a node and an actor (the network operators in this case). Once this decomposition has occurred the SHARD process can be applied. An outline of the SHARD process is given in Figure 3.1. As part of this process the application designer applies a set of 'guidewords' to each individual information flow between the individual components of the software, in order to analyse possible deviations from the intended behaviour of the application. Should any of these behaviours be unsatisfactory, these are noted as part of the SHARD process, and are known as hazards. The guidewords to apply as part of the process are given in Table 3.1. From these hazards, decisions can be made as to the potential disruptions to the application and the requirements that must be met to ensure that this hazard is unlikely to occur. These requirements are called derived safety requirements (DSRs), and are necessary to ensure correct operation of the application. [121].

Guideword	**Effect**	**Example Cause**
Omission	No Data	Battery Failure
Commission	Extra Data	Frequent External Stimulus
Early	Sent Early	Incorrect Time Synchronisation
Late	Sending Delayed	In-Network Delays
Value Subtle Incorrect	Data is slightly incorrect	Sensor ADC Drift
Value Coarse Incorrect	Data is largely incorrect	Corruption of Data

Table 3.1: SHARD Guidewords

Figure 3.2 shows the proposed methodology for DA. Initially the system must be split

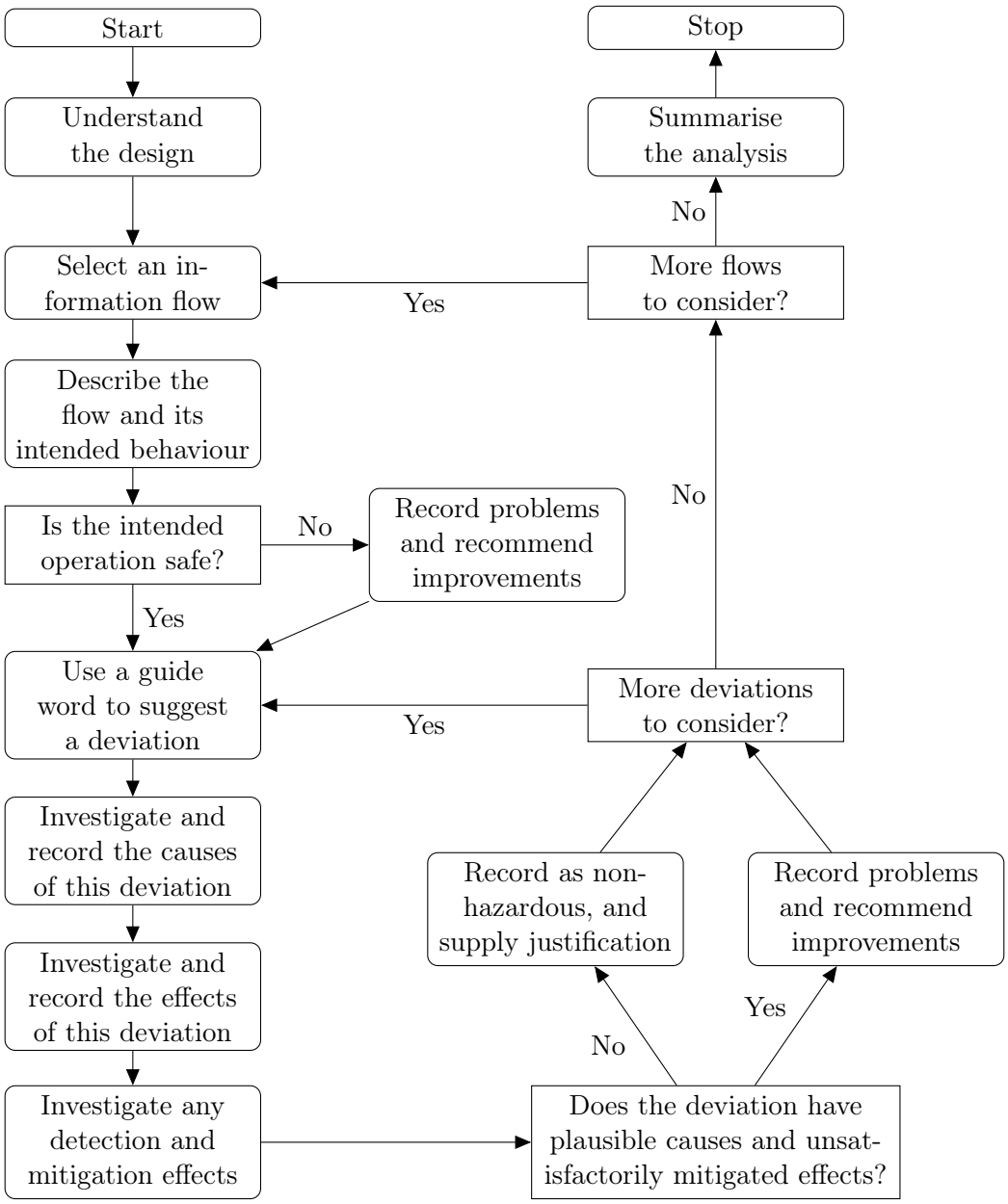

Figure 3.1: SHARD Process

to form components connected by information flows. These information flows must then be analysed using the SHARD keywords to generate a list of possible failure modes that may occur between each pair of communicating components. These failure modes are then turned into a number of safety requirements, the derived safety requirements (DSRs).

3.2.3 Defining Dependability Tests

Having generated a list of DSRs it must be ensured that they are met throughout the operational lifetime of the network. As identified earlier, however, the environment in

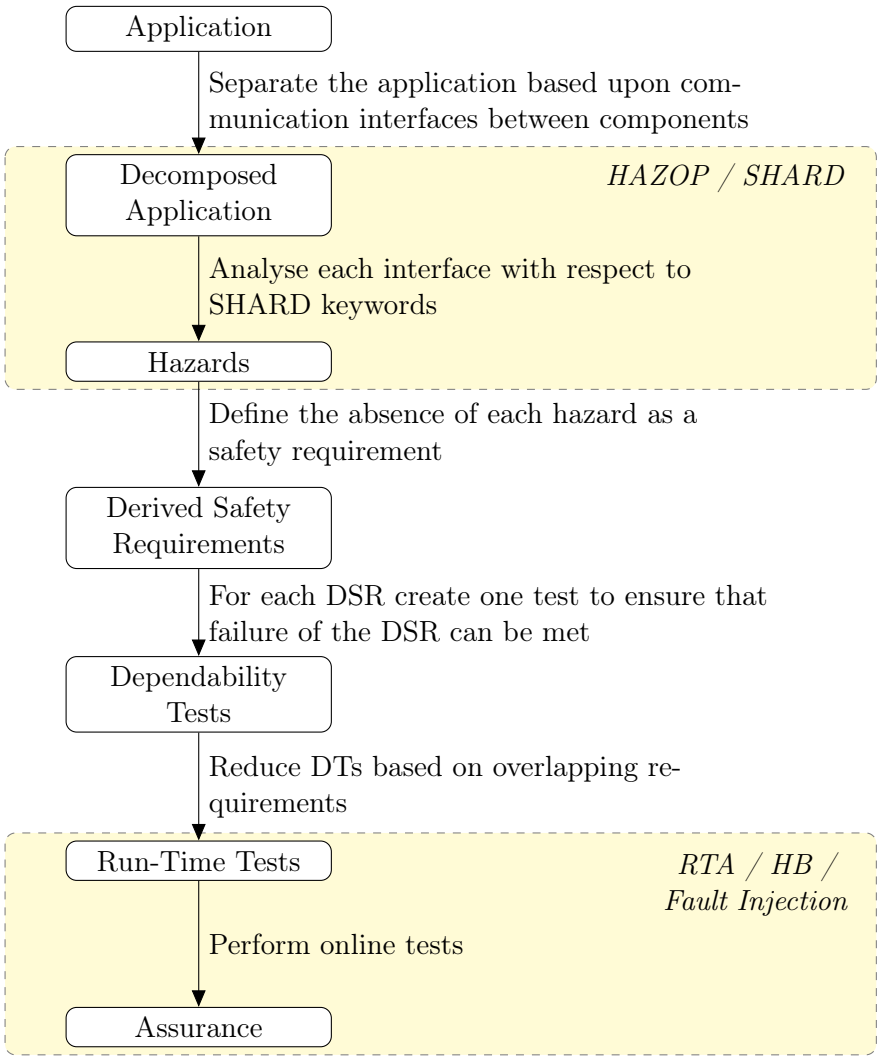

Figure 3.2: Dependability Assurance Process

which they operate is highly dynamic and thus offline analysis does not ensure that these requirements can be met. To ensure that the DSRs are met a number of Dependability Tests (DTs) are derived to ensure the status of the DSR at run-time, with each DSR having one associated DT. Each DT consists of two parts, a physical implementation which checks the DSR at runtime, and a run-time test which aims to violate the DSR to ensure that any such errors can be detected. Without the run-time test to violate the DSR no confidence can be obtained that an error will be detected. One example would be a DSR specifying that at no point should event X be detected. In this example the associated DT would be some logic on the sink node which raises an error if event X is detected, and a corresponding test which raises event X. As the DTs are specific to the individual DSRs, which in turn are application-specific, no single solution exists for all DTs. For this reasons the assessment of

the DTs may take many forms, such as assessing the quality of service provided, injecting failures into the network, or halting specific communications to ascertain the system's responsiveness.

As identified within the literature review, there are a number of systems that can be used to monitor the state of a WSN at run-time. These vary from basic health-monitoring (HM) solutions to simulated applications (RTA). In addition to monitoring the application, there are a number of approaches that inject faults into the network, however these are typically performed on an ad-hoc basis. By injecting faults and monitoring the response of the system, valuable dependability data can be collected, however its usefulness relies on systematically identifying the correct failure modes. DA aims to use systematic fault injection, together with simulated events, based on requirements derived from HAZOP. By providing monitoring of a wide range of WSN failure modes, faults can be detected as and when they occur and maintenance scheduled. This focus on Maintainability is the key to maintaining the underlying Availability and Reliability of the network throughout the application's lifetime.

Once DTs have been defined for all DSRs these must be periodically executed on the network to ensure the DSRs are being met. This periodic execution of tests however represents wasted network resources, as battery power and bandwidth are consumed in running the tests which could be used for supporting the primary application for a longer period of time. To reduce these overheads the DTs are analysed to identify overlaps, allowing these tests to be reduced into a single test, which covers multiple DSRs. This is commonly the case with timing requirements, as a single message may be used to check multiple timing requirements (i.e. messages that take no longer than X to arrive can be checked on any existing data packets from the sink). Reductions can also be achieved where the data required by a DT is already being supplied by the application within its normal operation. This allows for the conditions of a DSR to be satisfied without the need to request additional data from the network. Once a reduction of all the tests has occurred, the final set of tests must be run to ensure the original DSRs are being met.

3.3 Case Study

3.3.1 Dependability Assurance

To validate Dependability Assurance, an initial event detection system based upon static devices around the home will be investigated. Having analysed a generic event detection application, this will be refined in Section 3.3.2 where a fire detection system is considered.

3.3.1.1 HAZOP

Initially it is important to perform analysis of the proposed system, identifying the system components that communicate, and then analysing these communication channels in a systematic manner using HAZOP. Figure 3.3 shows a simplified view of the four main components of the system and the communications between them: the sensors reporting readings to the device; the device where readings are processed and analysed, possibly sending event information to the sink; the base station where event readings are collected and decisions to raise alerts are taken; and finally the operators who receive system alerts and take appropriate action. These four components have three clear communication channels to be analysed: readings from the sensors arriving at the mote; events sent from the mote over the multi-hop network to the sink device; and alerts raised at the sink being reported to the operators. It is important to note that the communications, whilst being shown as a single hop, may be over multiple hops across the network.

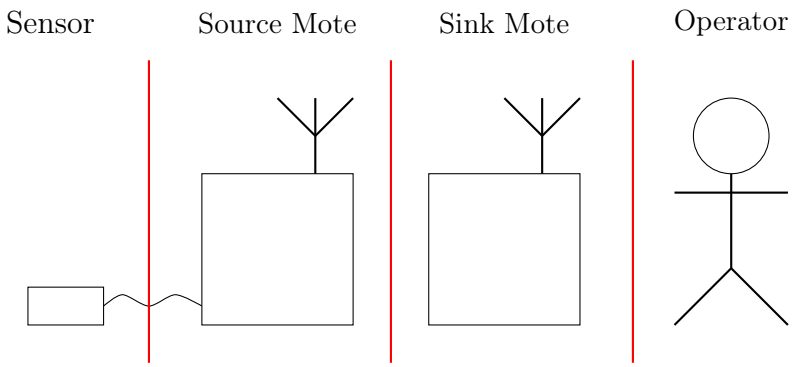

Figure 3.3: Simplified view of end-to-end WSN data flow.

SHARD

From Figure 3.3 the three communication channels are analysed using the SHARD keywords identified earlier. The results are given for the sensor communication in Table 3.2, the radio communications in Table 3.3, and the operator communication in Table 3.4.

It is important to note that this is not an assessment of how to prevent the devices failing, as this is outside of the operators control, instead it analyses how failures may be detected and what actions could be taken. Therefore any processes that improve the quality of the components are outside the scope of this work.

Guide-word	**Meaning**	**Hazard**	**Mitigation**
Omission	Sensor fails to report data	Event will go undetected	Easily detectable by host device, raise notification
Commis-sion	Sensor is reporting too much data	Possible overload of receiving device	Easily detectable by host device, disable sensor or call for urgent maintenance
Early	Data is sent from the sensor before required	Value may be out-of-date when used	Re-request the data, compensate for time differential
Late	Data is sent from the sensor after required	Events may be detected later than required	Raise alert, faulty sensor, requires maintenance
Value Subtle Incorrect	Data is slightly incorrect	Un-noticed if subtle, events should still be detected correctly	Generally undetectable
Value Coarse Incorrect	Data is largely incorrect	Events may be falsely detected (False Positive)	Replace faulty sensor

Table 3.2: SHARD Guide words applied to sensor readings

Guide-word	**Meaning**	**Hazard**	**Mitigation**
Omission	Radio fails to send message	Event detected but cannot be reported to other devices	indistinguishable from lack of events, periodic message to test radio correctness
Commis-sion	Device sends messages even when no detected event	Possible overload of communications network or receiver	Detectable, flag device as failed, isolate if possible, call for maintenance
Early	Event is reported before it is detected	Cannot occur (is commission if event is never detected)	N/A
Late	Message reporting event arrives late	Could be too late to respond to event	Specify a maximum delay for messages, check / enforce this at run-time
Value Subtle Incorrect	Discrete Events, cannot occur	N/A	N/A
Value Coarse Incorrect	Discrete Events, cannot occur	N/A	N/A

Table 3.3: SHARD Guide words applied to communication between devices

The hazard and mitigation columns from the three tables are finally collated in Table 3.5 to provide a final list of safety requirements, which mitigate the possible hazards. These

Guide-word	Meaning	Hazard	Mitigation
Omission	Base station fails to inform operators despite receiving the message	Event will go undetected	Undetectable. Traditional safety approaches to base station software
Commis-sion	Sensor is frequently reporting false positives	Operators may begin to ignore some readings	Cross-check reported fires between multiple devices in the same location
Early	Event is reported before it occurs	Not possible	N/A
Late	Event is raised much later than detected	Events may be acted to later than required	Delay would be on the base station only, and thus unlikely.
Value Subtle Incorrect	Raised events when close to threshold	Likely event is about to occur	Acceptable
Value Coarse Incorrect	Incorrectly reported events (False Positives)	Operators may begin to ignore some readings	Flag faulty nodes and disregard messages

Table 3.4: SHARD Guide words applied to operator interactions

will be referred to as the derived safety requirements (DSRs).

Hazard	Safety Requirement
Failure of nodes may lead to events occurring without detection.	Events must be guaranteed to be detected as long as Y nodes are operational.
Events are detected too late for the operators to react accordingly.	The WSN must ensure that an event is reported within X time.
Failed sensors may raise false events, reducing the advantage provided by the WSN	Failed sensors must be reported to the operators for maintenance.
Interference from external sources may raise false events or cause events to go unreported.	The integrity of the messages must be maintained throughout the WSN.
Failure of the system would cause no events to be detected.	Failure of the WSN must be reported to the operators so appropriate action can be taken.

Table 3.5: Derived Hazards and Safety Requirements

3.3.1.2 DSRs

DSR	Description
1	Detect event when greater than Y nodes are operational
2	An event is reported within X seconds
3	Larger errors and implausible values from sensors are detected
4	Network is tolerant to anomalies
5	Monitoring failure is detected within W seconds

Table 3.6: Summary of DSRs

From these five DSRs the DA process in continued by producing five DTs, one for each DSR.

The first DT to be reduced is DT4. This tests that the network is tolerant to anomalies. By making the assumption that the underlying stack will make a best-effort attempt to send data to the destination and in the process perform checks on the integrity of the data through the use of cyclic redundancy checks (CRCs), (as required to conform to the 802.15.4 specification), it is assumed that the communication will either succeed or fail. With this assumption, successful communication has encountered no anomalies and failed transmissions are not initially an issue as long as DT1 holds, that is, an event is detected when there are greater than Y nodes available.

The second reduction is the coverage of DT2 by DT1 due to the realisation that the timing of the event detection can be checked using the same data generated by the checks for application correctness. These two reductions leave the application with three DTs that must be checked at runtime as shown in Table 3.7. An overview of the process is given in Figure 3.4.

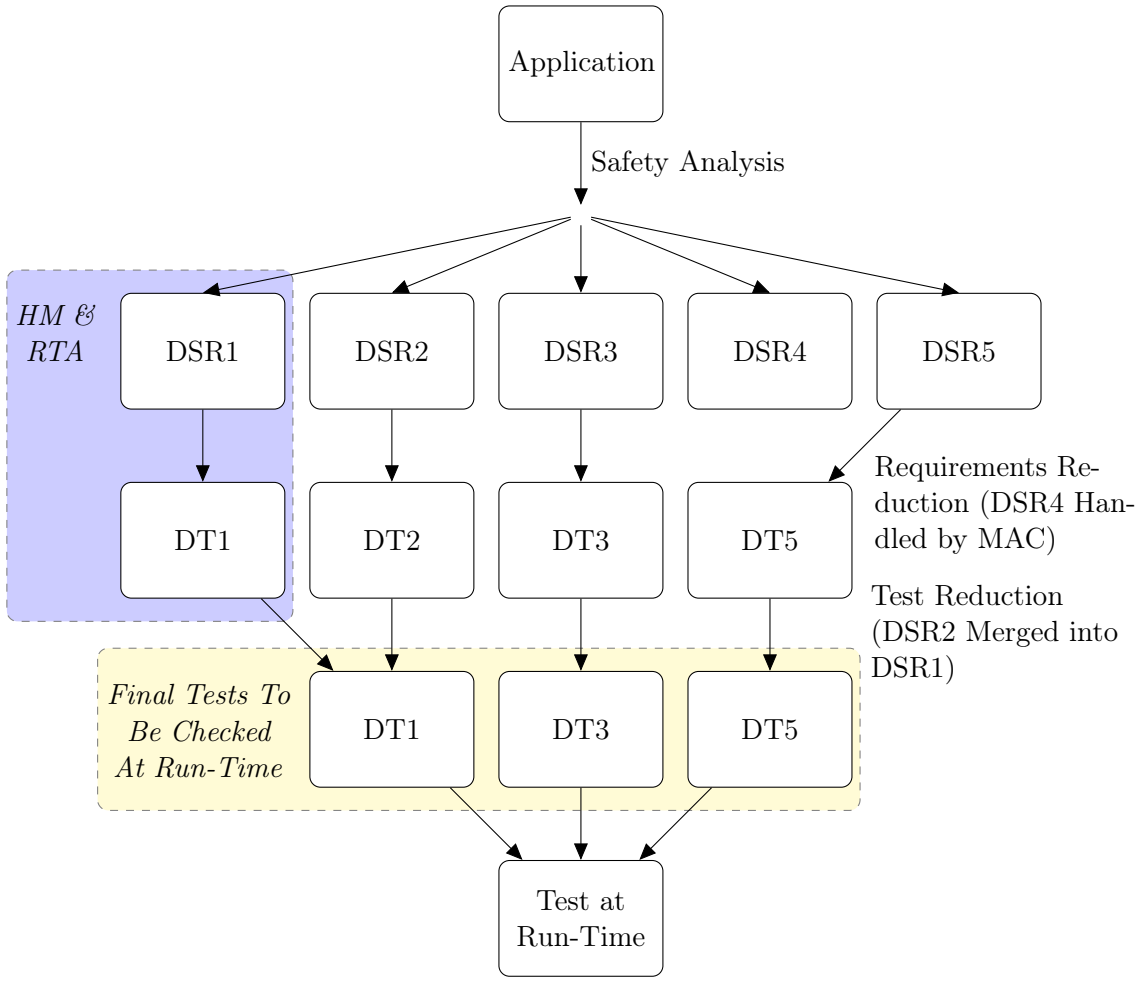

Figure 3.4: Overview of DA applied to the fire detection scenario

DT	Description
1	Detect event when greater than Y nodes are operational within X seconds
3	Larger errors and implausible values from sensors are detected
5	Monitoring failure is detected within W seconds

Table 3.7: Summary of reduced DTs

When these tests are being executed at run-time, a number of actions can be performed based upon the DT that has failed. Figure 3.5 shows the combination of actions that may occur and the resulting outcome.

3.3.2 Fire Detection

Initially, to validate Dependability Assurance, a fire detection system is envisaged based upon the static devices around the home. A fire detection system has been chosen as it is the same building fire detection application as proposed by Wu et al [163]. By choosing the same application as proposed by Wu et al a direct comparison to Run-time Assurance RTA can be performed under the same scenario. Another advantage is that the TelosB motes used for the physical testing also contain a temperature sensor which avoids the need to simulate real readings. As Section 3.3.1 describes a generic event detection system a specific example must be chosen for the remainder of this chapter. In this case a fire detection example has been chosen, with the event being a detected fire. The proposed system consists of four rooms, with each room containing three nodes. To detect a fire, each mote has a single temperature sensor, which records temperatures over a set threshold (arbitrarily set to 100℃ in our trials) which then sends a fire alert to the sink node within 1 minute. If any fire alerts are raised on the sink node, then this is an indication of a fire and the appropriate action taken, such as evacuating the building. From this definition, we can see that Y is defined to be 1 and X is 1 minute. The value for W is not defined, and therefore as 5 minutes has been deemed acceptable for the fire detection deadline, it will also be used for this variable. To further appreciate how the DTs cover the original 5 DSRs, each DSR will be analysed in turn with direct reference to the fire detection system.

3.3.2.1 DSR1 - Detect event when greater than Y nodes are operational

This DSR is concerned with the standard operation of the fire detection system and states that provided there is at least one node within a room then any fires should be detected. This is performed though the use of DT1 which simulates fires within a room. It is

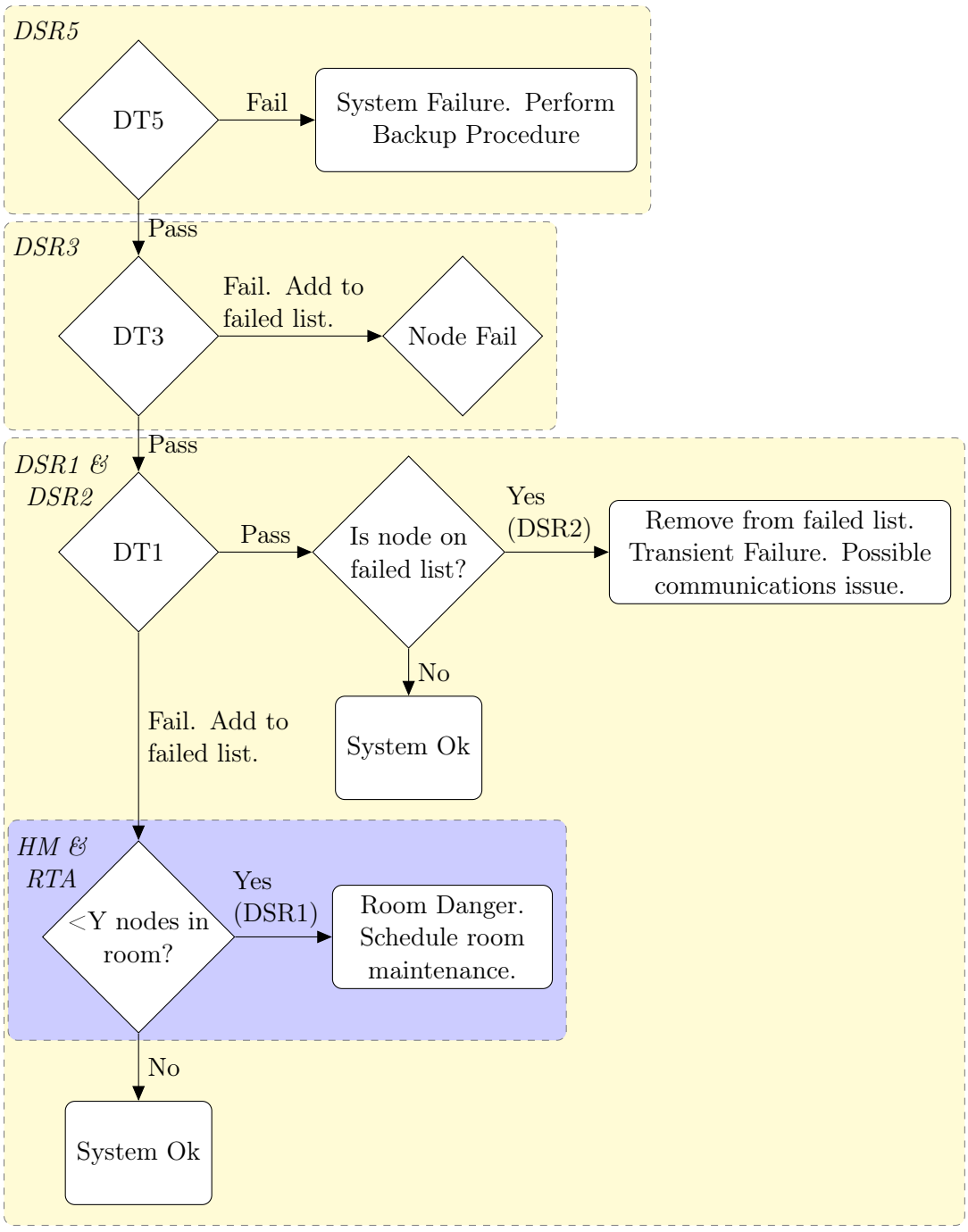

Figure 3.5: Detected failures and the resultant actions

important that the sensor readings are simulated instead of merely injecting packets at the radio layer to fully test the software on the motes themselves. To simulate fires as accurately as possible the values returned by the sensor are intercepted, in order to return values higher than our 100 °C threshold. To differentiate between real and simulated fires, an additional *sim* flag is appended to all fire messages, real or simulated, to avoid any

possible differentiation at lower network levels. This flag is simply set to the motes current *sim* flag.

3.3.2.2 DSR2 - An event is reported within X seconds

The timeliness of fire detection is important and as such this DSR ensures that when a fire has been detected it will be reported to the base station within 1 minute for our example. This DSR would require generation of additional packets as the event-driven nature of our application provides no data to measure, however as identified earlier DSR1 generates such additional data that can be used. Therefore the only requirements for DT2 are that the generated data contains the timestamp when the detection occurred and is enforced by checking that this timestamp does not exceed the 1 minute deadline when it is received at the sink node.

3.3.2.3 DSR3 - Larger errors and implausible values from sensors are detected

Within traditional embedded development devices can be designed to fail in a predictable manner such as full-scale-deflection or may simply return random values. Full-scale deflection allows for simple detection of failed devices using a simple threshold mechanism. The advantage of thresholding to detect failures are that failures returning random data will also be detected by the same system. To check that this mode is indeed detected additional messages are generated in the same manner as DT1 generates messages, instead returning erroneous readings. Further checks are performed to detect stuck-at faults, as the precision of the temperature readings on our target hardware is very high, causing constant fluctuations in the lowest precision of the readings. Should any sensor readings remain exactly the same for over x seconds then this is identified as a stuck-at fault, and reported as a DSR3 failure. Failures constantly returning plausible readings cannot be detected by a single test and must instead be covered through cross-checking readings between devices in the same spatial region. Care needs to be taken to understand the possible differential that can be allowed between devices as a fire in one region of the room may cause a large difference between motes in the same room. As this failure mode is unlikely we do not check for this form of error.

3.3.2.4 DSR4 - Network is tolerant to anomalies

This DSR is not tested for as it is assumed to be handled by other areas of the network stack or to be detected as a different form of error. Modification of radio data from sources such as external interference are assumed to be handled by the lower levels of the network stack in the form of CRC checks with the resultant packet being dropped. Dropped packets, and the blocking of transmitted packets by severe interference are handled in the same way, they are detected by DT1 and detected as a DSR1 failure. The only difference between a full mote failure, as handled by DSR1, and interference, is the possibility that interference may be transient. For this reason when such motes become functional, they should be noted as a caution on the operators console. Direct attack by an adversary is not handled by this DSR as such attacks vary wildly and are a research field in their own right.

3.3.2.5 DSR5 - Monitoring failure is detected within W seconds

To detect complete failure of the monitoring system DSR5 must be checked through the use of DT5. This DT is provided by checking that any DTs have been received at the operators console in the last minute. Should no DTs arrive then a failure of DT5 is raised. This DT does not generate data, but instead halts all other DTs for over a minute, catching the DT5 failure, passing the test. This ensures that in the worst-case scenario where DT messages are failing to be received, indicating possible total communications failure or other wide-spread failures will be detected and appropriate action taken. Total failure should never occur as maintenance should maintain a level of service for our scenario, but if it was to occur then complete building evacuation would be a possible cause of action.

3.3.2.6 Failure Modes

From these DSRs there are a number of failures that can be detected as follows.

Permanent Failure DT1 detects when individual motes have failed, and as such permanently failed motes will continue to raise an error until they are replaced. The cause of the permanent failure is not revealed and as such could be due to power failure, hardware failure or permanent communications failure.

Transient Failure Should a device which has failed a DT1 test successfully pass DT1 later in the systems execution, without being replaced, this is an indication of transient failures. These failures are commonly due to interference and could still cause

a maintenance request if they become too severe.

Total Failure Total failure of the system can be detected by DT5, indicating that the WSN can no longer function. This could be due to all devices failing (if no maintenance has been performed), or by communications to the sink being blocked, Total failure cannot be fixed easily and as such in our example the appropriate action would be the evacuation of the building until the cause has been found.

In our example there is only one sink node from which the operators function. Ideally there would be multiple sink nodes spatially separated around the network allowing for communications failures around one of the sinks without a total failure being raised. These spatially separated nodes would be hard-wired in terms of power but also for communications to assess the risk level in a specific room.

3.3.2.7 Time Synchronisation

As DTs need to be executed by all motes in the network at similar times, some form of time synchronisation is necessary. Primarily DT5 requires that all messages be suppressed at the same time. This time synchronisation does not need to be precise as it is only required to allow tests to be scheduled within the same minute, and as such synchronisation to within a second would be ample. To meet this requirement a variation on Cristian's algorithm is used to provide basic synchronisation between the devices, with nodes performing synchronisation upon joining the network. When powering up, motes synchronise to the base station mote's time, which is set from the hard-wired device. Synchronisation is performed on an iterative basis until the time difference between the new mote and the base station is less than one second, which is deemed acceptable. The benefits of performing such loose time synchronisation is given in Section 3.3.3.2, with no additional benefit being provided by increasing the precision. As identified in the literature review Section 2.7 the time within each mote is prone to drift due to manufacturing defects. For this reason time synchronisation is repeated when the timestamps on received messages are more than 10s different than the base node.

This algorithm relies on the notion of Round-Trip-Time (RTT), measured by sending a packet to the source, which is then echoed back immediately. The time difference between the packet being sent from the source, to the echo being received at the source is recorded to calculate the RTT. This RTT is halved to naively calculate the uni-directional delay.

When the base node replies to the source it also includes a timestamp for the source to use as its local time, which is calculated as its current source time plus the uni-directional delay. This algorithm is extended to include a final time acknowledgement from the source device. This final acknowledgement is checked for accuracy by removing the stored uni-directional delay, and should the value be outside of acceptable bounds, possibly due to routing or MAC issues, the time synchronisation process is repeated.

This approach allows for devices which have been replaced, due to maintenance, to directly request time synchronisation when they re-join the network, and is used for all work within this thesis. Flooding-based time synchronisation algorithms could be used, however these would require the new device to wait for the next time time-flood. Alternatively, if devices are allowed to request time-floods, this could become expensive if many nodes are replaced, or the network size is especially large.

3.3.3 Evaluation

In this evaluation we will check that the DTs that have been defined are enough to meet the DSRs. This will involve both simulated tests and physical test where appropriate. To ensure that the tests are performed fairly the base station logic was programmed for interoperability between both the simulated motes and the physical motes.

For simulations NS-2 was chosen as it allows for larger-scale and faster simulations than Cooja or Tossim, with some loss in fidelity. This scale and speed was required for two main reasons. Primarily, speed is required for the long-duration tests of mote failures and mote replacement, which requires simulations to run over many simulated months. Secondly the scale was required to assess the theoretical limits to the number of motes that may cause timing issues when many motes transmit in the same location. The intuition behind timing issues with many motes in the same proximity is due to carrier sense causing larger backoffs between motes, possibly causing our timing constraints to be invalidated. NS-2 was configured for a typical WSN application, with 802.15.4 as the MAC layer and the in-built AODV used as the routing protocol. Physical testing was performed using TelosB motes with TinyOS 2.1.1. TinyOS's default MAC layer was used with NST-AODV as the routing layer.

The layout for the simulated devices was a row of rooms, with each room being within communication distance of the neighbouring rooms. The connectivity between rooms can be seen in Figure 3.6, where the devices are WSN nodes and the dashed line represents

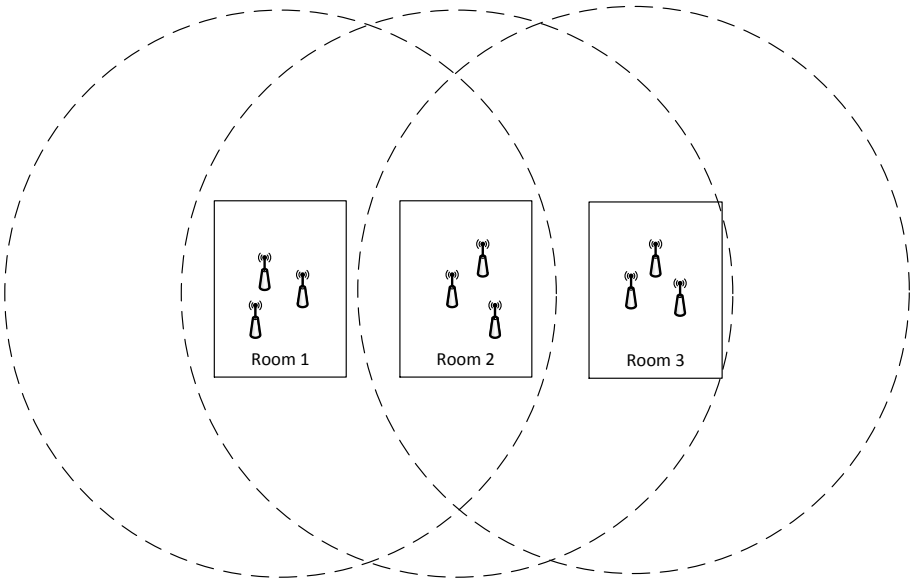

Figure 3.6: Layout of nodes in the simulations.

the transmission distance of the nodes. There are four rooms, but one has been omitted from the diagram for conciseness. For the physical experiments the closest to this layout we could achieve is shown in Figure 3.7. In this image, the nodes are the same purple icon as in the simulation layout. The transmission distances cannot be easily calculated and so are not shown on this diagram. However, it was checked that nodes within each cluster can communicate with the neighbouring clusters.

Both the simulated and the physical motes communicate to the same operator-console software, through the use of co-simulation for NS-2 and through a serial bridge device for the physical motes. This ensures that the management and detection of DTs is common between all the experiments, improving consistency. All of the simulator code for the source devices was written in Python, allowing automatic generation of data packet handling code for the physical devices and requiring only the data processing stubs to be written in C. The layout of the devices for both types of experiment will be covered in the appropriate sections.

For the evaluation, comparison will be made between this solution and Heartbeat (HB) and Run-time Assurance (RTA) which are representative of the two alternative approaches to health-monitoring within WSNs. HB will be implemented as a simple periodic broadcast, with the same frequency as the DA run tests. If any device receives a HB message, it simply records the senders address, if it does not already exist, and then increments the received HM message count from that mote. When a mote sends a HB message, it checks to see if

Figure 3.7: Layout of nodes in the physical experiments.

any of its counts for neighbouring devices is 0, in which case the neighbour has failed to send a message and is reported for maintenance. RTA is implemented by motes sending a HM message to the sink device every HM period. When the base node receives a HM message it looks up the room that the mote is assigned to and increments a counter for that room. If the base motes HM period is reached and any room has a counter of 0, all devices in the room are scheduled for maintenance.

3.3.3.1 DSR1

To test that failures can be successfully detected, maintenance requested and nodes repaired in a timely manner, we will evaluate the fire detection system under both simulation and

physical tests with motes failing at random. To generate failures of the devices, failure times was picked from the survival function. The survival function, or reliability function, returns the probability that a system will survive beyond a specific time given a mean and variation. To generate a failure time the inverse survival function is used to map a uniformly random real number between 0 and 1 to a specific time at which the WSN mote will fail. It is important to pick the failure times from a distribution that is similar to those experienced in real life, as the wrong distribution, such as uniform, will remove the chance of any common failure times between devices, providing the most optimistic failure distribution. By picking a low deviation, a pessimistic scenario is created where all devices fail at similar times, which has been demonstrated to be the case when common factors such as batteries are included. For this reason a pessimistic set of values is initially chosen, with a mean of 30 days and a deviation of 3 days for the failures. HM is performed every 6 hours, with maintenance occurring every week.

The deployments were constructed so that four rooms, each containing three nodes, could only communicate to neighbouring rooms. This was achieved in the simulator by ensuring that each node of three devices were only in range of the neighbouring groups. For the physical experiment, the connectivity of rooms was measured empirically before selecting rooms that met these criteria. The selected rooms from within the building can be seen in Figure 3.7. For the simulation, the base node was chosen to be in the end room, whilst room CSE/123 was chosen for the physical test as it contained the base station PC. All three of the health monitoring systems, HB, RTA, and DA, were configured to ensure that all rooms are checked simultaneously at the specified HM period. This ensures that all algorithms should provide the same level of coverage when all nodes are alive in all the rooms, with no particular algorithm having an advantage by checking only a subset of devices. The underlying routing and MAC algorithms also remains the same for all three HM systems. In this section the relative performance of HB, RTA and DA is dependant on the individual parameters such as the mean time between failures and the monitoring period. The relationship between these parameters, the time at risk, and the number of maintenance requests are studied in more detail within Figures 3.8-3.11 and the accompanying descriptions.

The criteria used to measure the performance of each approach is as follows.

Number of Packets This is the performance criteria most commonly found within the literature. Whilst it provides some emphasis on efficiency of the HM systems it

does however have a side-effect, namely that it also favours networks which allow a large number of failures to remain failed as these devices no longer contribute to the message count. For this reason we include two additional criteria, Number of Maintenance Requests and the Time at Risk.

Number of Maintenance Requests As a maintenance request requires a person to enter the building and replace the failed devices it is one of the most expensive parts of the system and therefore needs to be minimised. A single maintenance request is the replacement of all failed devices, and a count of how many devices were replaced, and since the failure rate for all devices is the same, should remain constant. Due to the cost, this is seen as the second most important factor of the HM system, with the most important being time at risk.

Time at Risk Time at risk is the cumulative time that any rooms have been left with no devices covering the room, potentially leading to an undetected fire if one was to occur in the same period. Individual rooms all contribute to the total time at risk, and therefore we need to reduce this factor as much as possible. Time at risk does not distinguish between rooms which have failed and are awaiting maintenance, and failed rooms which have not been detected.

Simulated Results

The first experiments were run under NS-2 and set to simulate 1/2 a years worth of runtime. This duration was chosen so a full experiment would run in 34 minutes of runtime. Tests were repeated 10 times for each of the three HM types. Table 3.8 lists the average of all 10 repeated runs.

Node Type	**Time At Risk (Min)**	**# Maintenance Requests**	**# Messages**
HB	15	58	61275
RTA	7955	16	43916
DA	2198	23	46700

Table 3.8: Overview of the three HM systems in the simulated deployment showing the time rooms were at risk, the amount of maintenance and the number of failures.

These results show that the lowest time at risk was the HB approach, however this is because it also had the highest number of maintenance requests with 3.6x and 2.5x more requests than RTA and DA. This is due to HB tolerating no failures, and therefore having the largest number of active devices at a given time. This is a limitation of HB as the

lack of application knowledge means that tolerances to a set number of failures cannot be easily added whilst keeping coverage within given rooms. As RTA only replaces devices once a room can no longer be monitored it has a highest time at risk, as the room must be at risk before replacements can be made. HB also uses 31%-40% more messages than the other two approaches due to the repeated broadcasting of messages to assess the health of the neighbouring devices. RTA and DA are similar in number of messages, however DA takes 44% more maintenance requests for a 262% reduction in time at risk. This higher number of maintenance requests is due to early call for maintenance before the room is at risk, causing the reduction in the time at risk.

Physical Results

To verify the trends identified in the simulations, physical experiments were conducted on the same scenario. As 1/2 a year's worth of run-time is unreasonable, time acceleration has been used on the physical devices. Reducing the times between executing events by 800 times reduces the period of HM tests from 6 hours to every 27 seconds, and maintenance from weekly to every 12.6 minutes. This acceleration allows all tests to be completed in 5.5 hours, removing any battery power issues. A disadvantage of acceleration is the increased likelihood of collisions between packets, as devices may transmit at similar times. This introduces pessimism into the system, however with dependability it is prudent to be conservative rather than optimistic. Table 3.9 shows the results of the experiment for each of the HM modes.

	Time at risk (Min)	**# Maintenance requests**	**# Total Failures**
HB	23806	72	47
RTA	63996	52	45
DA	31532	65	45

Table 3.9: Overview of the three HM systems in the physical deployment, showing the time rooms were at risk, the amount of maintenance and the number of failures.

These results clearly show that the physical experiments dropped substantially more messages across the board, however the same trend between the three approaches from the simulations still holds. This higher level of time at risk is most likely due to the increased interference experienced in the physical deployment which in turn causes more HM messages to be dropped, creating a higher time at risk. A higher density deployment with multiple paths to the sink may help to reduce the high rate of dropped messages. In these results it can be seen that HB takes the highest number of maintenance requests for

the lowest time at risk as it maintains the highest number of active devices. RTA and DA perform less maintenance requests but with higher times at risk. RTA takes 103% more time at risk than DA with only a 25% less maintenance, similar to the simulated results.

Parameter Tests

To assess how changes in the failure rate of nodes and the HM frequency affect the ability of the application to detect fires, a number of tests were performed. To avoid penalising applications which may fail for a large number of small periods, which may be safer than an application which fails for one large period of time, it was decided to introduce the notion of an undetected fire. An undetected fire is when a randomly selected room experiences a fire event, which occurs for a duration of 10 minutes. All nodes sample their sensors every 5 minutes, and if a fire is detected the fire event is removed. If any fires remain active after their 10 minutes they are simply removed and recorded as an undetected fire. Each experiment is run for 2 simulation years.

Mean Time Between Failures

To assess the affect of the mean time between failures on the detection rate the mean time between failures was varied from 0 to 24 months in 1 week increments. HM is performed every week and maintenance performed directly afterwards.

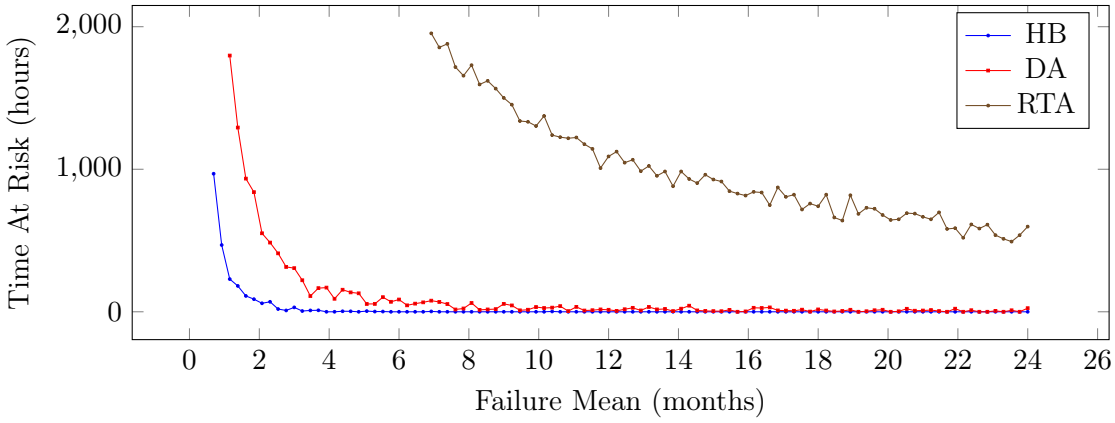

Figure 3.8: Failure Mean vs Time at risk

Figure 3.8 shows how DA performs closer to HB than RTA, with this trend getting stronger as the number of failures is reduced. The performance of the two approaches is almost identical when the mean time between failures is greater than 10 months. Figure 3.9 shows the number of maintenance requests for the same experiments in Figure 3.8. This figure shows that where HB and DA are similar in detection rate, DA is performing with substantially less maintenance requests. Therefore we can deduce that if the failure

rate is relatively low DA can dominate both HB in terms of efficiency and RTA in terms of detection rate.

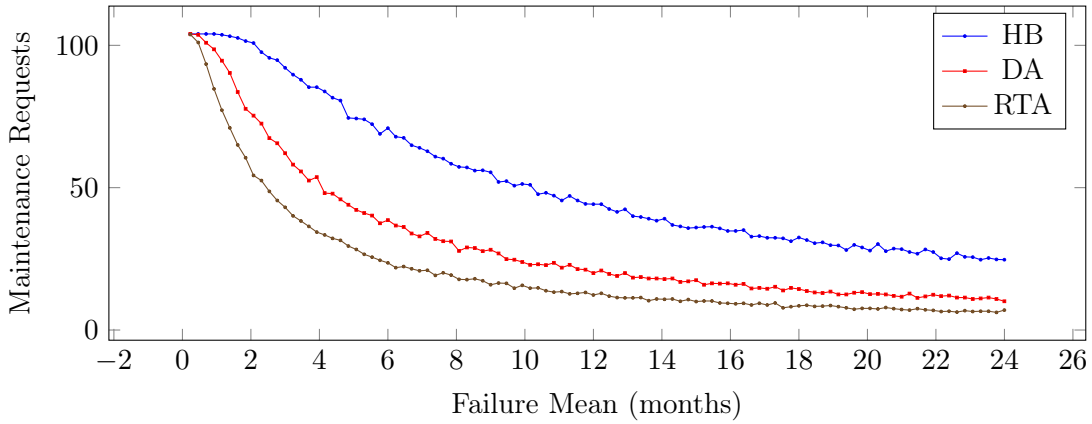

Figure 3.9: Node failure mean vs Maintenance Requests

Length of HM period

In these experiments the monitoring period is varied between 1 to 35 hours in increments of half hours , with the mean time between failures fixed to 6 months, with the experiment simulating 2 years. Figure 3.10 shows the results of these experiments.

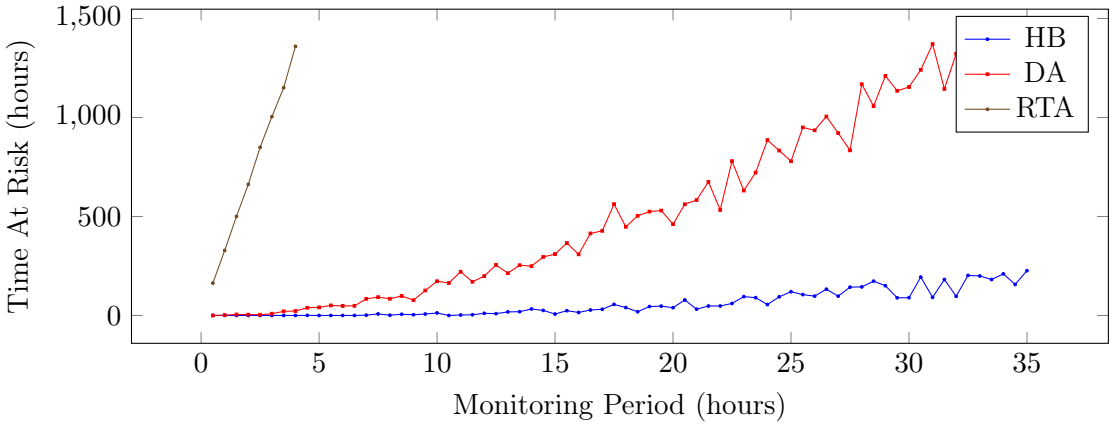

Figure 3.10: Time At Risk vs Monitoring period

As the HM period increases failed nodes are detected less quickly, causing the time at which there may be a fire in a room to increase, and thus the number of potentially undetected fires increases. The graph shows that DA performs in a similar way when compared to HB, with this similarity becoming increasingly clear as mean failure times increase. Figure 3.11 shows how the number of requests drastically drops as the monitoring period is increased.

To check that the identified trends were correct a large simulation was performed. The HM period was chosen to be 24 hours, with maintenance occurring 24 hours later. A larger

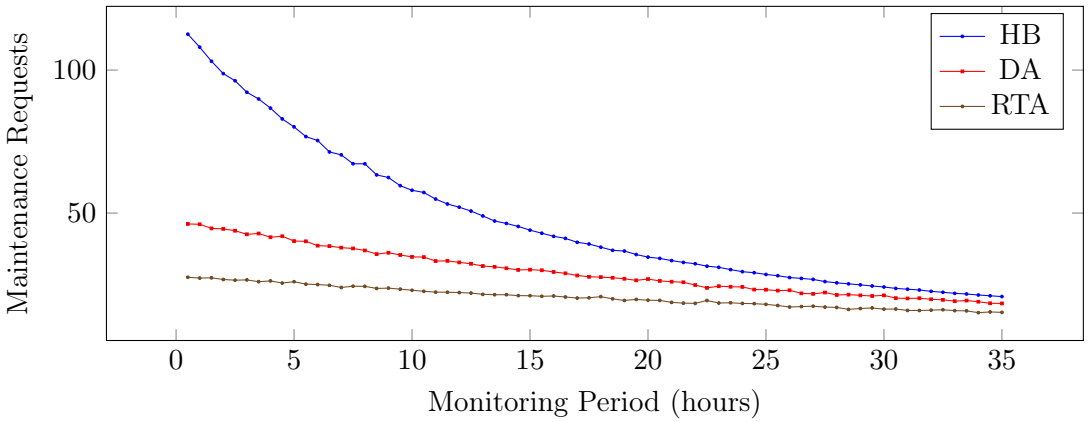

Figure 3.11: Maintenance request vs Maintenance period

mean lifetime between the mote failures was chosen, with the inverse survival function being used with a mean of 3 months and a deviation of 2 months. These values were chosen to be more realistic whilst still being suitably pessimistic. From these parameters a simulation was run for over a day of real-time, providing 100,000 years of simulation time, with the results being shown in Figure 3.12.

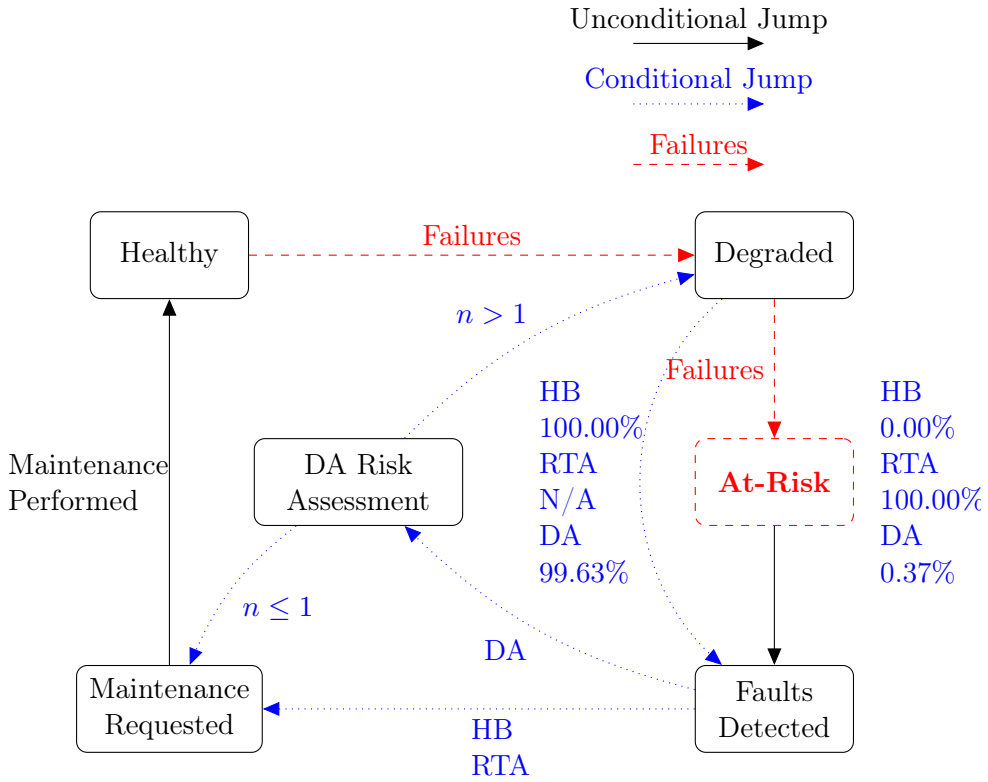

Figure 3.12: Long term simulation of failures.

This diagram gives an overview as to the three HM systems and how they compare when looking at the probability of entering set states. Throughout the entire experiment HB never entered the At-Risk state, in which a fire, if it occured, would not be detected.

DA only had a 0.37% chance of being at-risk, due to the risk assessment performed within the DA, to reduce the number of maintenance requests. RTA on the other hand always enters the at-risk state.

3.3.3.2 DSR2

DSR2 is concerned with the timeliness of messages and assumes that the timeliness of messages generated by DT1 will be checked when they arrive at the base node. Dropped packets are not a concern for this DSR, as dropped packets are checked by DSR1, however if a packet took longer to arrive than 1 minute then this DSR would be invalidated. For this reason a number of experiments were created which measure the maximum delay that can occur, in order to assess the timeliness of messages in the presence of the underlying radio stack, including the Routing layer and the MAC layer. One main issue with the transmission of packets is collisions, which causes backoff at the radio layer.

To assess the maximal delay a number of tests were conducted using NS-2, configured the same way as in the previous simulations however with all nodes being located in the same room. The number of nodes in the room was increased from 1-256, with all nodes attempting to send a packet to the base station simultaneously. This instance, when all nodes attempt to communicate at once, is referred to herein as the critical instant. Figure 3.13 shows the results of this experiment. From the graph it can be seen that the packet delay increases as the number of devices increases, with a maximal delay of 22 seconds occurring with 128 devices. However, as the number of nodes increases further, the packet delay is reduced, down to 13 seconds with 256 devices.

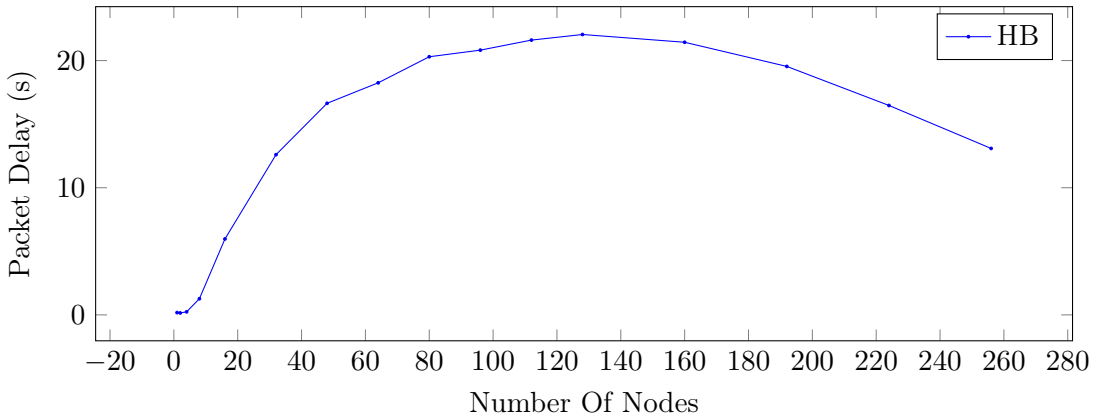

Figure 3.13: Number of Nodes vs Packet Delay

The reason behind this behaviour can be seen in Figure 3.14 which shows the number

of packets successfully received at the base station. As the packet delay is increasing some nodes are failing to transmit, resulting in packets being dropped, which in turn allows messages from other nodes to be received which reduces delay. From these results it can be seen that timeliness is not an issue, as the maximal delay was well below the accepted delay for the application, and where packets were dropped, they are caught by DSR1.

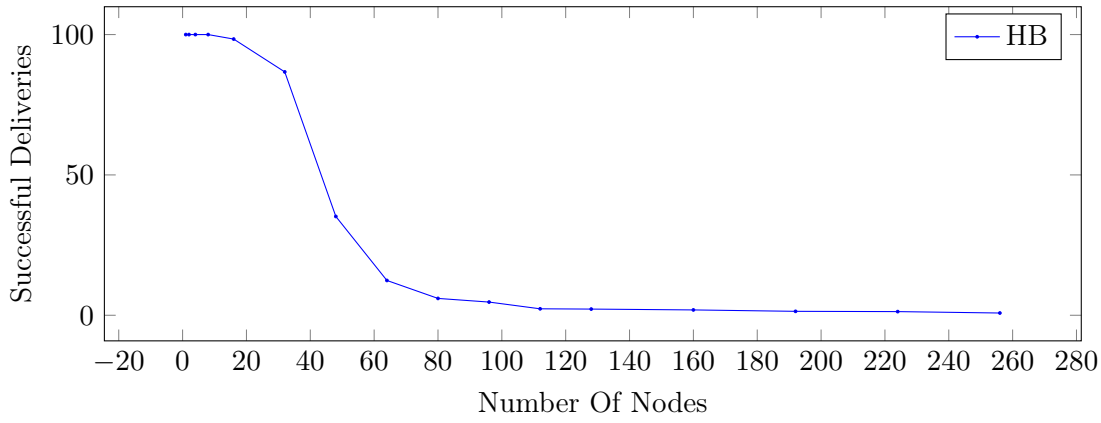

Figure 3.14: Number of Nodes Vs Successful Packet Delivery

In reality, nodes will not have perfectly synchronised times, resulting from either inaccuracies in the time synchronisation or clock drift within the hardware. To assess the impact of desynchronisation on the packet delay, both packet delay and dropped packet rates were recorded whilst the start times of nodes were chosen in a random uniform manner between 0 and $maxTime$, where $maxTime$ is increased to 2 seconds in 0.2s increments.

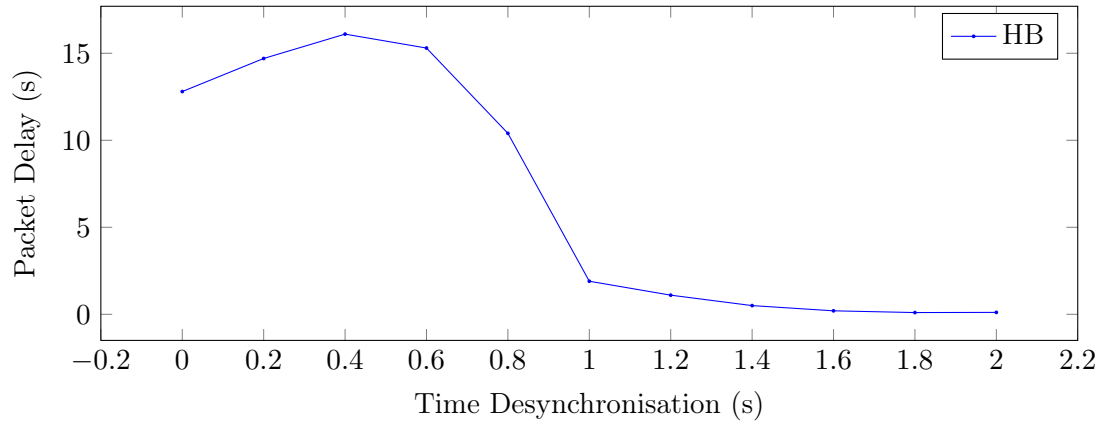

Figure 3.15: Time Desynchronisation Vs Packet Delay

Figure 3.15 shows that as time is desynchronised the time delay of packets gradually decreases. Figure 3.16 completes the picture showing that the number of dropped packets also reduces until 1.8s where all 256 packets are successfully delivered. From these results

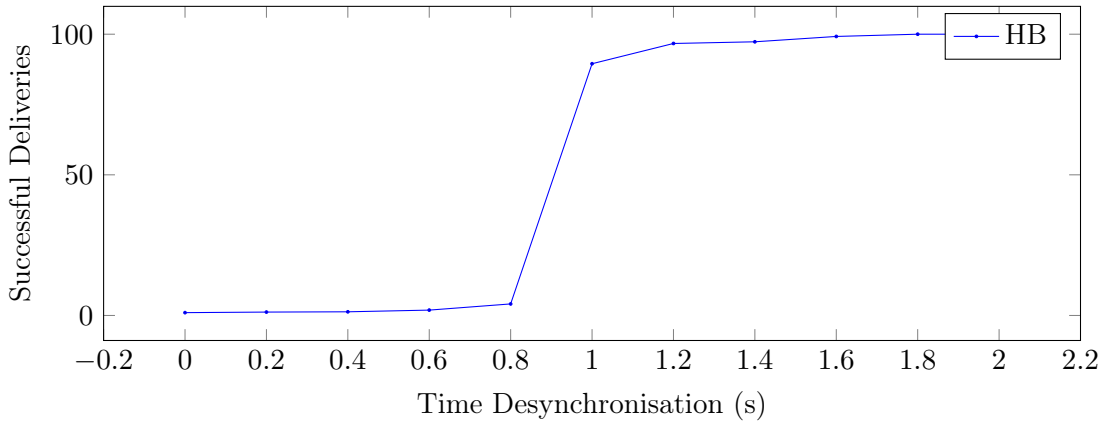

Figure 3.16: Time Desynchronisation vs Successful Packet Delivery

it can be seen that the timeliness of packets for our application does not need to be monitored, as the acceptable delay is substantially larger than the maximum that can occur. For this DSR the system can simply monitor for dropped packets, which is covered by DSR1, requiring no additional messages.

3.3.3.3 DSR3

Should sensors fail they may return implausible values. This could either be due to the sensors failing with full-scale deflection, or in the case of sensors returning random values, some would be erroneous. To ensure that these errors can be detected sensor readings are periodically intercepted when being returned from the sensors, and replaced with erroneous readings. These readings will then be reported as normal to the sink node, which in turn will detect the failure. As the sink node knows the DT schedule, it will know these failures are part of the test, and thus will flag DSR3 errors as detectable. As this requirement was systematically derived as part of the DSR process our approach is the only to specifically check for this failure, with the other health-monitoring approaches commonly raising a large number of false positives (a fire event when there is no fire).

To ascertain how the different approaches detect the failures and the extent to which any false positives are raised, experiments were performed in which the sensors on the devices fail every 24 hours, reporting random readings in the range 0-1000° C. This was chosen over full-scale deflection as it is the most relaxed assumption, as full scale deflection would be easier to detect, allowing DDC to perform better.

Table 3.10 shows the results of this experiment. It can be seen that both HB and RTA raise a large number of false positives as they fail to check the plausibility of the sensor

Node Type	Sensor Failures	False Positives	Error Rate
HB	105	105	100%
RTA	102	102	100%
DA	106	7	6.6%

Table 3.10: False positive rate of the three HM systems in the presence of sensor failures.

readings. Since DA is a systematic approach to deriving tests, it explicitly covers this case and correctly raises maintenance requests in response to detected failures. A small number of false positives can occur however where the erroneous readings are plausible for a fire, resulting in the device not being flagged as having failed.

If full scale deflection is used for the devices then the results are as seen in Table 3.11. In this mode a sensor reports its highest possible value when an error has occurred, which would be a plausible task for digital sensors. In these results it can be seen that the DA approach is even more effective than with random failures, as failures are detected rapidly and maintenance is scheduled immediately. The other two approaches raise the same number of failures as they cannot distinguish between failed devices or detected fires.

Node Type	Sensor Failures	False Positives	Error Rate
HB	104	104	100%
RTA	109	109	100%
DA	101	0	0%

Table 3.11: False positive rate of the three HM systems in the presence of full scale deflection sensor failures.

3.3.3.4 DSR4

This DSR covers the ability of the network to perform in the presence of network anomalies. No explicit tests are performed for this DSR as it was argued that network anomalies will be manifested as dropped packets at higher levels of the network stack. These dropped packets will then be covered by DSR1 as in Section 3.3.3.1.

Environmental noise will affect the network in one of two ways, either being detected by the mote by CSMA/CA, causing back-off to occur, or interfering with the packet in-transit. Interference en-route will either cause the reception of the packet to fail, or a modification of the packet the contents, which will also cause the radio layer to reject the packet as it would fail the CRC checks. Both of these events will cause the packet to be retransmitted, either due to the back-off, or the failure of an acknowledgement causing a

retransmission. In these circumstances the packet is becoming further delayed, however Section 3.3.3.2 has established that should there be delays due to packet collisions, this will manifest as dropped packets after the radio timeout, and is covered by DSR1.

3.3.3.5 DSR5

The final issue that was identified by our analysis was the complete failure of the monitoring system. This is a possible issue as should the health-monitoring system itself fail, without notifying the operators, then the DSRs will fail to be met. A failure of the DSRs could cause failed devices to go unnoticed, possibly allowing the system to silently miss fires. This DSR establishes that should the monitoring system itself fail, the operators will be notified. To establish that this is the case, another test is defined which pauses all of the DTs for a specific period of time, should any DTs be received in this time the appropriate device is flagged as failed. During this period, if all devices are operating as intended, no tests will be received (as they are explicitly paused), causing a error to be flagged on the operator device. This error, as it was deliberately raised, will be caught, ensuring that in practice a similar error would be detected.

Finally, as this test has the ability to suppress all network errors if it was implemented incorrectly, it is proposed that this component of the system is implemented using traditional safety-critical software development techniques. This should ensure that there is a basic level of service that can be guaranteed, which allows procedures to be put in place to evacuate the building or otherwise should all communications, and thus tests, be blocked.

3.4 Summary

In this chapter, it has been shown that by following the DA process: performing HAZOP, generating the DSRs, generating and reducing to a number of DTs, and then checking these at run-time, assurances can be made about the availability and reliability of the system. It has been discussed how the system will interact with the operators of the network and how the various failure modes of the WSN can be detected and reported in a timely manner. It has been shown how changes in the health-monitoring frequency and the failure rate of the devices affect the maintenance of the system, and ultimately the reliability. To assess the impact of these parameters DA has been compared against two alternative solutions, Run-time Assurance (RTA) and Heat Beat (HB). When considering

the detection of normal events (DSR1) it has been shown that when the failure mean is less than 8 months HB provides lower amounts of time at-risk, however with a much greater number of maintenance requests. When the failure mean is greater than 8 months the time at-risk is similar between HB and DA, with DA using less messages and thus lower power consumption. RTA on the other hand uses slightly less maintenance requests then DA, but with a much higher time at-risk over the full range of failure means and monitoring periods used within the evaluation. It has also been shown that false positives that HB and RTA experience can be reduced by using DA as it performs on-line tests to ensure it can successfully detect this type of failure. Finally it has been identified that should the assisted living application require low levels of maintenance (weekly replacements) and last for a considerable amount of time (greater than one year), the mean failure of the devices ideally needs to be over 10 months. These failures can be due to either hardware failure, which was not experienced in any of the trials in this chapter, or to no remaining power in the devices, which is to be addressed in the following Chapters.

Chapter 4

Dynamic Duty Control

In Chapter 3 event-driven applications were analysed for dependability, identifying that the lack of information in these scenarios requires a method of deriving a set of run-time tests to ensure that they are operating as intended. In contrast to event-driven applications there are also a large number of applications that are streaming-based, where large numbers of messages are being continuously sent to the sink. In these applications it is easier to deduce that the system is operating as intended, however with this increased amount of transmitted data, battery power becomes a major issue. As failure of the battery typically leads to failure of the network, battery failure can be identified as adversely affecting dependability. Energy harvesting approaches can be used to extend the battery power, however in the majority of cases this is not sufficient to permanently power the devices, and merely delays the failure of the system.

As identified in Chapter 3 the main contributing factor to the Reliability of WSNs, and thus Dependability, is the mean time between failure. It was shown that a mean time between failure of 10 months is ideally required for a dependable assisted living scenario. As identified in the literature review the major contributing factors are the processor and the radio. As the processor can already automatically put itself into power saving states when there is no pending processing work we instead focus on the power required when the radio is active.

Radio power has been the focus of a large amount of literature, with the main focus being on the lower MAC level protocols, typically aimed at providing a set level of service with the minimal amount of disruption. The majority of these approaches use periodic probing of the radio to achieve this goal. These approaches, as evidenced by ContikiMAC, do perform well when the application is relatively unknown, however by taking a similar

application-centric view, as in Chapter 3, power consumption can be reduced further.

DDC takes the identified application timing requirements and further constrains the radio allowing the radio to be turned off for longer. This is achieved by applying an additional radio duty-cycle above the routing and MAC levels, but one layer below the application layer. This allows current routing and MAC protocols to be used with no modification as shown in Figure 4.1, where the normal MAC radio cycling is restricted based upon the application. In this figure the X axis represents time, with R packets at the Routing layer being route requests, D being data packets, and A being acknowledgements. This application attempts to periodically send packets, this initiates a route request taking two packets, which in turn causes the probing MAC layer to transmit data. If DDC has information about the packet generation it can restrict the radio to just the required area as shown, removing excess radio probing. This figure is not to scale, as in reality if the application was of the order of 10s, with the MAC at ContikiOS' default 8Hz probing rate, there would be substantially more wasted radio time between transmissions. To ensure that the timing requirements are met without needing a large amount of state information, a feedback-based approach has been chosen. This approach measures the current timeliness and uses this information to inform the duty-cycle accordingly.

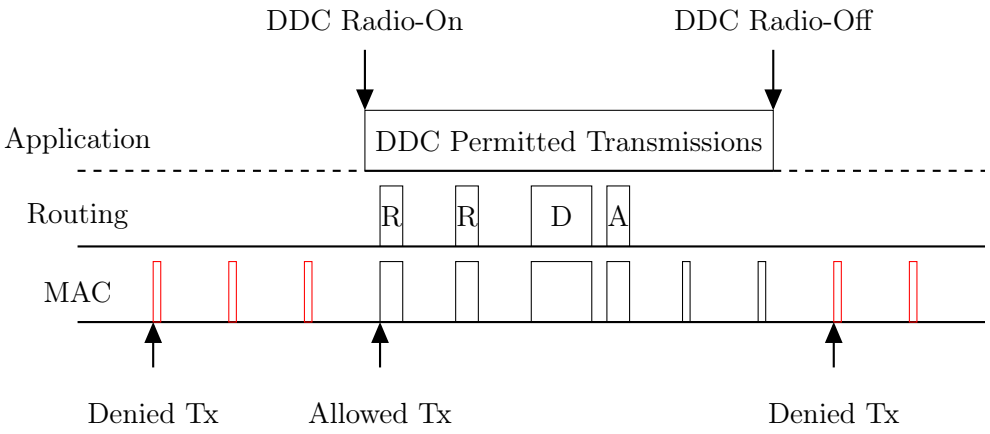

Figure 4.1: The effect of DDC upon network traffic

The objectives for this chapter are as follows:

DdcObj 1 - Given an application with a set deadline and data-generation rate, a protocol can be derived which uses this information to further restrict the radio-on time, saving power.

DdcObj 2 - In restricting the radio-on time the protocol must also ensure that under

reasonable network conditions, the deadlines of the application are always met.

DdcObj 3 - Should network conditions change, the protocol must adapt at a suitable speed. Large changes in conditions should have rapid changes in protocol response.

DdcObj 4 - Any such protocol must use values known, or easily derived, by the operator. Any parameters which need to be tuned to a specific deployment should be avoided.

Similarly, as in Chapter 3, it is important to emphasise that no guarantee can be made about factors such as deadlines or power consumption under all environmental conditions. We can only make guarantees when these are within a specified variance of the current conditions.

4.1 Overview

As identified within the literature review it is common for modern MAC protocols to periodically wake, sample the radio channel to detect if a packet is incoming, and then receive the packet or sleep accordingly. This probing, as in the case of ContikiMAC and LPL, is set to a specified frequency, which is dependent on the acceptable responsiveness in sending a message. Higher frequencies allow for lower latency transmissions at the expense of wasted power probing the radio channel.

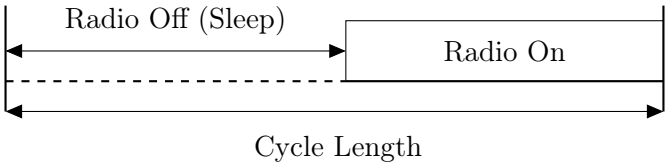

Figure 4.2: Relationship between the sleep duration and the cycle length.

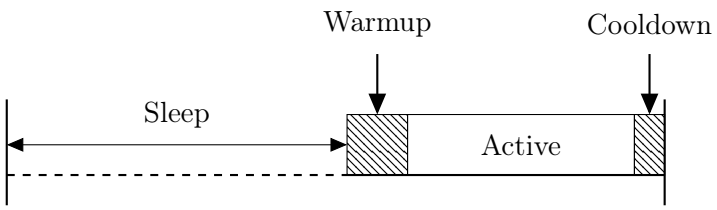

Figure 4.3: Overheads incurred when powering up the radio.

Upon initial observation it may be decided that if the data-generation rate, and thus the message sending rate is pre-determined, then the MAC layer could be set to the same frequency (f) to minimise wasted radio-on time. This approach however has a number of issues, primarily, that it assumes transmissions are always successful, which in the presence of interference, may not be true. Secondly, should the packet need to take a number of hops towards the sink, it may experience upto $hops * f$ delay, which may exceed any application deadline. Approaches such as DutyCon by Wang et al [159] attempt to solve this issue by taking the generation rate, the number of hops, and the current interference levels, to calculate an optimal MAC probing frequency. This however requires substantial knowledge about the network topology, traffic levels and the background interference, requiring further data collection and organisation.

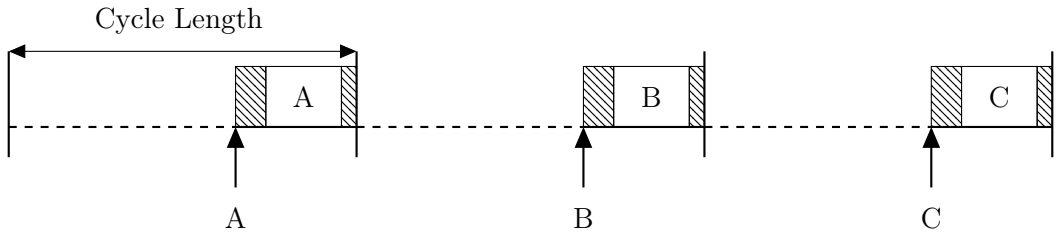

Figure 4.4: Repeated overheads when sending each message individually.

DDC measures the delay a packet experiences through the use of timestamps, and uses this information to wake the radio less frequently. Importantly, unlike Wang's solution, DDC does not assume a fixed MAC or Routing protocol and therefore DDC restricts the radio-on time at a higher level, in effect restricting the times in which the MAC and Routing layers can operate. Similar to Wang et al, DDC does however have the same assumptions about interference, that interference can be modelled as a reduction in the Packet Reception Ratio (PRR), due to the increase in required retransmissions, as also supported by Section 3.3.3.2.

Another difference between DDC and those used by alternative approaches is that DDC's duty-cycle, being placed above the MAC and Routing layers, allows DDC's awake period to cover full end-to-end communication, whilst relying on the original MAC protocol to efficiently use the radio during the awake period. This allows DDC to receive the efficiencies provided by the MAC layers when it wishes to transmit, such as those given by ContikiMAC and LPL, and save more power by disabling their use at other times. To ensure that DDC's awake window is of acceptable size for end-to-end transmissions, even in the presence of varying external interference, additional awake-time called slack

is introduced. Slack represents the main inefficiency within DDC, however it cannot be removed as it allows DDC to handle variations in the background interference, manifested as varying transmission durations due to the re-sending of messages as identified in Chapter 3. DDC still represents an improvement over traditional approaches as in the worst case, where DDC has an awake window the same length as the cycle length, it is effectively the same as the traditional approach. Should the awake window be smaller than this case then the radio-on time is further restricted, leading to more power savings.

DDC's only other requirement is the inclusion of timestamps on messages sent to the sink. Typically these are already included on sense-and-send type messages as they monitor the environment, however as their size is relatively small they add minimal overhead if they are not already included. Timestamps can be set to either relative or absolute time, as long as time synchronisation has been performed. This allows either the relative offset for the specific node to be obtained, or for the specific node to be set to the current global time. This timestamp is included with all data sent to the sink, where if necessary, it is translated to the current global time before being used to inform the DDC algorithm.

Until this point only message-generation rates have been discussed, however as noted in Chapter 3 there are many types of application where there are additional deadlines that messages are required to meet. In the fire detection example analysed in Section 3.3.2, it has been demonstrated how DDC works when messages should be sent as soon as they are generated, however by delaying when DDC activates, the radio packets can be implicitly queued until activation time. For this concept, the notion of relative and absolute deadlines is used. Relative deadlines are the maximum allowed time from the generation of a message until the message is received at the sink (i.e. temperature readings must be reported within 60 seconds). Absolute deadlines are the actual time at which the messages must arrive and are simply the synchronised global time plus the relative deadline. By delaying the release of packets we can effectively form a packet queue, with packets arriving at the sink node closer to their absolute deadline as shown in Figure 4.5. In this figure the incoming arrows are the points at which the device reads its sensors and attempts to send a packet, with the labelled rectangles being the points where the associated packets are transmitted. The hashed sections of the diagram are the time it takes from starting to turn the radio on or off, to the radio being ready to send packets or being in its fully powered down mode.

It can be seen from this diagram that this can also provide further energy savings, as the overhead of starting and stopping the radio is only incurred once for sending all

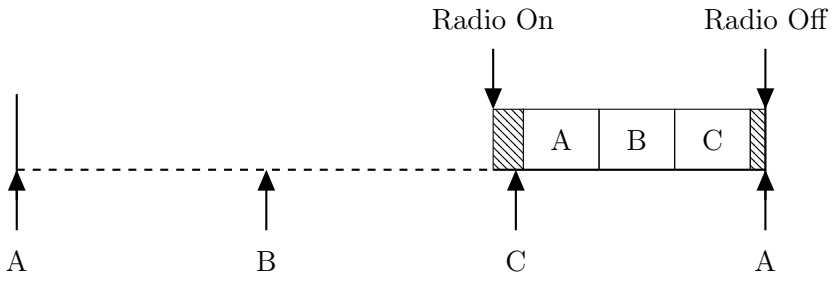

Figure 4.5: Restriction of radio-on times forcing packet queueing.

three messages, whereas otherwise the overheads would have occurred four times as shown in Figure 4.4. Therefore, when the deadline is greater than the generation rate packets are implicitly batched up, saving power. It is important to make the distinction between purely batching transmissions, as in DDC, and batching data into the same packet, or data compression. Whilst DDC clearly provides an opportunity to batch and compress data, compression is outside the scope of this work, instead this work focuses on the savings obtained from purely transmitting a number of packets in rapid succession.

There are three main categories of overhead that may be incurred when transmitting a packet, physical layer delay, MAC layer delay, and routing layer delay. The physical layer delay is caused by the time it takes to turn on the radio circuit on the device, for the radio to become stable and ready to accept packets for transmittal. The MAC layer adds a variable amount of delay depending on the MAC protocol used. Should MAC protocols such as nullMAC be used then any delays experienced due to CSMA/CA will be experienced by all packets regardless of batching. However should MAC protocols such as LPL or ContikiMAC be used, where the radio is cycled in a periodic manner, with selective send and receive times, these delays may be incurred when attempting to send a packet. These MAC protocols have a number of additions to help alleviate this issue, with the main addition being the ability to transmit successive packets immediately, whilst the sender and the receiver are synchronised and communicating. DDC clearly benefits from the addition of this feature. Finally the routing layer can also add overheads, commonly in the form of creating and maintaining routes. These overheads are especially true in adaptive routing protocols such as AODV, where routes are formed on-demand, with timeouts specified to indicate when a route should be discarded. In this case it is clear that DDC, by transmitting multiple packets in succession can use the same route, incurring only one set-up cost, whilst also having possibly more chance of success due to the small amount of time between the route being established and the packets being sent.

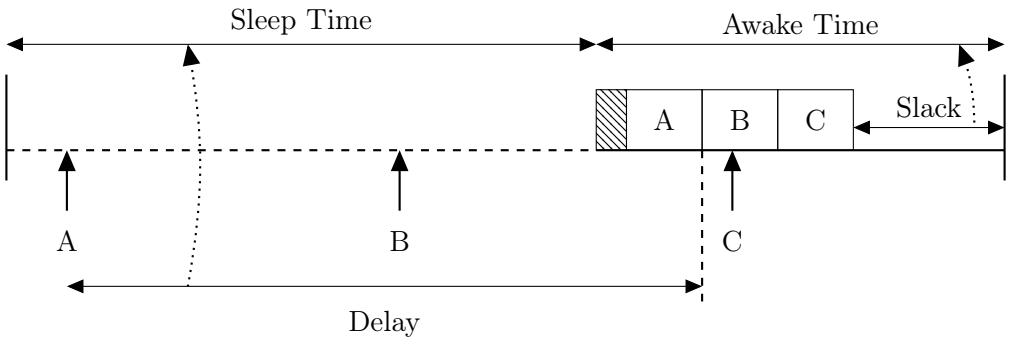

Figure 4.6: Flow of control between Slack, Awake, Delay and Sleep values.

As the need has been established for controlling the awake period in both frequency and duration, two controlling variables are introduced, Sleep Time and Awake Time, as shown in Figure 4.6. The frequency of transmissions is controlled by the Sleep Time, which in combination with the Awake time defines the number of messages that have been delayed, and thus the resulting batch size (the number of packets sent in the transmission window). The combination of the Awake and Sleep times is referred to as the Cycle Length. This can be used to increase the batch size by increasing the cycle length, or reducing the batch size by reducing the cycle length. As overheads are reduced as the batch size increases, it is clear that some limiting factor is required to restrict the size of the data batches. In DDC this is $Max(delay)$. The second variable, the Awake Time, must be large enough to ensure that all pending messages are sent in this transmission window, as required to meet the deadlines.

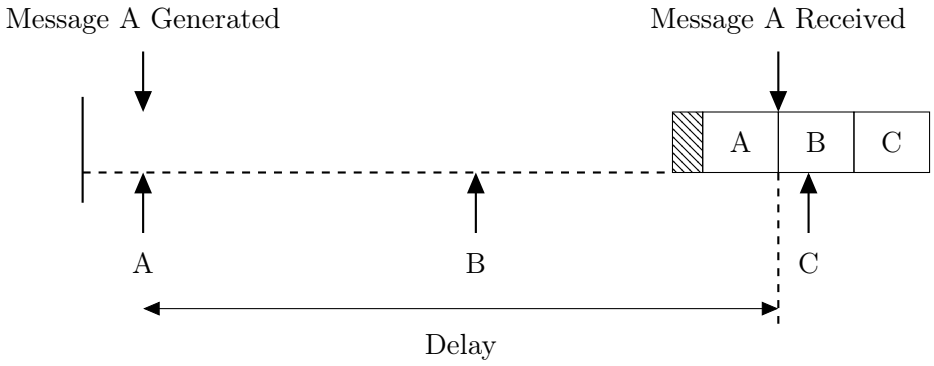

Figure 4.7: Calculation of the delay for a given transmission window.

The delay of an individual packet is defined to be the amount of time from a packet being generated, to the time of it being received at the base station as shown in Figure 4.7. This delay is a product of a variety of factors from packet orderings, routing layer decisions, MAC layer protocols and environmental interference, all of which are abstracted

away into a single delay factor, meeting objective 4. The programmer must specify the maximum acceptable deadline for generated messages, which is a relatively easy number for the operator to provide, which in turn will be used as the delay limit. In the previous fire detection system the deadline would be 5 minutes, this being the maximal acceptable time between a fire being detected and it being reported to the operators (note the delay is measured from the event being detected, not the event occurring, and thus may need to take into account the detection time).

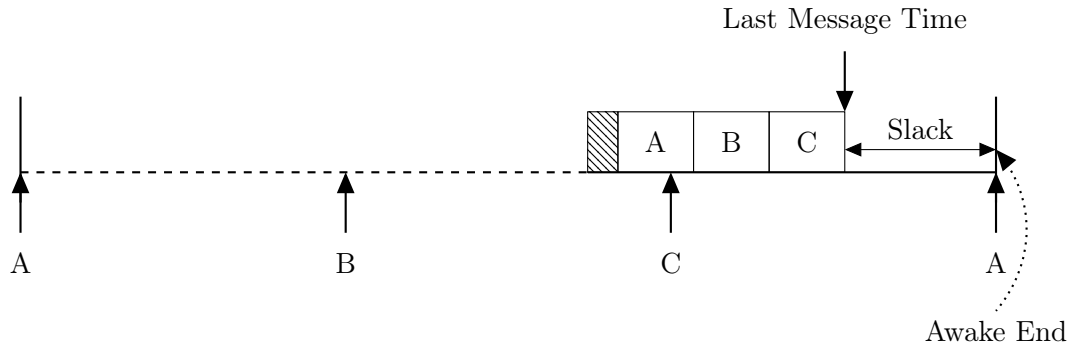

Figure 4.8: Calculation of the slack time for a given transmission window.

The slack factor is the amount of time left at the end of a transmission window after all messages have been sent. This value can be calculated by subtracting the end of the awake window from the time at which the last message was received, as shown in Figure 4.8. The least power-costly value for this factor is 0, indicating no excess time, however this means that if the transmission times of any packets were to be delayed or extended then the transmission window would be missed, forcing these messages to wait for the next window. As delays can be incurred due to external noise or collisions, which are relatively common, some packets will fail to be transmitted. As objective 2 states that deadlines must always be met under reasonable conditions, this slack needs to be increased to accommodate any delay that may occur under these conditions. This increased slack is referred to as the targetSlack.

Variable	Increase	Decrease
Cycle Length	More packets per batch	Less packets
Awake Length	More slack per batch	Less slack

Table 4.1: Effect of changes on cycle and awake length on packet count and slack

There are a number of approaches that could be used to select the appropriate cycle length and awake length for a set target frequency and duration, but to minimise the ex-

Listing 4.1: Conditional Controller

```
1 void updateValue(pid* in_var,int error) {
2     if (error > 0) {
3         in_var -= dV;
4     } else if (error < 0) {
5         in_var += dV;
6     }
7 }
```

ecution time and the power consumption required by the processor, the solution must be computationally cheap to run on the motes and therefore can't be overly complex. This rules out any simulation-based approaches, such as search based solutions, and instead creates a requirement for an on-line approach. Initially, for this analysis, a simple conditional approach was used, as shown in Listing 4.1, with the error being the Target Slack - Current Slack. Whilst this approach works to a degree, one important issue is choosing dV, the amount by which the variables are modified. If this value is set too large, the algorithm repeatedly overshoots, oscillating by a large amount around the desired value. If the value is too small, the algorithm is too unresponsive to changes, taking a long time to approach the target value. To resolve this issue, PID loops were chosen as they make smaller corrections to the necessary variables as the desired setpoint is approached, and larger corrections when the target is far away.

Figure 4.9 shows the final set-up for the DDC approach. Two PID loops are used simultaneously to control both the number of packets in each batch and the amount of slack that is present. This allows the system to adapt slowly with small changes in the environment (such as weather), but more rapidly when the environment suddenly changes (if a large source of interference such as a microwave were to occur), thereby meeting objective 3 for DDC.

PID loops consist of three main components, Proportional, Integral, and Derivative components. When working with continuous signals the Integral and Derivative components would be expensive to compute on mote-class hardware, however as the rate of PID loop updates is relatively infrequent, on the order of minutes, the discrete version of the algorithm can be used as shown in Listing 4.2. To further reduce the computational complexity of the algorithm, dt is set to 1, removing the need to divide for the Derivative component. This change allows the PID algorithm to be implemented without any internal divisions, which are expensive, and removing the need for floating point operations,

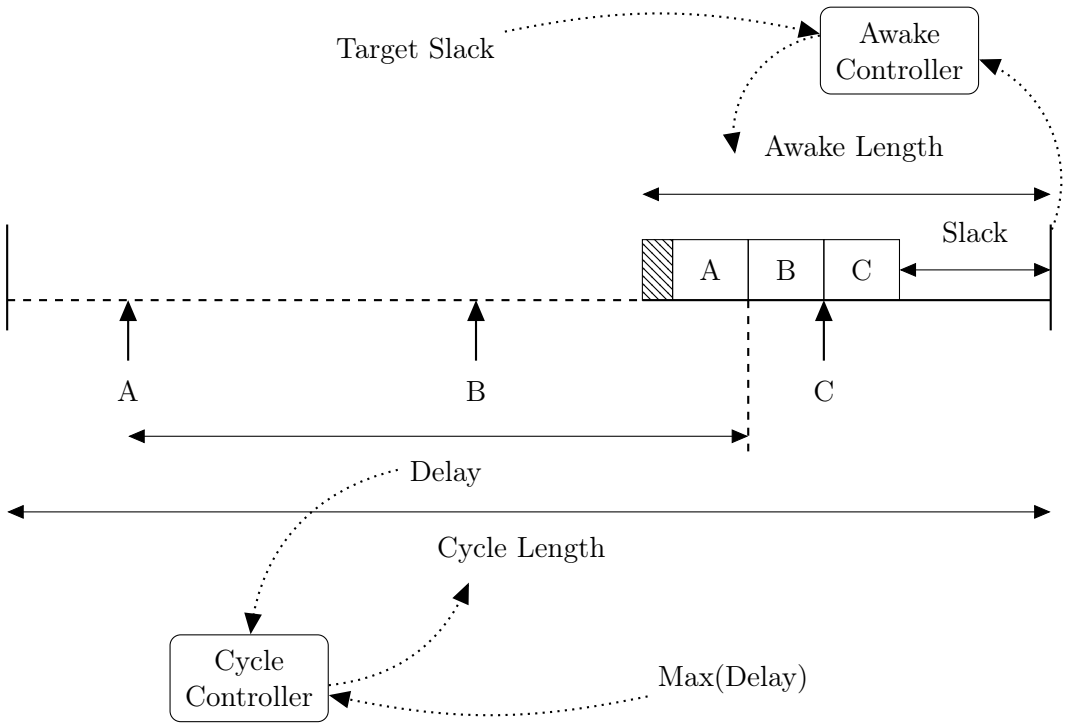

Figure 4.9: DDC PID controllers with input, setpoint and output variables.

which are prohibitively expensive. This allows multiple updates to be performed with low overheads, however the return value is especially large, requiring a final reduction step when reading back the PID values. Even when ignoring the removal of floating point operations, this arrangement is in the worst case equal to the original algorithm, and when considering that more updates may occur than reads, resulting in savings. The resultant *updatePid* function is given in Listing 4.3.

As these PID loops are based on the network's performance when gathering data at the sink from all source devices, the PID loops themselves will be located only on the sink device. The sink will update the PID loops at the end of each transmission window, when the delay and slack values are available, with the resulting awake and sleep sizes being broadcast across the network if they deviate by a set value from the previously transmitted value. This centralised approach is necessary as all devices must be online at the same time instant to ensure messages can traverse the span of the network, and as such all the periods and durations must be consistent. Secondly as the sink device is the destination for all the data packets, any attempt to distribute the PID loops to obtain redundancy is fruitless, as with a failed sink the application cannot function.

Listing 4.2: PID Loop

```
typedef pid {
    int p = 0, i = 0, d = 0, dt = 10;
    int pVal = 0, iVal = 0, dVal = 0;
    int last_val = 0;
};
void updatePid(pid* in_pid, int in_val) {
    in_pid.pVal = in_val * in_pid.p;
    in_pid.iVal += in_val * in_pid.I * dT;
    in_pid.dVal = (in_val - last_val) / dT;
    in_pid.last_val = in_val;
}
int readVal(pid* in_pid, int in_val) {
    return in_pid.pVal + in_pid.iVal + in_pid.dVal;
}
```

Listing 4.3: PID Optimised C

```
void updatePid(pid* in_pid, int in_val) {
    in_pid.pVal = in_val * in_pid.p;
    in_pid.iVal += in_val * in_pid.I;
    in_pid.dVal = in_val - last_val;
    in_pid.last_val = in_val;
}
```

4.2 Method

As the DDC algorithm has been described it must be evaluated to ensure that it meets all of the objectives for this chapter. For this reason a number of experiments will be performed which show the following:

1. Feedback Stability - As the two feedback loops are not completely isolated, due to them operating on variables which affect each other, it needs to be shown that the solution is stable and free from side-effects caused by the cooperating feedback loops.

2. Handles Network Variance - Feedback loops are used due to their adaptive nature. It needs to be shown that this feedback-based approach provides resilience to variance in both the reliability of network transmissions and also in the presence of node failures.

3. Reduces Power - The main purpose of DDC is to use the application parameters to realise additional power savings over alternative approaches. It needs to be shown that when compared to other feedback approaches DDC performs equally or better than the alternatives.

4. MAC and Routing Agnosticism - One of the claims of DDC is that its design makes it MAC and Routing agnostic. DDC must be shown that with real MAC protocols it maintains the same level of performance and also still saves power, and also within Multi-Hop routing scenarios.

To assess the DDC approach the application used needs to be changed as the previous event-driven fire detection generates very few packets. For this reason a climate monitoring application has been chosen, which in a similar manner to the previous application repeatedly monitors the temperature. In contrast to the event-driven application the raw readings are sent directly to the sink, instead of assessing the readings on-line. This allows for a different class of applications, such as ensuring rooms are sufficiently warm in an assisted living facility, detecting when windows may possibly have been left open. The classification of the data is outside the scope of this work, instead it is focused on delivering all the data in a timely manner to the sink, whilst also attempting to keep the energy consumption of the network as low as possible.

DDC will be evaluated in a number of ways so as to meet the objectives stated above. Preliminary results are gathered through numerical simulation to assess the effect of different P, I and D parameters on the responsiveness of the application without the Routing or MAC layers affecting the results and to compare against alternative approaches. Simulations using real WSN operating systems are undertaken to assess the impact of the MAC and Routing layers on the approach. And final results from real devices are gathered for verification of the simulations.

Simulations were undertaken using the Cooja simulator. This was chosen over NS-2 as it allows the real binaries to be run on the simulator using emulated hardware. This allows for greater accuracy of the simulation, as the algorithms under the simulator and the real hardware are the same implementation and all OS overheads are also included. However, the Cooja simulator is substantially slower than NS-2, with simulations of 10 devices running in almost real-time, as opposed to NS-2 which can run years of simulation time in a few hours. A slow simulation speed is acceptable for this application where events are happening on the scale of minutes, and the overall outcome is avaliable in no greater than a few hours, unlike failures which occur over months. More detail about the environments are given in the following sections.

4.3 Numerical Simulation

Initial evaluation is performed using numerical simulation, with this form of simulation being specifically chosen over other simulators such as TOSSIM or Cooja for a number of reasons. The main reason is to provide a fair platform to compare DDC to alternative approaches, such as that proposed by Wang et al. This is due to Wang's approach requiring global information about the routing paths and interference experienced at every node in the network, which a custom simulation easily provides. Another advantage of a custom simulator is the ability to introduce interference, in the form of changes in Packet Reception Rate, which allows the reactivity of the algorithms to changes in the environment to be measured. Within the numerical simulator, each node is modelled individually, but the communications are abstracted to simple message queues and formulae which are used for testing successful delivery. This is done so that the following three main objectives could be achieved.

1. **Evaluation of P, I, and D settings** It is important to obtain insight into how the different input parameters affect the performance of DDC, and so evaluation of different P, I, and D settings is required. The custom numerical simulator allows large-scale testing of different P, I, and D settings to be performed. Ideally this evaluation should show that our approach, whilst obtaining better performance with optimal settings, does not require specific values for good performance, simplifying the process for application developers.

2. **Testing the reactivity of the approach** Any solution should ensure that the application requirements are met even when changes in the environment occur, especially for dynamic approaches. The custom numerical simulator allows the Packet Reception Rate (PRR) to be varied at run-time, allowing for the speed and accuracy of the feedback in response to changes in the environment, to be tested. The population count of the network can also be varied, with nodes being removed or added to the network throughout the experiment to assess the response of the algorithms.

3. **Power consumption evaluation against other approaches** To ensure that the network lifetime objectives as outlined in the conclusion of Chapter 3 are met, an estimation as to the lifetime of the approaches is required. This requires a transformation from the target-independent duty-cycle values from the previous objective, into target-specific power consumptions using values obtained from within the literature.

These power consumptions can then be combined with typical battery capacities to obtain a network lifetime, to evaluate against objective 1.

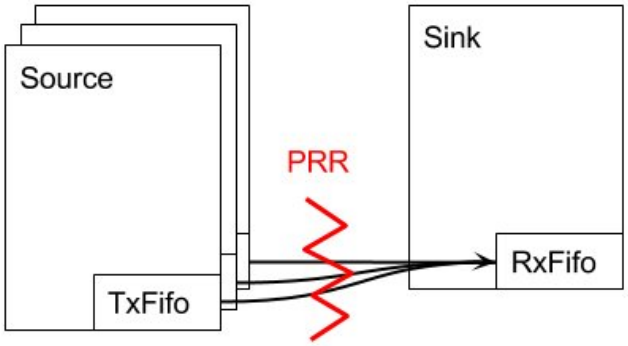

Figure 4.10: Numerical Simulator with simple communications

The implementation of the custom simulation can be summarised by Figure 4.10, with individual motes realised as instances of the mote class, and communications modelled by TX and RX FIFOs in each class. All motes share the same duty-cycle, with all devices attempting to send once the radio is activated. When a pending transmission is completed, the packet is moved from the TX queue to the corresponding RX, once two conditions have been met. Firstly, the transmission duration must have elapsed, this indicates that the radio was on long enough to transmit the appropriate packet. Secondly, once the transmission duration has elapsed, the packet is probabilistically moved from the TX to RX queue, based on the PRR. Should the radio be deactivated whilst there is a pending transmission, the transmission attempt is aborted and the packet remains at the head of the TX queue for the next transmission attempt. The transmission times in the simulator are obtained from measured transmission times for real devices within the literature.

For all of the following experiments, the same device layout is used, with 4 source devices and a single sink device. Each of the source devices are directly connected to the sink, using single-hop communication. Single-hop is used to avoid choosing a particular routing protocol for multi-hop communications, thereby focusing the analysis more directly on the timeliness of packets being received at the sink in the presence of radio contention.

4.3.1 PID Tuning Theory

This section will be split into two parts. The initial section focuses on showing how DDC, given no prior knowledge, converges on an optimal solution. This section will also

study how changes to the P, I, and D values, in conjunction with the setpoint, affect the convergence rate of DDC. The second section will study how in practice, with basic application knowledge, the PID loops can be primed in order to perform better in the initial instance. The second section will also search for optimal PID values, and show how these differ in performance to naive values.

Initial testing was undertaken with a singular PID loop with a setpoint of 1000. At each update round the return value from the PID loop is divided by 10 and then fed into the PID loop to calculate the next step. This division modification of the output values is to ensure similarity with the DDC PID loops, where the output values are correlated, but are not on the same scale as the input values. This is especially the case when considering the slack in the transmission window, where the slack is much smaller than the overall transmission window size specified by the PID loop. The implementation of the PID loop also follows that described earlier, avoiding divisions within the main controller and thus treating all values as integers. For this reason any input values to the PID controller are truncated to avoid floating point arithmetic, keeping the implementation efficient for WSN class devices. Finally all internal state of the PID loops are naively set to 0, where later in Section 4.3.2 it is shown how a basic estimate can be used to provide better initialisation values.

From initial experimentation a number of observations can be made, as shown in Figure 4.11, which correlates with the literature from Section 2.6.1. In this figure only a subset of all analysed results is shown, with each line showing a separate PID run, with different P, I and D values as identified in the key. The value is the output value from the PID loop, with the input value being this number divided by 10.

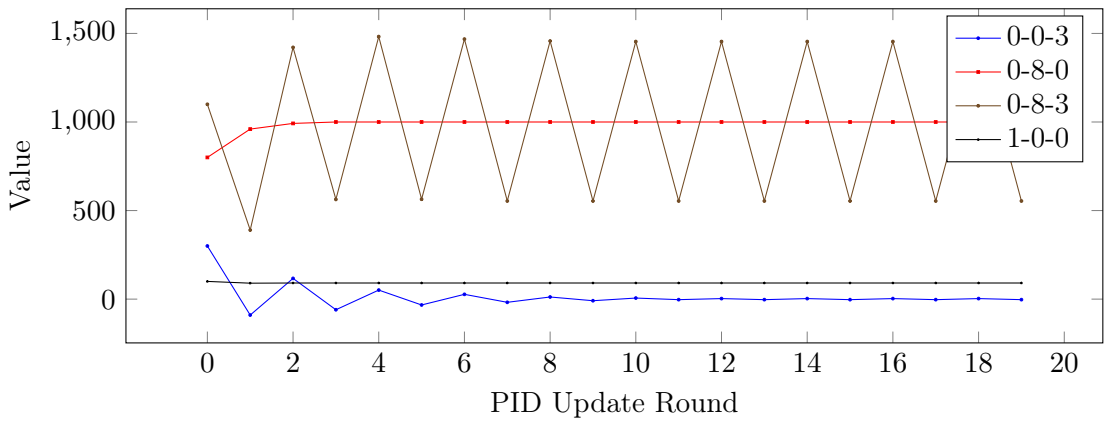

Figure 4.11: Four different effects of PID setting on results

The first observation is that any set of parameters which do not include an integral

component fail to reach the setpoint of 1000. This is due to the static nature of the P and D values, requiring that they be set perfectly correctly to map from input values to correct output values (a P of 10 in this case would counteract the division by 10, but in practice we do not know this correlation directly). Secondly it is identified that D values greater than the P values cause instability in the output, as the derivative factor, which aims to dampen the proportional factor, over-compensates causing oscillations. Finally we can see the ideal behaviour of our PID loop in settings 0-8-0, where the progress towards the setpoint is consistent and relatively rapid, however without the P or the I parameters the response time is slower than otherwise possible.

4.3.2 PID Tuning Experiments & Results

The initial objective is to assess how different P, I and D values affect the convergence of the PID controller towards the setpoint. Where two controllers are used, this is especially important as this solution has two controlled parameters that are correlated, possibly leading to competition between the two controllers. As identified in Section 2.6.1 of the literature review there are a number of methods that can be employed in order to tune the PID controller, however traditional approaches do not consider simulation. When simulations can be conducted search based methods can be used. As simulation allows for running multiple experiments simultaneously an exhaustive search over a range of possible P, I, and D values is performed, with the results being automatically pruned to leave a selection of candidates for manual review. Pruning is simple to implement as the two erroneous cases identified in Section 4.3.1 can be easily identified: constant oscillation, easily detectable by counting the magnitude of overshoots as they occur; or flat-lining, where the output variables become fixed at a value away from the set-point. The resulting candidates are then automatically graphed for easy assessment of convergence speed and stability. An additional advantage to performing such a search is that the effect from varying from the optimal solution can be assessed and conclusions drawn about the importance of obtaining precise P, I, and D values.

To more accurately represent the WSN scenario the setpoint for the delay is 60 seconds, with the setpoint for the slack being 200ms (57 packets). These values have been chosen to be representative of an environmental monitoring application, with 4 devices sampling readings every second, with readings being sent every minute. The slack has been chosen to allow for 24% more packets to be re-transmitted. It is important to note that this does

not represent a minimum PRR of 76%, as this slack represents the deviation in PRR over a single transmission window, and thus represents a reduction in PRR of 24% from 100% in one minute, for example. The experiments were also modified so that any runs taking more than 1000 PID update cycles are terminated, as this is commonly caused by poor PID settings, causing the output to always be 0, (PID of 0,0,0), and thus repeatedly updating the loop with meaningless values. Simulations are run for 1,000,000ms (16.7 mins), which represents 17 PID update cycles at the optimal PID settings.

Experiments are conducted with P, I and D values in the range of 0 - 20, in increments of 4. To assess the PID loops it is necessary to define when the PID loop will be classified as stable. This is due to the fact that the PID loops rarely settle on a singular value, normally fluctuating slightly around the setpoint. For this reason delay has been defined as stable when it is within 300ms of the 60 second deadline, with the slack being stable when it is within 50ms of the 200ms target. Using this term of stability the convergence time for each PID run is easily calculated by recording the time at which the controller was last unstable. The number of times that the setpoint has been exceeded is also recorded, with the value needing to exceed the stable region before it is recorded. Finally the sum of the absolute error between the setpoint and the current PID value is recorded, indicating how rapidly the current solution converged on the correct value. To find the absolute error for a pair of PID loops the magnitude of the errors must be similar, otherwise the result will be biased towards one loop, the delay in this example. The bias applied is based on the difference between the setpoints, with the slack error being multiplied by 300 to be the same magnitude as the delay.

Table 4.2 shows the top 10 ranking results from the exhaustive search, rated by sum errors.

From these experiments it can be seen that the optimal P, I and D values for each of the loops are 0-12-0 for the awake and 8-4-0 for the sleep. It can also be seen from these results that the correct operation of the system does not depend heavily on these values, as local deviations also perform well, just with a slower convergence speed. This indicates that there may be a recommended set of good values which could be used in the default case, with more tuning of the parameters should more performance be required.

Investigating more of the poorer solutions shows why some of these values are not obvious, as Figures 4.12 and 4.13 show competition between the slack and the delay PID loops. In this case the initial start-up causes severe competition between the loops, and so

Results			Awake Length			Sleep Period		
Total Error	**Settle Time**	**Overshoots**	**P**	**I**	**D**	**P**	**I**	**D**
369802	195428	4	0	12	0	8	4	0
376665	194840	4	4	12	0	4	4	4
381488	194552	4	0	8	0	8	4	0
381968	195191	4	0	12	0	4	4	0
389958	199715	4	0	16	0	8	4	0
391807	198271	4	0	16	0	4	4	4
392669	194261	4	4	12	0	8	4	0
394578	195164	4	4	8	0	8	4	0
395876	198538	4	0	12	4	8	4	0
396092	196534	4	0	12	0	0	4	4

Table 4.2: Top 10 ranking results from the exhaustive search

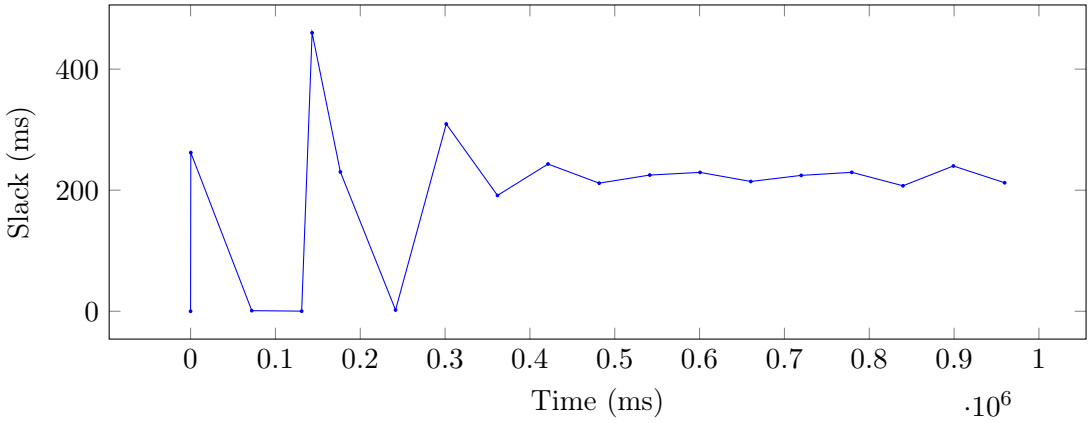

Figure 4.12: Competing slack over time with poor PID loops

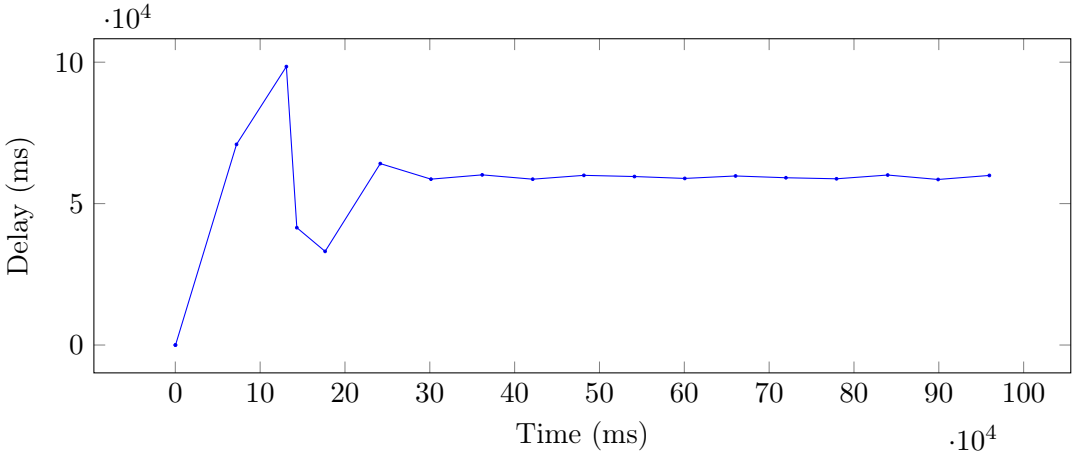

Figure 4.13: Competing delay over time with poor PID loops

in this case the delay, being orders of magnitude larger than the slack, causes many more packets to be generated than the slack can handle. This reflects the information discovered within the literature review in Section 2.6.1, stating that the initial start-up of the PID may

be erratic due to the large differences between the current error and the standard errors expected once the PID is running, causing an issue known as windup. Some of the same solutions identified in the literature review such as changing the PID values throughout the experiment could be used. The case presented sofar is the extreme case where no preliminary information is provided to the PID loops, however by providing an estimate of appropriate awake and sleep lengths it is possible to achieve much more optimal start-up performance. For the following tests the initial values for the Sleep and Awake periods are deduced from the packet generation rate and packet deadline. The Awake length and Sleep period are set to the values in Equations 4.1 and 4.2 accordingly. These initial values are not intended to be accurate, merely a value that should be easily estimated by the network operators. Unlike traditional approaches, DDC is an adaptive scheme which should not rely on well-informed initial values.

$$AwakeLength = \frac{deadline}{generationRate} \times txTime \tag{4.1}$$

$$SleepPeriod = deadline - Awake \tag{4.2}$$

Using these initial values it can be seen in Figures 4.14 and 4.15 how the convergence speed is greatly reduced from the original 3 PID rounds to 1 PID round. Another side-effect of providing the initial values is that the sum total error is no-longer dominated by the need to get to the setpoint as fast as possible, and instead some of the focus is transferred to ensuring stable operation later in the experiment. This is due to small oscillations causing a noticeable error, where a non-oscillating solution may take longer to converge, but it will incur less error during the stable phase.

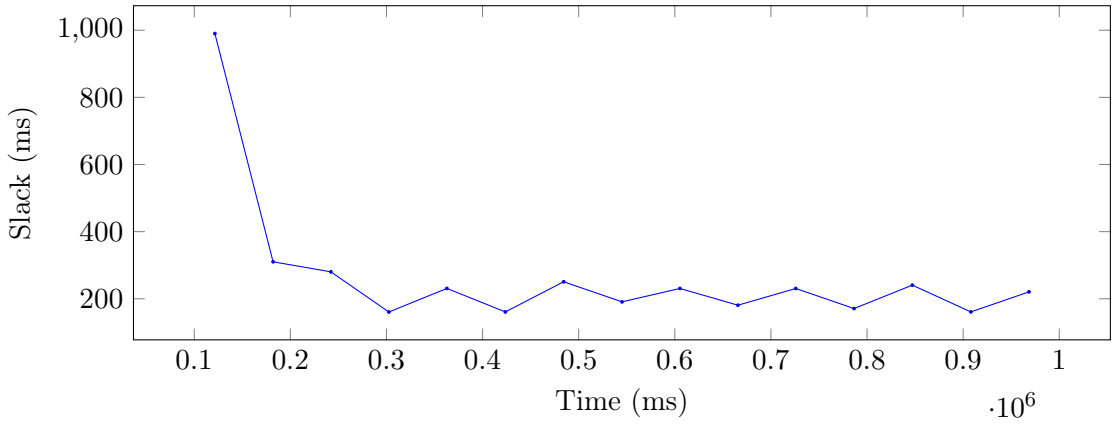

Figure 4.14: Slack over time with primed PID loops

As shown in Figure 4.15 the PID loops oscillate around the specific set-point, in this

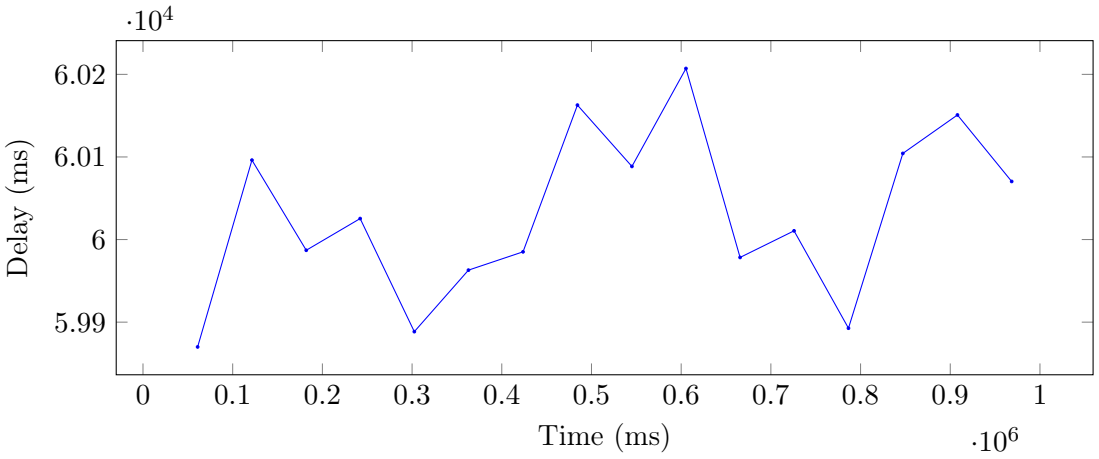

Figure 4.15: Delay over time with primed PID loops

case the 60s deadline. To ensure that this deadline is not commonly exceeded the variance in the delay will be measured. This variance is defined as the maximum absolute difference between the delay measurement and its associated set-point. The preceding 10 PID rounds will be measured, as it has been shown that oscillations occur on a scale smaller than this value, and the resultant variance subtracted from the set-point. This has the effect that the maximum oscillation may exceed the deadline, but in the majority of cases this will not occur.

4.3.3 Reactivity Theory

Tests need to be undertaken to assess the performance of DDCs reaction to changes in the environmental noise and the population count. To ascertain if the performance of DDC is acceptable it is also compared to two other approaches, DutyCon by Wang et al, and X-MAC. DutyCon was chosen as it represents one of the few reactive schemes that adapts to environmental changes, network traffic, and population changes, with X-MAC being chosen as a representative of traditional static schemes. In order to provide a fair comparison this section will use a version of DDC with default un-tuned parameters. This lack of tuning should only affect the time for DDC to adjust to new parameters, however the initial reaction should be the same regardless of the parameters (any deadline misses would happen regardless of the speed of the reaction, the only effect is how fast DDC will return to an acceptable level of performance). Deviations from the defaults could cause instability depending on the individual P, I, and D values, however this is outside of the scope of this section and is covered in Section 4.3.1.

To fairly evaluate the performance of the three approaches the same two factors will

be measured as an assessment of their reactivity, the delay and the duty-cycle. The delay is measured as it is the main application requirement that the approaches must meet, as should the delay exceed the deadline specified by the operator by a large amount then the system is operating outside of acceptable operating limits, effectively being a failure of the application. Duty cycle is the secondary factor as it directly correlates to the power consumption of the network, with the minimisation of this factor being requirement 1 as identified at the end of Chapter 3.4. The ideal solution is the one which manages to minimise the duty-cycle whilst meeting the application deadline.

4.3.4 Reactivity Experiments & Results - Packet Reception Rate

The first experiments assess the reactivity of the three approaches in relation to changes in the interference, from background interference, to sudden strong interference. To test changes in the environmental noise PRR is used, as it is the same approach that is used in DutyCon, and as such represents a fair comparison between the algorithms. The rate of change in interference is modelled as rate of change in the PRR ($\Delta PRR/\Delta T$), with sudden strong background interference being a rapid change in the PRR. For these experiments the rate of change of PRR is the main variable under scrutiny, with the reactions of the three approaches being monitored as the delay and the duty-cycle. This section will begin with a comparison of the three approaches followed by detailed analysis of the DDC approach.

Tests begin with a $\Delta PRR/\Delta T$ of 5%, incrementing by 5% every day to the maximum level of 80%. In each test the PRR begins at 100%, decreasing down by $\Delta PRR/\Delta T$ and then increasing back to 100% before the next $\Delta PRR/\Delta T$ change. The pseudocode for this test is given in algorithm 1.

Algorithm 1 Interference Testing

```
PRR = 100
ΔPRR = 5
while ΔPRR < 100 do
    while PRR > ΔPRR do
        PRR = PRR − ΔPRR;
        wait();
    end while
    while PRR ≤ 100 do
        PRR = PRR + ΔPRR;
        wait();
    end while
    ΔPRR+ = 5
end while
```

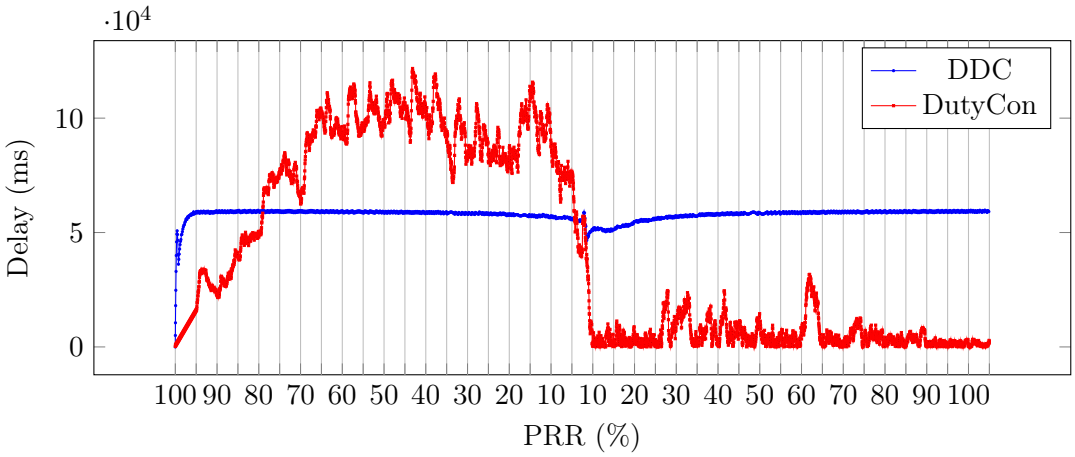

Figure 4.16: Delay over PRR for the two adaptive approaches

As the graphs for each step of $\Delta PRR/\Delta T$ are similar, Figure 4.16 only shows the results for a $\Delta PRR/\Delta T$ of 5%. It can be seen that DDC performs well throughout the experiment, due to the slack never reaching 0. It can however be seen that DutyCon misses deadlines each time the PRR is decreased, even at high values of PRR. This is due to DutyCon requiring time to obtain information from the network about all PRR links and traffic flows before a new period can be decided, in which time messages are starting to be delayed. DutyCon's main setback is the time which it takes to acquire the global state, with faster acquisition allowing for faster response to network changes. X-MAC has been omitted from this graph as it constantly has delays upto a maximum of 200ms, well below 60s, at high PRRs, and so always meeting the deadlines. However at low PRRs deadlines are missed. This is due to X-MAC having no information about the PRR, instead periodically waking up every 200ms and delivering any pending packets. In high PRR scenarios X-MAC successfully sends all pending messages. However at low PRRs it does not have enough time to send all the messages, creating a backlog, missing deadlines. With higher values of $\Delta PRR/\Delta T$ DutyCon performs even more poorly, due to the time required to obtain the network state. DDC however takes into account the variance in the PRR, which is higher with larger $\Delta PRR/\Delta T$ values, and so meets the deadline by using more slack.

Figure 4.17 shows the duty cycle of DutyCon and DDC. DutyCon performs well, with relatively low power consumption throughout the experiment, increasing the duty cycle over time to compensate for the poor PRR. DDC however performs best, as unlike DutyCon it does not send each packet in their own transmission window, instead batching all readings and sending them close to the deadline. X-MAC, the approach which best meets the deadlines, has a fixed duty cycle of 8% when the PRR is at 100%, which is substantially

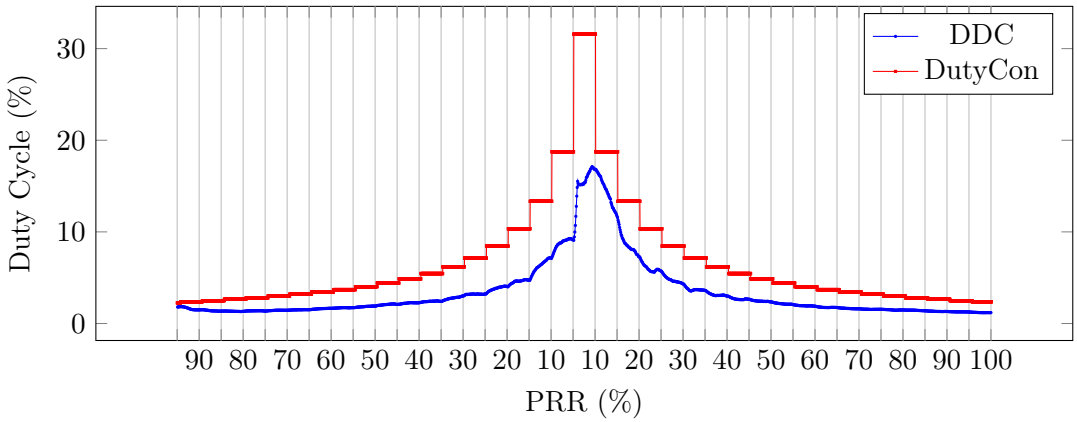

Figure 4.17: Duty cycle over PRR for both of the two adaptive approaches

larger than the other two solutions, wasting power. As X-MAC misses deadlines it is not shown in this figure.

By studying the internals of DDC we can see why the Delay is met at lower PRRs, including the performance of the PID loops. The following four figures show the input and output for each of the PID loops.

The first of these PID loops is the slack controller which aims to keep the slack constant by varying the awake window according to the number of packets transmitted. The slack can be seen in Figure 4.18, with the slack varying more as the PRR decreases, due to the increased randomisation in successful packet delivery. This graph also shows how as the variance in the slack increases, the set-point of the PID loop is adjusted accordingly so that the chance of the slack being exhausted is minimised. Without this adjustment it can be clearly seen that as the PRR is decreased, and the variance increases, the slack would frequently be exhausted. Figure 4.19 is the output of this PID loop, which clearly shows that to maintain this constant slack the awake window is increased as the PRR reduces, requiring more time to successfully transmit packets, with a reduction in the transmission window as the PRR is increased.

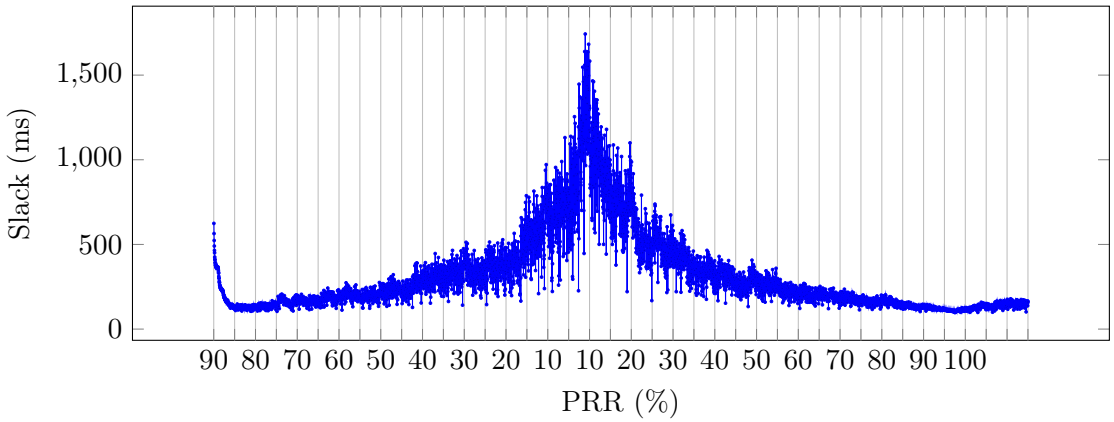

Figure 4.18: Slack over PRR

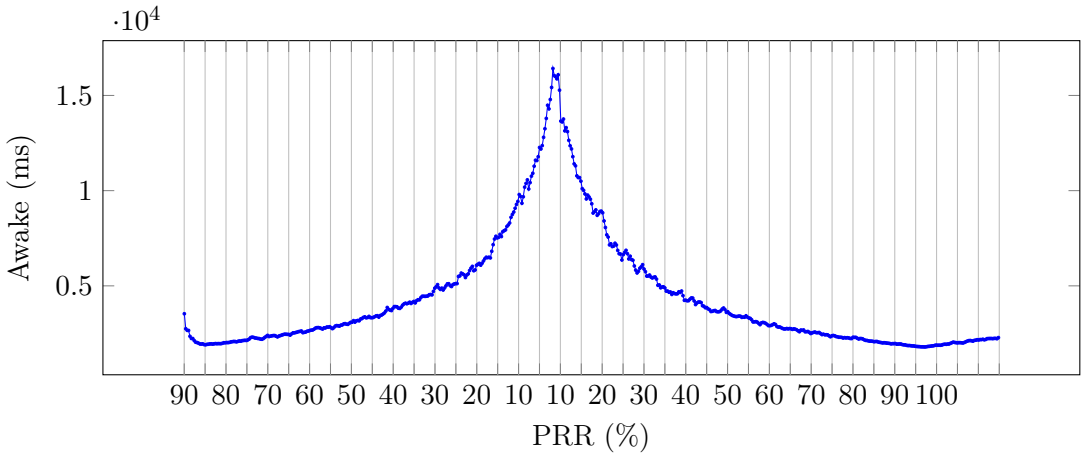

Figure 4.19: Awake over PRR

The second PID loop controls the sleep period to regulate the delay, with Figure 4.20 showing this delay. It can be seen that the delay is relatively constant when the PRR is small, however as the PRR increases, and the variance starts to increase, the delay starts to be reduced as the feedback loop assumes the worst-case PRR from the variance, creating more slack, and thus making it more likely that packets are successfully delivered earlier. Figure 4.21 shows how the sleep period is regulated to allow for the size of the awake window. It can be seen that whilst the awake window is being enlarged, the sleep period is being reduced to ensure that packets sent near the end of the transmission window will not exceed the 60s deadline.

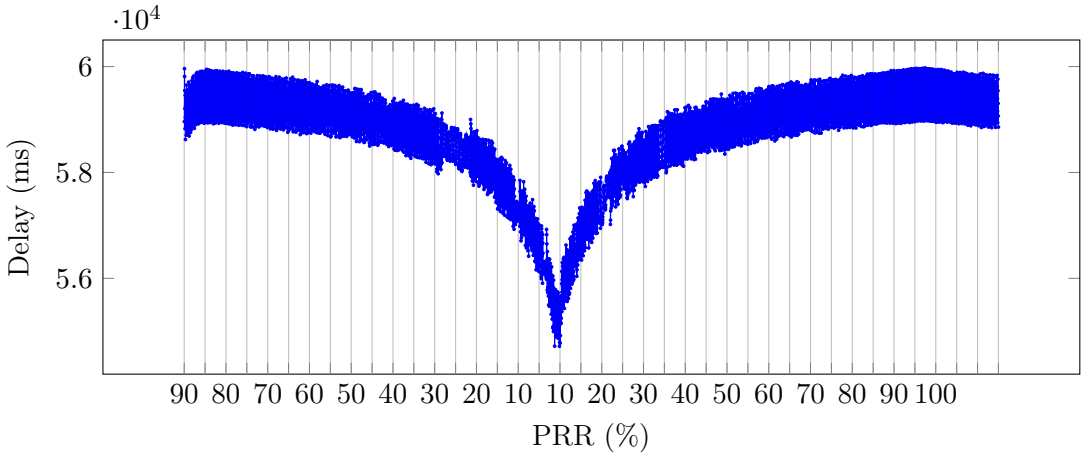

Figure 4.20: Delay over PRR

This section has shown how DDC successfully ensures that the deadline is met over a large range of PRR values. It demonstrates that the feedback-based approach to sizing the awake window and the sleep period allows DDC to accommodate these changes in the environment with no additional external information. Finally it has been demonstrated that by considering the variance in the PRR the slack can be adjusted such that it is

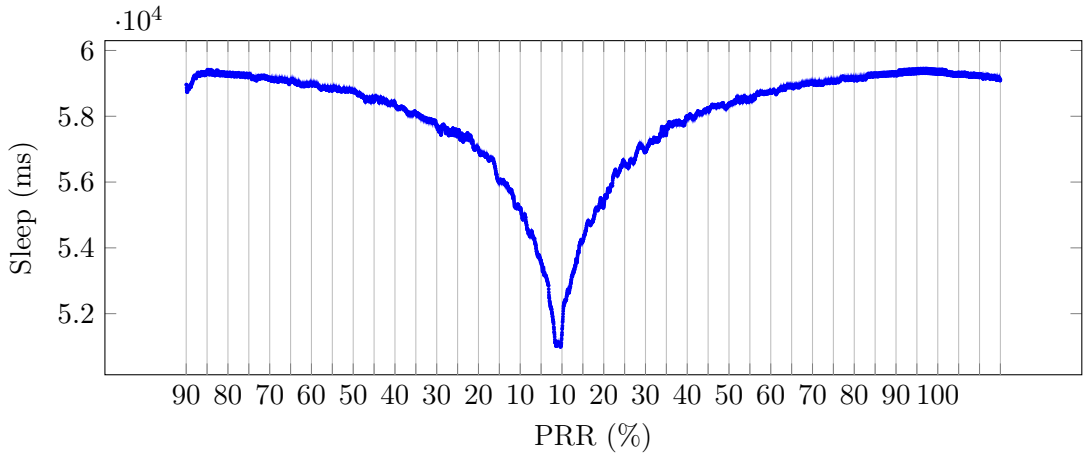

Figure 4.21: Sleep over PRR

unlikely to become exhausted, ensuring that application deadlines are met.

4.3.5 Reactivity Experiments & Results - Population Count

One of the longer-term issues that a WSN must handle is the failure of nodes as discussed in Chapter 3, and as such it is important that any reactive protocols can handle these failures. Secondly it provides an additional opportunity to save power as the reduced number of devices also reduces the amount of radio traffic, which in turn should allow for more energy savings.

As all devices are within one hop of the sink, there are no issues with specific nodes being on a critical communications path needing to relay messages. With no dependency between source devices, there are no requirements as to the ordering in which devices fail, and therefore the focus is purely on how many nodes are alive when a device fails or joins the network. To test the full range of values nodes will be failed individually from a population of 5 until 1 device remains, at which point devices will be added individually until the population reaches the original value. The reason for the low number of devices is due to the ratio of dying nodes as the effect of one node dying upon the network is greatest at low population counts. This is due to 1 device failure in a population of 2 causing half of the traffic to cease, and the traffic to double when it re-joins the network. As the number of devices approaches 5 this effect is smaller, and thus larger values should not reveal any more information. Therefore the maximum population is set to 5 devices, whilst still providing meaningful results. The algorithm for these tests is given in Algorithm 2. During these experiments the same parameters as in Section 4.3.4 will be measured throughout the experiment.

Algorithm 2 Population Testing

$\psi = RootNode$
$\Omega = AllNodes$
$\Omega = \Omega \setminus \psi$
while $\Omega \neq \varnothing$ **do**
    $x = random(x) \in \Omega$;
    $failNode(x)$;
    $\Omega = \Omega \setminus x$
    $Wait()$
end while
$\Omega = \Omega + \psi$
while $\Omega \neq AllNodes$ **do**
    $x = random(x) \in (AllNodes \ / \ \Omega)$;
    $wakeNode(x)$;
    $\Omega = \Omega + x$
    $Wait()$
end while

Figure 4.22 shows the delay that both DDC and DutyCon experience as changes in the population count occur. X-MAC is not shown on this figure as it meets the deadlines in all possible cases due to its high duty cycle. It can be seen that during the failure of any devices the deadlines are always met. This is due to the traffic decreasing, allowing transmissions to occur as scheduled.

As nodes are added, DutyCon has issues due to the ratio of new nodes to existing nodes being high. This means that large changes to the sending schedule within DutyCon are needed, which in turn requires the network state to be gathered before the new schedule can be calculated. At lower ratios DutyCon correctly handles the inclusion of new devices due to the smaller changes in the number of new messages added. DDC on the other hand has no issues with devices being added as the slack provided can accommodate the increase in traffic.

Figure 4.23 shows the duty-cycle of the two approaches as the population is decreased and increased. As with the previous section the step-like behaviour of DutyCon can be easily seen. Each time a node has failed, and DutyCon has global information which reflects this, the appropriate duty cycle is calculated and then used. DDC on the other hand can be seen to smoothly reduce the duty cycle as nodes fail, and as nodes are added to the network the duty cycle increases as more messages are generated. It can be seen that in all but the lowest of node counts DDC performs using a lower duty cycle than DutyCon, which is a direct result of the larger number of messages in the network, and thus the higher efficiencies gained by batching.

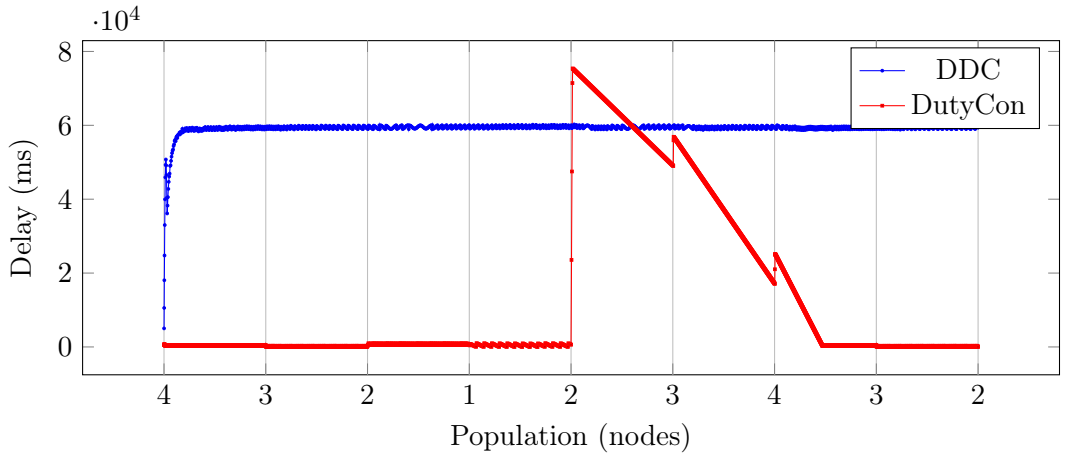

Figure 4.22: Delay over population for the two adaptive approaches

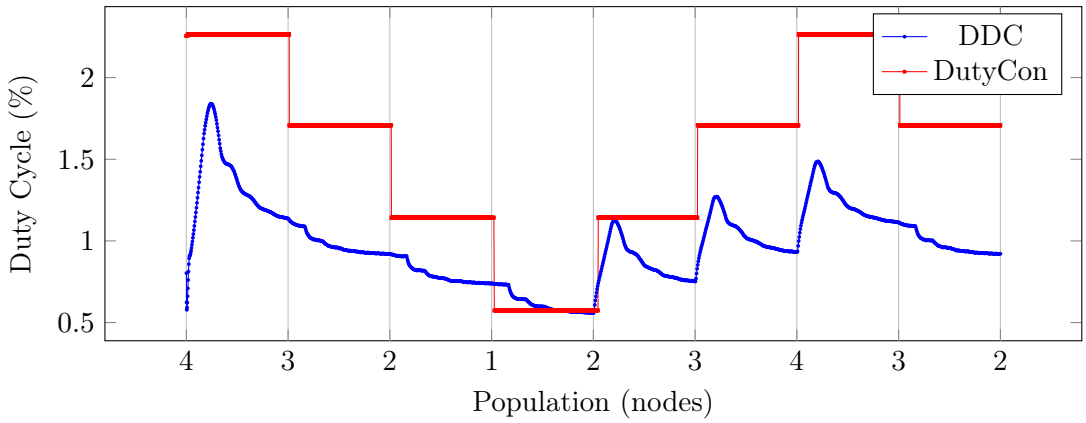

Figure 4.23: Duty cycle over population for the two adaptive approaches

To assess the outcome when DDC slack is too small, further tests were performed. In these tests the slack setpoint was changed from 200ms as in the previous case to 50ms.

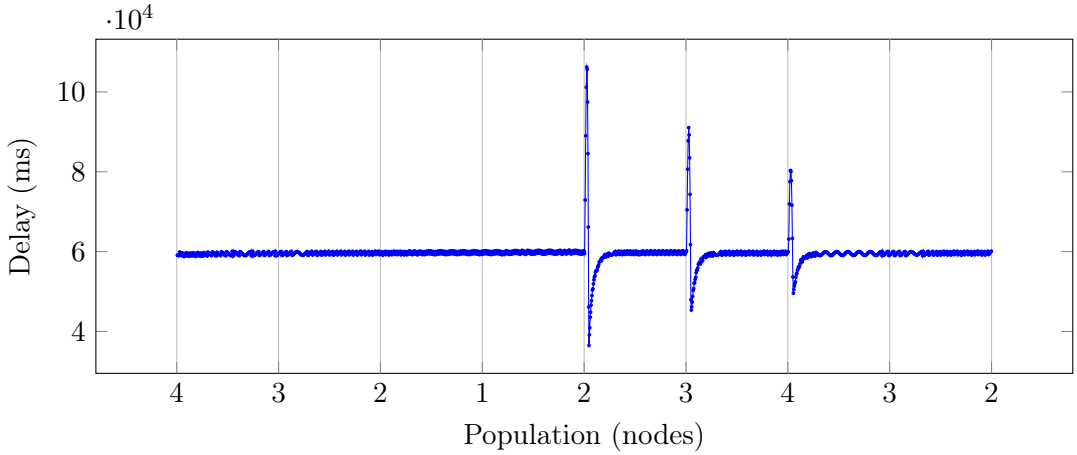

Figure 4.24: Delay over population with changes in node count

Figure 4.24 shows the results of this experiment. In this figure we can clearly see that the deadline is exceeded as more nodes are added to the population. The reason for this is shown in the following figure.

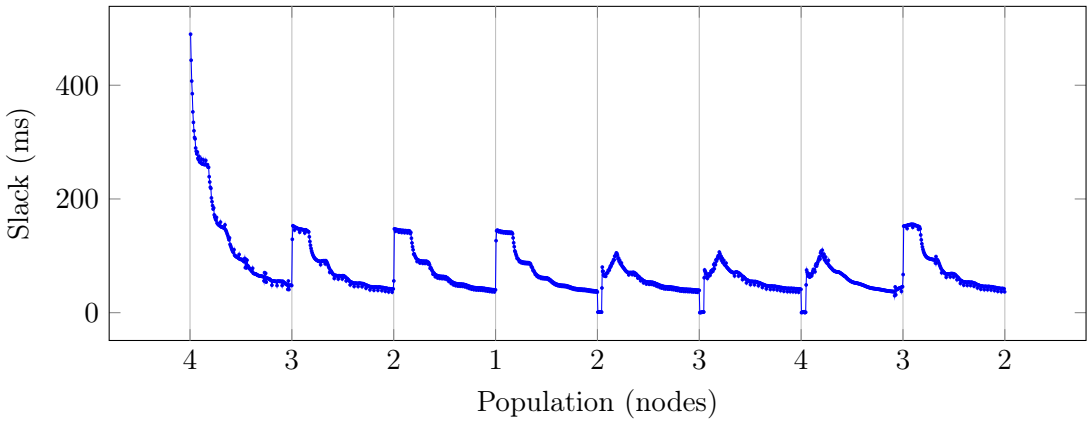

Figure 4.25: Slack over population with changes in node count

Figure 4.25 shows the slack for the same time period. From this figure it can be clearly seen that as nodes die more slack is created, with the PID loop reducing the slack before the next node fails. As nodes are added to the network the number of packets they are generating is larger than the allocated slack, therefore it can be seen that the slack reaches 0, causing packets to miss their deadlines.

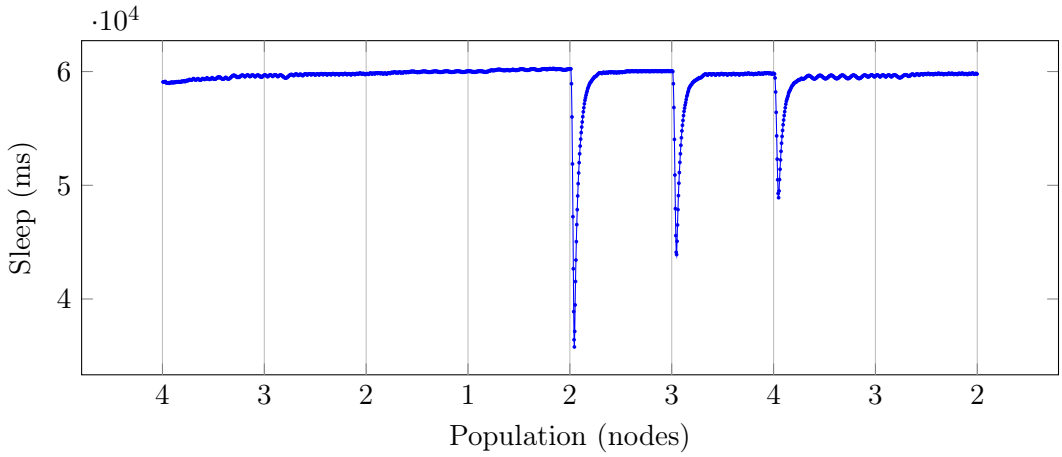

Figure 4.26: Sleep over population with changes in node count

The sudden response by the sleep PID loop can be seen in Figure 4.26, with the sleep period being rapidly reduced in order to reduce the delay experienced. Finally, Figure 4.27 shows how the awake period is adjusted to the initially decreasing number of packets, reducing the awake time, and then increasing the awake time as more devices send data.

This section has shown that during the failure of single motes and the addition of single devices DDC correctly meets the application deadlines though the use of the slack. This slack also allows DDC to meet deadlines in cases where the alternative DutyCon fails, such as the initial case where one additional node is added to a network consisting of only one node. It has also been shown how, in more extreme cases, where the addition of a device (or multiple devices) exhausts the slack, DDC rapidly responds to correct the missed

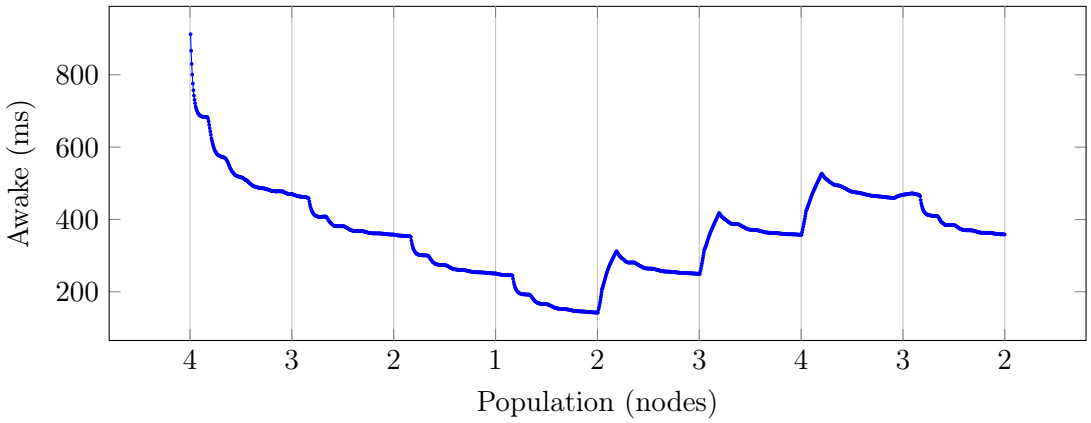

Figure 4.27: Awake over population with changes in node count

deadlines.

4.3.6 Reactivity Experiments & Results - Power Estimate

The previous two sections have evaluated the performance of the three approaches in terms of delay and duty-cycle, however whilst duty-cycle provides us with a relative comparison between the approaches, it does not give us an accurate indication if the network lifetime requirements from Chapter 3 are being met. To solve this issue the results from the three approaches can be analysed, in conjunction with accurate power readings from real devices, to obtain a power consumption estimate. This consumption estimate can then be used to calculate the expected network lifetime. As the sink is assumed to be hard-wired to a power source, a source node is analysed.

As identified within the literature review radio state transitions take a non-zero amount of time to switch between power off and on states, during which time a higher level of power is being consumed when compared to the radio being off, but no radio transmission can be achieved. In addition to this the literature review identified that power consumptions in the radio-on mode also vary depending upon whether the radio is in a receive or send state. For these reasons power evaluation not only requires the amount of time spent with the radio off and on, but also the number of state changes of the radio, further enforcing the idea that duty-cycle alone cannot be used. As the numerical simulator already records the full sequence of actions, this information is already available.

Finally as the power consumption of an idle node is also non-zero, two additional approaches are also evaluated, radio fully off, and radio fully on. These two modes will be used to provide a power baseline for both the minimal amount of power that can be possibly consumed, and also the maximal amount of power that can be consumed (as

radio-on in RX mode is the highest power state in the system).

Approach	Radio off-on #	Radio tx (s)	Radio rx (s)	Power (mAh)
Radio-On	1	86	86400	22.02
LPL	777600	1123	14774	4.01
X-MAC	518400	1123	5702	1.70
ContikiMac	696774	432	929	0.33
DutyCon	172800	432	440	0.21
DDC	26105	432	390	0.19
Radio-Off	0	0	0	0.00

Table 4.3: Power consumption for the 5 approaches under numerical simulation.

It can be seen in Table 4.3 how DDC consumes a lower amount of power than DutyCon and X-MAC. The main reasoning that DDC consumes less power than DutyCon, whilst having similar duty-cycles, is the lower number of transitions that DDC performs. This reduction in the number of transitions is a direct result of DDC's transmission approach, using a single transmission window to send all the data, with the oldest packet being close to its deadline, maximising batching. DutyCon on the other hand sends each packet individually, spaced out over the acceptable tx window, leading to a greater number of radio cycles per deadline period.

When considering the power consumptions it is important to note that all non-DDC approaches have the same power consumptions if the deadline is changed. For most of the protocols this is due to them being unaware of the deadline, or in the case of DutyCon whilst it is only used in the calculation of the transmission period, but as one packet is sent per period, the same power is consumed. DDC however, due to its batching of packets based on the deadline, reduces the power consumption if the deadline is longer, and increases it as the deadline is reduced.

4.4 Cooja Simulation

As Section 4.3 validates the basic theory behind the DDC approach, demonstrating its performance in relation to alternative approaches, it remains to be proven with real Routing, MAC layers, and operating systems. In addition a method of monitoring the actual radio-on time, including the radio states under these conditions is required as DDC's main goal is to constrain the radio-on time. For these reasons Cooja has been chosen based on the review of simulators in Section 2.9, as it can output a log of radio events, and is

one of the few simulators that can run TelosB binaries. Cooja also has one of the most advanced radio models out of the simulators under review. The decision to use the application binaries in the simulator aims to ensure the maximal level of consistency between the simulations and the any future physical deployments. Another advantage to this approach is that it removes the need to re-implement the algorithm for a specific simulator which would raise concerns about the accuracy of the implementation. It was decided to deviate from TinyOS as the operating system and use Contiki for all further experiments, primarily due to its close integration with Cooja, but also due to the higher performance of ContikiMAC when compared to other approaches.

The first tests are run with only single-hop communications so direct comparison with the numerical simulations can be performed, assessing only the effects arising from the introduction of the new Physical and MAC layers. Tests on PRR and Population Count are performed identically to that from Section 4.3 to allow direct comparisons between the obtained results in both numerical simulation and Cooja. Tests with multi-hop communications are performed later in Section 4.4.4 to assess DDC under more typical operating conditions, including the effect of the routing layer upon DDC.

4.4.1 Packet Reception Rate

Initial tests are undertaken to assess the similarity of the simulator results to those obtained under numerical simulation in Section 4.3. The method used remains the same, varying the PRR from 100% to 5% and back in increments of 5%. Throughout the experiment the delay experienced by the packets is recorded, with Figure 4.28 showing the results of this experiment.

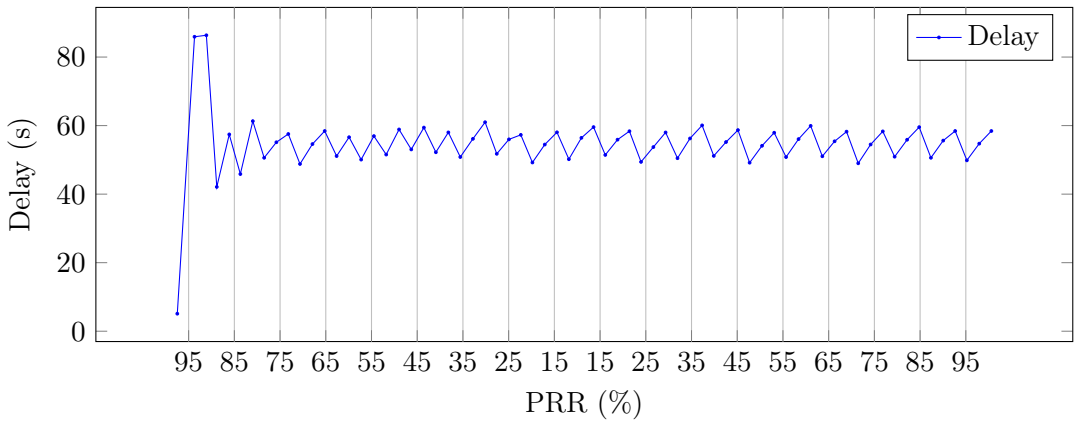

Figure 4.28: Packet Reception Rate over Delay

From Figure 4.28 it can be seen that the delay experienced is similar to that obtained

under simulation, with the delay being close, but importantly less than 60 second deadline throughout all PRR levels. One interesting difference is that the slight reduction in delay at high PRRs experienced in the PRR is not demonstrated within Cooja. Upon inspection of the radio details this lack of variation is due to the simplified assumption in the numerical simulator that all packets from all nodes are simply interleaved and transmitted in rapid succession. In the Cooja simulator this is not the case as the MAC protocol, including backoff, does not fully saturate the channel, allowing fluctuations in transmission timings to be smoothed out, increasing the delay, but importantly not the variance.

As in the simulations the performance of DDC can be evaluated by looking at the duty cycle of the devices as shown in Figure 4.29. This Figure shows the increase in the awake period and the respective decrease in the sleep period in response to the PRR decreasing.

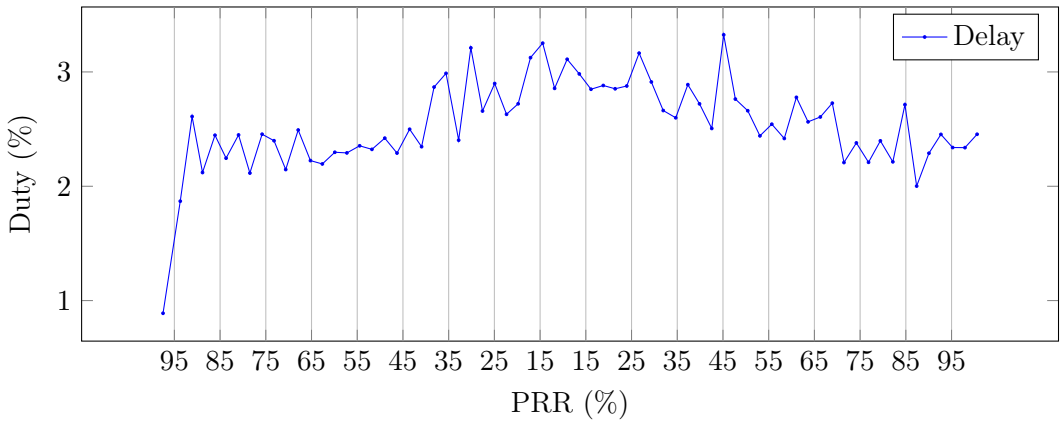

Figure 4.29: Packet Reception Rate over Duty Cycle

The trend in these results matches those shown in the numerical simulation, however the duty cycle is much higher, indicating that the efficiency may be lower than expected. This duty is the awake time imposed by DDC over the sleep period imposed by DDC, however as DDC operated on top of the MAC layer the duty cycling performed beneath is not represented. To analyse this further a single transmission window from within Cooja is analysed. During the awake period, the radio is additionally cycled according to the MAC algorithm, helping reduce wasted power in the transmission and importantly in the slack phase. Cooja also shows that ContikiMAC supports further power savings when a device wishes to send multiple packets, with the synchronisation phase being performed only once, and with multiple packets being sent back-to-back from the source. This indicates that the CSMA/CA causes excess slack, however once transmission begins between nodes more packets can be transmitted at low cost. As shown in the numerical simulation section DDC does not rely on these additional MAC features to save power, however their inclusion

further enhances performance.

This section has shown that not only does DDC exhibit the same trends as shown in the numerical simulator, but that the inclusion of a real MAC protocol does not adversely impact the performance of the algorithm.

4.4.2 Population Count

The second test ran under numerical simulation in Section 4.3.5 measured the response of DDC to changes in the population count of the network. The same experiment is also performed under Cooja simulation, with the 5 devices in the network, with 4 source nodes sending packets to the sink node. The population is reduced to 1 alive source node, before being increased back to the original 4 source devices. Figure 4.30 shows the results of this experiment.

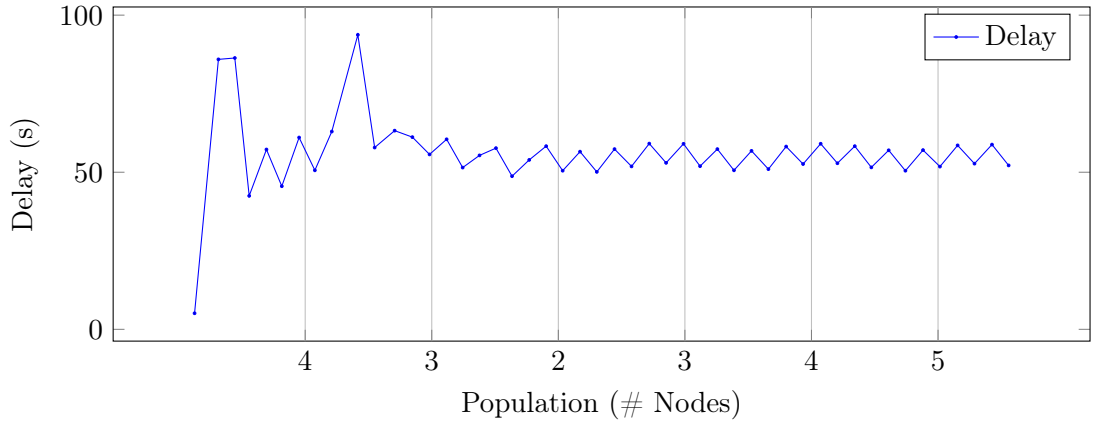

Figure 4.30: Population Count over Delay

This figure shows that the changes in population have no affect on the packet delay, as also shown in the numerical simulation. This is due to the slack handing the number of packets either removed or added to the network as a device is removed or added respectively. In Figure 4.31 the Duty Cycle can be seen to change in a similar manner to that from the numerical simulations, with lower duties as the number of devices, and thus number of packets, decreases.

From these experiments we can see that the numerical simulations are similar to that shown in the Cooja simulator, which in turn is running the same binary which will be executed on real hardware. This reduces the chance of discrepancies between the experiments, allowing for a more accurate validation of the simulated results. Finally the power consumption of these devices is analysed, followed by the additional complications added by using a real MAC and Routing protocol that was evaluated in Section 4.4.4.

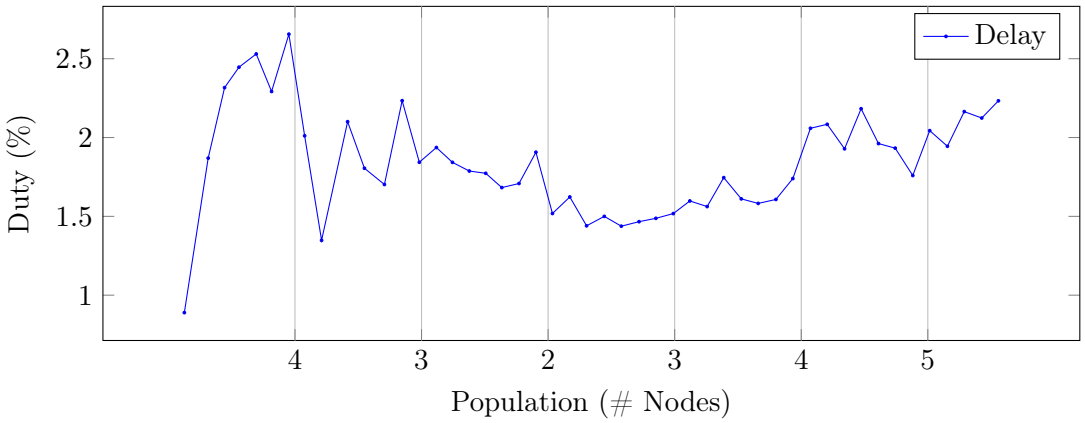

Figure 4.31: Population Count over Duty

4.4.3 Power Estimations

As the Cooja simulation is executing the real Routing and MAC algorithms, the power estimations obtained in Section 4.3.6 can be refined accordingly. Routing adds overheads to the power estimation as additional packets need to be sent to discover routes to the sink before data can be transmitted. The MAC layer, in this case ContikiMAC, will however reduce power consumption by cycling the radio during the transmission window if a specific node cannot transmit the packet. This could be due to simply having no packets to transmit, or other nodes currently transmitting on the same channel. .

To support analysis of the radio duty cycling Cooja provides a log of all radio state transitions for all nodes within the simulation. These state transitions not only include on and off states, but also transmit and receive, which as shown in the literature review, also have different power consumptions. Using these traces allows the same approach used in Section 4.3.6 to be performed, combining both the state and transition information, with power consumptions from real devices, to obtain a power consumption for the simulation. The ContikiMAC case represents the typical use-case, where only the MAC layer cycles the radio to obtain power savings.

Approach	**Radio off-on #**	**Power (mA)**
Radio-On (nullMac)	1	22.23
LPL	814942	4.37
X-MAC	527287	1.81
ContikiMac	720130	0.36
DDC	34026	0.22
Radio-Off	0	0

Table 4.4: Power consumption for the 5 approaches within Cooja.

Table 4.4 shows the results of the power analysis from the Cooja simulations. These figures show that there is not a substantial difference between both DDC with real MAC protocols, and with the figures obtained within the numerical simulation in Section 4.3. Across all the protocols the figures acquired are similar to the numerical simulation, however all are larger. This is possibly due to the lack of collisions in the numerical simulator, whereas within Cooja packet collisions require the re-transmission of packets, reducing the effective duty cycle.

4.4.4 Multi-Hop

All the experiments up until this point have been run under single-hop conditions, where the source and sink are within radio range, with no nodes providing message relaying. Whilst this may be true in small-scale scenarios, and is useful to compare against the numerical simulations, typically WSN devices must communicate over multiple-hops to reach the sink, and therefore investigation as to the effects of hop-distance upon DDC need to be conducted. Unlike the previous scenarios this introduces an additional class of node to the previous source or sink nodes, the relay nodes. Relay nodes are devices which can communicate with other devices, but do not sample information from the environment and are not the destination for any messages. They may be used as either reserve devices should others fail, or to pass on messages on behalf of other source devices in the system. This class can be further broken down into two distinct types, active relays and idle relays. Active relays are nodes which are on a current data transmission route from a source to a sink, requiring that the node transmits and receives messages for other nodes. Idle relays are not on a current transmission route, and therefore do not need to forward any messages. As all idle relays behave similarly, only one is monitored for the experiment. Assuming a single sender and a single sink, all active relays also behave similarly, and therefore only one such device will be monitored for this experiment. If more than one sender was included then there would be multiple levels of relay node depending on the number of upstream sources.

Both the delay, slack, and duty-cycle of the devices will be monitored, with the hop count of the source device being increased and decreased throughout the experiment. The layout for this experiment will be 10 nodes arranged in an oval, with the sink at one tip of the oval. Initially the source will be adjacent to the sink, before being moved around the oval, until the mid point, at which time it returns the same way back to the sink. This

ensures that there is one edge of the oval that experiences no traffic, providing a baseline comparison for the active relay nodes.

Figure 4.32: Packet Hop Count over Delay

The delay remains constant throughout the experiment, never exceeding the 60s deadline as the slack never reaches zero. The reasoning for this can be seen in Figure 4.32 as the the hop count increases the duty cycle is increased. This is due to the awake period needing to be extended to accommodate the additional delay in the packets being transmitted by the intermediate relay nodes.

As in the previous power consumption testing within the Cooja simulator, the radio logs can again be analysed to obtain an accurate estimate of power consumption for the devices. As there are 4 classes of device, each one of these is shown in Table 4.5.

Node	**Radio off-on #**	**Power (mA)**
Source Node	37284	0.24
Sink Node	37284	0.21
Active Relay	37284	0.26
Idle Relay	37284	0.19

Table 4.5: Power consumption for the 4 node types

This table shows that whilst all nodes within the network are using the same DDC duty cycle, those on the idle transmission path, and hence not relaying packets, do not consume much power. Ideally these nodes could be powered down further if knowledge existed about the active routes, however as one of the primary goals of DDC is to have no additional information about the network other than that gathered by the feedback, no further savings can be obtained. In addition the routing behaviour may change between transmission windows depending on the background interference and the routing protocol that is used, making any predictions about the routing topology difficult.

This section has shown that DDC performs in a similar way to that experienced under numerical simulation and that even with the inclusion of MAC and Routing layers, including multi-hop communications, DDC continues to perform as expected.

4.5 Summary

This chapter has defined an adaptive protocol called Dynamic Duty Control which takes a set application deadline, and reduces power consumption whilst ensuring this deadline is met in the majority of scenarios. Unlike alternative approaches, DDC requires no global knowledge, instead using feedback from the application itself to ensure the requirements are met. It has been shown in numerical experimentation and simulation how the DDC adapts to changes in the PRR of the network, and the number of active nodes, ensuring that the deadline is met in all but the most extreme circumstances. From these simulations power consumption has been analysed, showing that DDC consumes lower power than alternative adaptive schemes when the generation rate of packets is greater than the deadline, allowing packets to be batched. In all circumstances DDC performs better than traditional approaches, as the worst-case performance is bound by the best-case of the underlying MAC protocols.

DDC aims to be independent of the underlying Routing and MAC protocols, but due to DDC enforcing that the radio remains of during its sleep period some protocols may work less effectively, however these can be mitigated by using MCWs as defined in Chapter 5. Protocols that require periodic messages to be sent in order to maintain network state, such as proactive routing algorithms, will fail to send these messages during the DDC sleep period. If the algorithm cannot handle such failures then the routing algorithm can be presented to MCW as an application with a specific application deadline. These messages must be extended to include the required DDC timestamps and DDC must be explicitly notified when a routing protocol message arrives to allow the processing of its feedback loops. Modelling the routing protocol as an application ensures that DDC, in combination with MCW, will periodically schedule a transmission window of the correct size for the routing protocol. A drawback of this method is that all transmission windows, regardless of their size, will have a constant overhead in the form of the slack, and as this slack is the same for all transmissions, even with a single routing message, considerable radio-on time is wasted.

As the MAC layer also provides additional duty cycling there is typically a number of

additional parameters that can be tuned to tailor the MAC layer performance. Normally it is assumed that the MAC protocol will never be disabled, and therefore a trade-off between lifetime of the network and the responsiveness of the MAC protocol can be achieved through the notion of a polling frequency. The higher this frequency the more responsive the MAC protocol, however at the expense of more power. When DDC is used the radio-on time is automatically restricted and therefore energy expenditure can be a lower priority for the MAC protocol. With DDC more power savings can be realised if the data can be transmitted to the sink sooner, allowing DDC to turn off the radio sooner, and therefore the priority for the MAC protocol should be speed of transmission to the sink with only minimal focus on efficiency. When performing online tuning of the MAC protocol the polling frequency should be increased, which in turn will cause the DDC awake window to decrease. At the point where the awake window remains constant or begins to increase, the polling frequency is either successfully saturating the transmission window, or the higher frequency is causing more collisions, reducing efficiency, taking more time to successfully transmit messages, and a local optimum has been reached.

DDC may also provide an opportunity to perform data aggregation and compression due to the implicit batching of packets. This may allow for further power reductions in a manner that is adaptive to the application deadline and the environmental conditions. A limitation with the current system is that it is assumed that the application will always have a single specified deadline, however there are a number of multi-modal applications which may require different deadlines under different circumstances. In addition this work has only focused on a single application existing within the network, whereas it may be desirable to support multiple applications simultaneously, each with different deadlines. Both modes and multiple applications will be investigated in Chapter 5.

Chapter 5

Mode Change Windows

Chapter 3 analysed the dependability of a data-sensing WSN, and presented Dependability Assurance, a method to ensure that any dependability requirements are met at run-time. DA however raises concerns over the run-time of WSNs, identifying that the mean lifetime of WSN devices needs to be over 10 months to provide a maintainable deployment. Chapter 4 addressed this concern by presenting DDC, optimising network lifetime for a specific application by using a deadline provided by the operator, reflecting the maximum delay that the application can tolerate. DDC however assumes that there is only one application within the network, and that the deadline specified for any application remains the same throughout. The application designer may choose to change the deadline, however doing so invalidates DDC's prerequisite that these remain fixed, removing the guarantee that under reasonable conditions the deadline will not be missed. DDC also only considers a single application running within the network at any given time, and therefore further work must be performed to support the multiple application use case. The set of possible deadlines that an application may choose to use, in combination with a specific data-generation rate, are herein referred to as modes.

Depending on the application, the impact of an application missing its deadlines may vary. In some circumstances such as monitoring the movement of occupants, the data may need to be acted upon quickly, requiring that the deadline is met as much as possible. Other scenarios such as environment monitoring may have more relaxed deadlines, where missing a small number of deadlines may be acceptable. In all of these cases the justification for ensuring missed deadlines is not only that one deadline may be missed, but that more than one deadline may be missed. In the case where we move from a low criticality mode with a deadline of 1 hour (to report room temperatures), to a high criticality mode with

a deadline of one minute (when an individual device detects a fire and wishes to report readings rapidly), there may be up to 60 deadline misses before the mode change is enacted. In this case missing multiple deadlines is clearly unacceptable, and therefore a mechanism needs to be provided which allows these deadlines to be met.

This chapter presents Mode Change Windows (MCWs) to ensure that deadlines are met when multiple applications are run within the same network, with each application having multiple modes. This is especially important as some applications may move from large deadline mode (low criticality) to a small deadline mode (high criticality), requiring that data be delivered sooner than it otherwise would have been. Early delivery of data is problematic as the decision to change mode may be on a device which currently may be unable to communicate due to the network being in a radio-sleep mode. The objectives for this chapter are as follows:

McwObj1 - Given an application with a set of modes, with each mode specifying a deadline and data-generation rate, a policy can be defined which ensures deadlines are met in an energy efficient manner.

McwObj2 - Within the same deployment multiple applications must be able to run simultaneously. These applications must still meet their deadlines and be able to change modes at will.

Section 5.1 provides an overview of the problem in more detail, followed by the proposed MCW solution. Section 5.2 provides model checking of the basic MCW mechanics to ensure that the desired properties hold over all possible scenarios. Section 5.3 implements MCW for the same target hardware as Chapter 3 and similarly evaluates the solution under cycle-accurate simulation. Finally Section 5.4 draws conclusions from this work.

5.1 Overview

Within WSNs multiple applications may run simultaneously on the same device, each with a distinct mode which defines its deadline and packet-sending rate. An extended version of this is where a single application may change modes at-will, not only changing the current deadline, but also the data-generation rate. When using only traditional MAC protocols such as ContikiMAC, where the radio is cycled several times per second, deadlines and data rates are less of a concern. Fast radio cycling allows delays to be minimised, allowing

deadlines to be easily met. In addition to timing fast duty cycling also removes the need for large data buffers as packets can be sent with minimal delay, requiring only small queues. In cases where the radio is cycled much more slowly, with sleep periods on the scale of minutes, the time delay to inform the sink of the new mode may be considerable, during which data may be generated at a new, faster rate than that seen initially. This raises important concerns, as not only must deadlines be met in all circumstances, but with increased amounts of queued data come greater transmission times, possibly exceeding the transmission window size. To address this issue it is proposed that before data is sent the mode-changing device alerts the sink at the first possible opportunity, which in turn broadcasts appropriate information (such as new duty cycles) across the network to support the new mode. This section will analyse what information is required to support such mode changes, whilst meeting the deadlines for the new mode, which may possibly be shorter than the previous mode.

Clearly, within Wireless Sensor Networks, no hard deadlines can be completely guaranteed in all circumstances as ultimately the communications can be interrupted by external events. Within this section it is assumed that deadlines are guaranteed to be met under the condition that the external interference, either caused by internal network communication or external sources, does not vary by a large amount over time (radio jamming) and no abnormal external events occur (large numbers of devices simultaneously failing). Another assumption that must be made in order to guarantee that deadlines are met is that the deadlines themselves are known ahead-of-time. This does not indicate when the deadline will be used, just that the application may require this deadline to be met at some point during its execution. This assumption is required, as without prior knowledge of the operating modes it is easy to construct a case where the system is in deep sleep state, and an application may decide that it has a deadline before the network wakes, with no other devices being awake notified, making any such deadline unachievable.

To motivate this work an example of an assisted living facility will be used, where residents are monitored through a series of static devices placed around the environment, and a number of mobile devices attached to the individuals.

It is noted that whilst the mobile device may send data at a lower rate when the occupant is stationary (as the actual data in this mode may not have much variance), this does not indicate that the device samples outputs from its sensors at a lower rate. Instead the sensors can still be sampled to monitor for movement, triggering a move into

a higher power mode should movement occur (producing readings that may be of interest to the sink device), but this mode change should occur quickly to meet the tight deadline. This chapter is motivated by the classification techniques identified in the literature review Section 2.6.2, where raw data from moving occupants can be used to classify events such as changes in location, but delivering such data is expensive from a communication aspect, with the value of the data being low when the occupant is stationary. This chapter assumes that there are user-provided methods for detecting the current mode, and therefore mode changes can be derived. Despite this detection of the current mode, it is also assumed that the raw data is important to the sink node, and therefore that all readings should be reported. These assumptions are met by methods such as classification algorithms, where running a trained classifier is computationally cheap, and can be performed on the motes to detect the current mode. However training such a classifier requires a large amount of computational power, possibly including a large amount of historical readings from all devices in the network, and therefore should be conducted by the hard-wired sink node. One such example would be location tracking as discussed in Section 2.6.2.

An overview of the proposed applications and their modes, with associated data-generation rates and deadlines is given in Table 5.1. These values are not taken from real deployments, however we believe that they accurately represent a standard assisted living scenario. As identified earlier a special case may occur as some applications may not be operating in the network at a given time. These absent applications are represented by defining a special absent mode, in which the generation rate is 0 and the deadline is ∞, with a mode change to this state should the application cease operating. This allows all possible conditions to be captured in the mode table.

Application	**Mode**	**Sample Rate (Hz)**	**Deadline (S)**
Environment Monitoring	Off	0	∞
	Low	0.1	60
	High	0.5	10
Occupant Tracking	Off	0	∞
	Low	0.01	60
	Medium	0.5	10
	High	2	2

Table 5.1: Both assisted living applications, the respective modes, and their settings

Transitions between events can occur at any time in the network between any two modes. When a mode is changed the data-generation rate of the devices may change, with

more or less messages being generated than in the previous mode.

To conduct analysis into the worst-case scenario for the above modes a number of initial assumptions must be made. Firstly it is assumed that modes with a large number of packets to transmit to the sink will take longer to transmit than others with less data. Secondly the round-trip-time for a mode change is known, this is the time that it takes to inform the base station of the intent to change mode and the time that it takes for the base station to inform the network of the change. These assumptions are relaxed in Section 5.3.

If the worst-case scenario for deadlines is assumed, where data is transmitted just before the respective deadline, then the deadline and the generation rates can be used to calculate the amount of data that is queued for sending as shown in Equation 5.1.

$$PacketsGenerated = Deadline/GenerationRate \tag{5.1}$$

Using this formula the worst-case sleep time can be derived as shown in Equation 5.2.

$$SleepTime = Deadline - Tx(PacketsGenerated) \tag{5.2}$$

If it is assumed that the mode change occurs as soon as the radio is turned on, then in the worst case scenario a mode change may occur as soon as the sleep period has been entered. If the *SleepTime* of the new mode is smaller than the *SleepTime* of the old mode then the deadline will be exceeded. This scenario is shown in Figure 5.1, with Figure 5.2 showing the best-case scenario where the deadlines are met. In these figures the transmit slot labelled M is the mode change request, with M' being the broadcast mode change. The labelled arrows are the generation times of the associated messages, with the respective boxes containing the labels being the transmission of the messages.

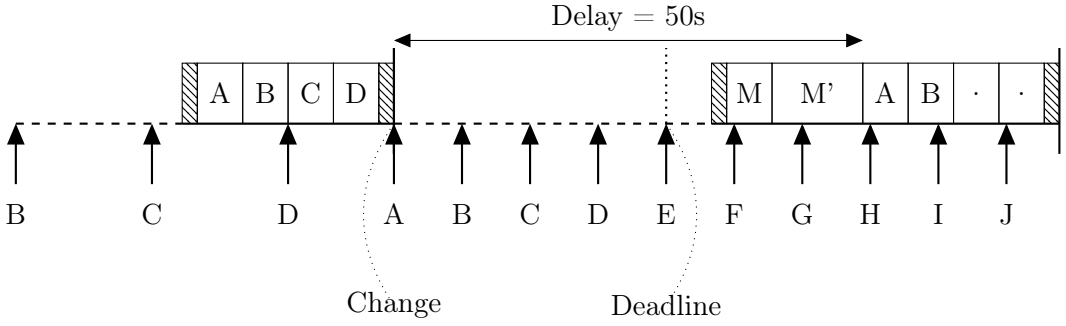

Figure 5.1: Application running without MCW, with a worst-case mode change occurring causing missed deadlines.

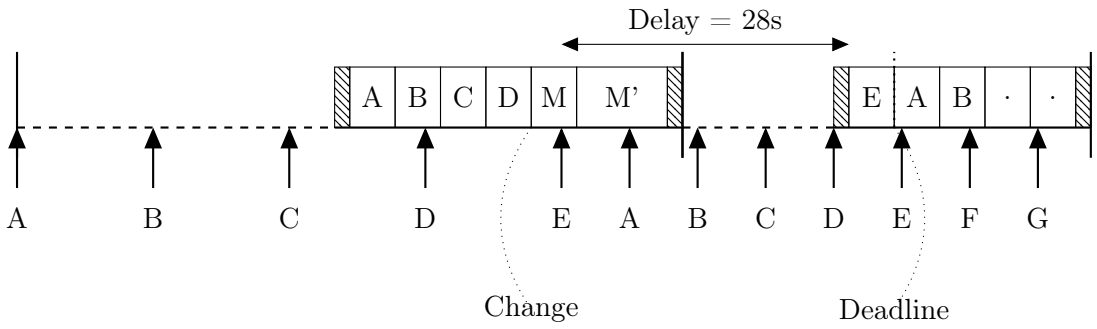

Figure 5.2: Application running without MCW, with a best-case mode change occurring allowing deadlines to be met.

One simple solution would be to fix the sleep time for all modes to one set value, the minimum sleep time of all possible modes, however this would cause all modes to awaken and send data early, defeating any benefits that may arise from batching transmissions (power savings, data aggregation, etc.). Therefore a new transmission window is introduced, the MCW, which allows mode changes to occur if necessary, but otherwise is not used, as shown in Figure 5.3. Only if a node needs to request a mode change should this window be used, initiating a mode change, sending the old data, and then sending the new data, as shown in Figure 5.4.

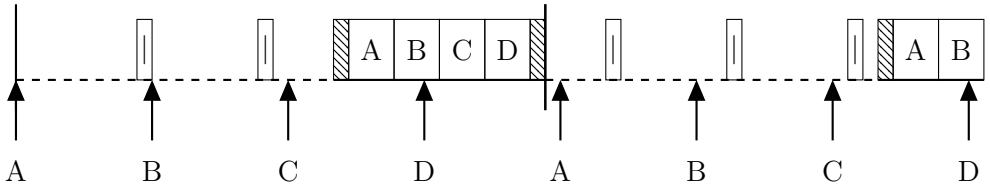

Figure 5.3: Application running with MCW without a mode change occurring.

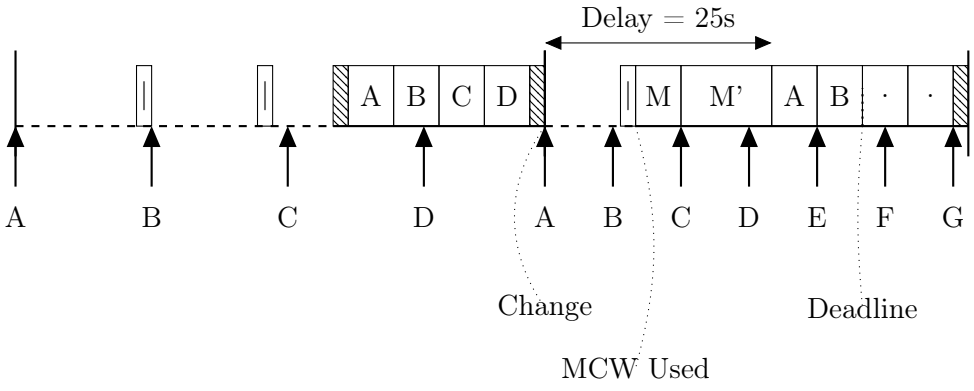

Figure 5.4: Application running with MCW, with a mode change occurring early in the sequence.

The frequency of these windows must be calculated to ensure that all deadlines of all modes can be met, and therefore it is derived as shown in the following formulae. Any such mode change needs to ensure that in the worst case, all deadlines are met. The worst case is the entering of a new mode just as the sleep period begins. Therefore mode changes must be complete before that shown in Equation 5.1.

$$modeChangeDeadline = min(sleepPeriod) \tag{5.3}$$

As a mode change involves sending a mode change notification, followed by other devices sending all queued packets if any are available. The maximum time for a mode change to take place is therefore shown in Equation 5.4. In this equation the maximum awake time is taken as the worst case time to send all possible queued messages. This factor could be reduced further, as the awake period is based upon the sleep period, which is $min(sleepPeriod)$, however as these factors cannot be simply scaled due to the interaction with other components such as batching at the Routing layer, $max(awakePeriod)$ is used as a worst-case starting value. This scaling is due to Routing and MAC issues meaning that half the number of generated packets may not necessarily take half the transmit time.

$$modeChangeTime = max(awakePeriod) - informTime \tag{5.4}$$

These two factors can be combined to calculate the required period as shown in Equation 5.5.

$$MCWperiod = modeChangeDeadline - modeChangeTime \tag{5.5}$$

Scheduling a MCW at the period of $MCWperiod$, assuming that the awake time $informTime$ is set to a sufficiently large value such that the mode change can be enacted in time, ensures that all deadlines are met even in the presence of mode changes.

As stated within MCW Objective 2, multiple applications running on the same network must be supported, either multiple copies of the same application or completely distinct applications. As the same application has the same deadline, all devices running the same application are scheduled to awake at the same time and all use the same transmission window to communicate with the sink. This allows for maximal aggregation opportunities and also avoids the need for the motes to wake up numerous times for the same application. Other applications however do not share the same deadline, and thus, depending on the

start time, may have a transmission window that collides with another application. In these circumstances, as it is known which applications run at which times, window avoidance must be performed.

Window avoidance takes the window with the latest absolute deadline as the baseline, and moves all other transmission windows earlier until they no longer collide. Once the window with the latest absolute deadline has been successfully scheduled that it no longer collides with any others, the process is repeated with the remaining windows until all collisions are avoided. Offsets must be applied so that the windows occur earlier than scheduled, as this will not cause deadlines to be exceeded, whereas transmission occurring later than intended could cause deadline misses. It is also noted that it is not just this one instance that is advanced earlier, all future instances must be offset by the same amount. This global offset to a window period ensures that no delay is propagated into later transmission windows, as if the original schedule was returned to without careful thought, it would occur later in time and thus has the potential to cause missed deadlines. To avoid the constant drifting backwards in time of all colliding windows, the original schedule time is used when sorting windows by their absolute deadlines. Once a single window has been moved, its previously used deadline is evaluated to check if the windows have been effectively moved earlier in time, and if so a MCW is additionally scheduled at the original time. The algorithm used to implement this is outlined in Listing 3, and whilst it may appear computationally expensive, for the majority of the time there should be no collisions and thus most of the algorithm is avoided. When avoiding collisions by performing re-scheduling of transmission windows there may be a scenario where no collision-free schedule exists. In this case the number of messages, and thus the transmission time, has exceeded the available transmission time. In this scenario there is no solution which will meet the required deadlines, even without the use of MCWs, and thus deadlines will be missed.

This section has outlined how MCW handles the single application mode change scenario by introducing MCWs. These MCWs ensure that should a mode change be required there is suitable time to inform the network, flush any pending data from the old application, and then send any messages generated from the new application. Importantly the timing of these MCWs ensures that the deadlines of both the old and the new modes are not violated due to the mode change. Secondly this section has described how multiple applications can be handled with respect to colliding transmission windows, ensuring that the deadlines of either applications are not adversely affected by the other applications.

Algorithm 3 MCW Collision Avoidance

```
toSchedule = currentEventItem
currentScheduleOffset = 0
while toSchedule ≠ ∅ do
    sortByDeadline(toSchedule)
    X = toSchedule.pop()
    collisions = calculateCollisions(X,currentSchedule)
    if collisions = ∅ then
        addToSchedule(X,getNextTime(X) + currentScheduleOffset)
        if currentScheduleOffset < getPrevOffset(X) then
            addToSchedule(MCW,getNextTime(X) + getPrevOffset(X))
            currentScheduleOffset = currentScheduleOffset - MCW.duration
        end if
        currentScheduleOffset = currentScheduleOffset - getDuration(X)
    else
        removeFromSchedule(collisions)
        toSchedule = toSchedule + collisions + X
    end if
end while
```

To ensure the correctness of these solutions the following sections will check that deadlines are met using both model checking and real experiments.

5.2 UPPAAL Model Checking

To verify that the proposed solution to scheduling MCWs is correct, the UPPAAL model checker [17] is used to ensure that in no circumstances can the deadline of any mode be missed during a mode change. By using a model checker an exhaustive but efficient search can be conducted over the search space which would be expensive to perform in simulation. Simulation also does not guarantee that all the edge cases will be exercised, and therefore does not guarantee that in our scenario deadlines will never be missed.

For MCWs model checking provides assurance that in the given scenario the deadline is guaranteed to be met and that buffers will not overflow. This guarantee is not strictly required as in some circumstances a deadline miss might be acceptable, however this guarantee ensures that the more critical case of multiple deadline misses in succession, which may happen without MCWs, will not occur. Buffer overflows are also important to avoid as it indicates that the transmission buffers have not been fully cleared in the appropriate time. This failure to clear the transmission buffers means that the backlog might continue to grow, eventually filling the available buffer size and begin to drop packets. This packet overflow, and thus the dropping of packets, is therefore presented as an error state within

the model to ensure that it will not occur.

The drawbacks of using model-checking to explore these issues is that the accuracy of the model checking relies on the accuracy of the underlying model. As this model is a generalisation of the physical scenario, any results from the model may not fully represent the behaviour in the physical deployment. In addition to the fidelity of the model there are many possible sets of parameters that could be used, however exploration of all possible states would take prohibitively long to explore, and therefore only a limited number of all possible parameter sets can be explored.

UPPAAL is a model checker based upon the concept of timed automata, which includes a model simulator, allowing for quick and easy verification that the implementation behaves as desired. Checking that the model is deadlock-free is performed internally by UPPAAL by performing reachability analysis on the deadlock states, ensuring that from the initial state there are no possible transitions that allow us to enter the deadlock state. This model however must make a small number of simplifications which will be revisited in Section 5.3. These simplifications are as follows:

1. The time that it takes for all devices within the network to be informed of a mode change is bounded by a known value.

2. The time that it takes to successfully transmit a single data packet to the sink is bounded by a known value.

3. In all nodes the radio must be on for the full time that it takes to inform the network of a mode change, otherwise the mode change will not succeed.

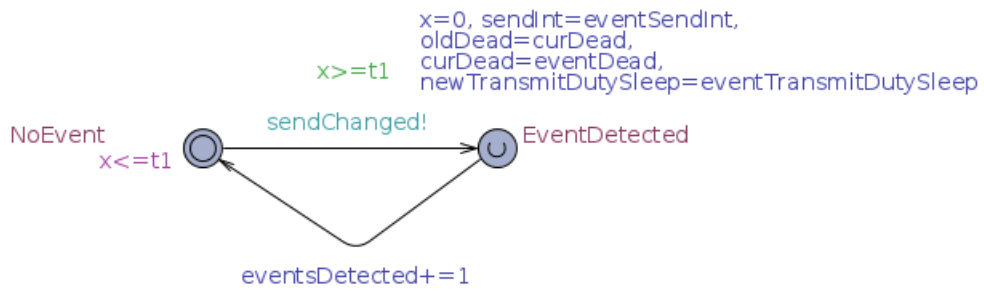

Figure 5.5: Mode triggering based on events, in this case a set time.

The model in Figure 5.5 shows how at a set period $t1$ a mode change event occurs. This mode change event triggers a *sendChanged* event, which is representative of the system-wide mode change broadcast message. As the mode has now changed the the generation

rate of packets, current deadline, and the pending sleep time are updated to the new values in the new mode. The new values for these parameters are immediately used. The sleep time however is a pending sleep time, meaning that only at the next transmit window the pending sleep time will become the current sleep time. This pending status is due to the delay between the mode change occurring at the source node and the point at which it would have informed the network, which in turn would cause a new sleep period to be used.

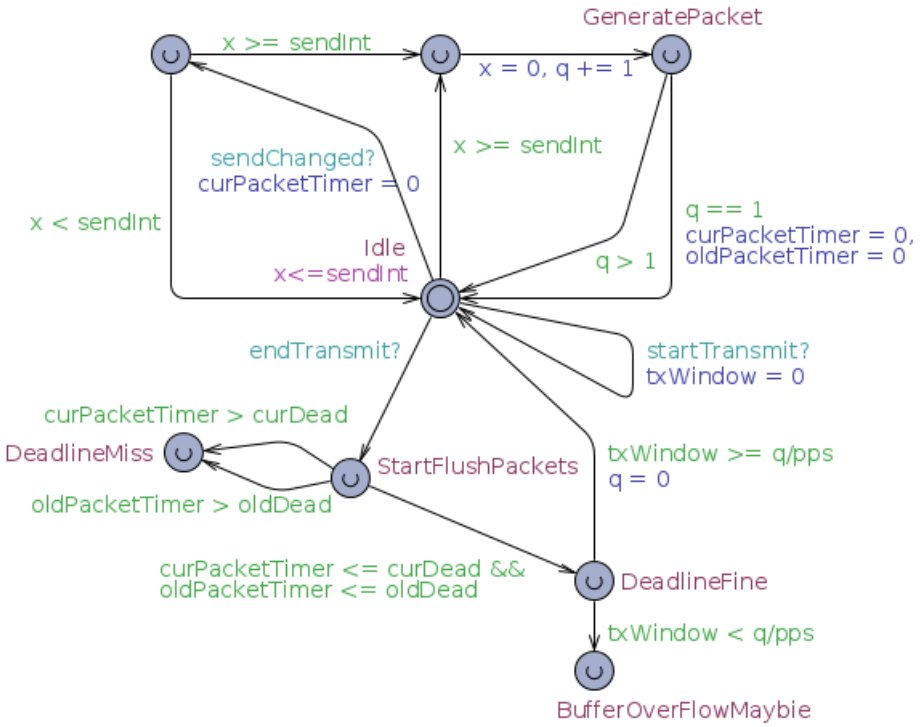

Figure 5.6: Sensor periodically sampling the environment generating packets, transmitting the data when possible.

Figure 5.6 shows the constant generation of packets at periodic time *sendInt*, with the *timestamp* of the oldest packet being recorded. Should the radio have just switched into the transmit state the awake time is recorded, with the deadline correctness being checked at the end of transmission followed by a check that the transmission time was large enough for all the queued messages. The transmit time check is necessary as with a mode change taking place, changing the deadline and the packet-generation rate, a situation could occur where the transmit window is not large enough, causing some packets to be undelivered.

Figure 5.7 shows the simple cyclic behaviour of the MCWs, requesting that the radio be turned on and off cyclically. Should there be a pending mode change, signalled by an event detection, a mode change event is signalled to the rest of the software components.

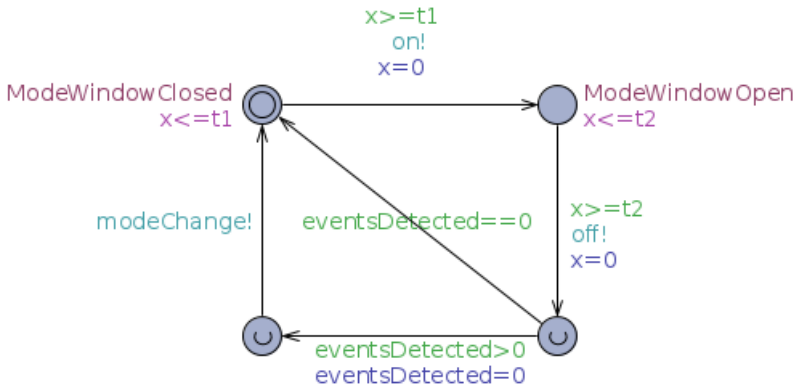

Figure 5.7: Mode change windows periodically cycling the radio and initiating pending mode changes.

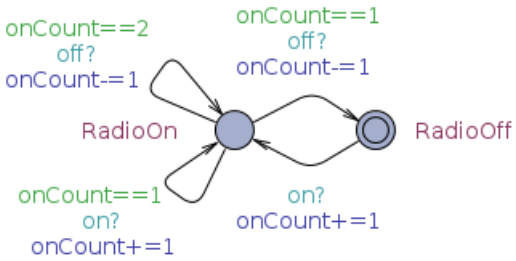

Figure 5.8: Radio's true state based upon requests from other software components.

Figure 5.8 tracks the individual radio state requests and ensures that if there are any outstanding radio-on requests, the radio is kept on until all requests are satisfied.

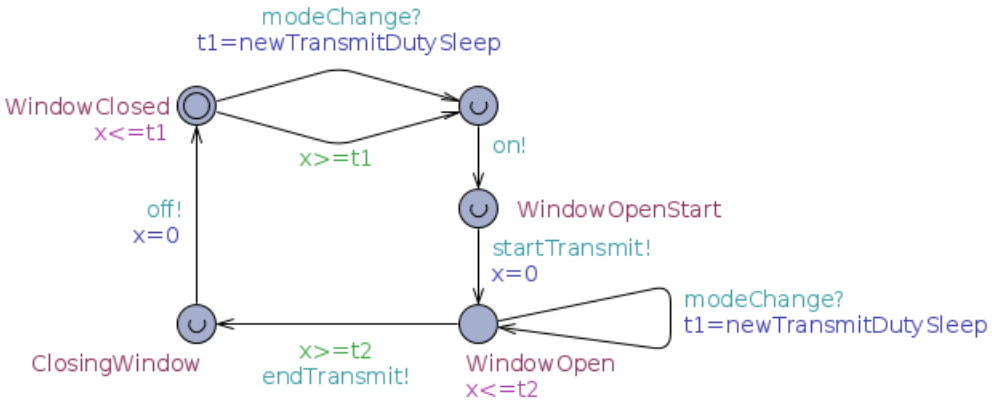

Figure 5.9: Transmission of packets and enacting of mode changes.

Figure 5.9 handles the duty cycle of the radio when mode changes occur, including the signalling of the start and end of the transmit window to the other software components.

In order to ensure that in no case the system can enter an erroneous state, such as the deadline being exceeded, these erroneous states have been modelled with no exiting edges

as shown in Figure 5.6. This allows for the system as a whole to be checked in a single deadlock-free pass, with the state in any found deadlock being the error that occurred. These states are either a deadline being missed, where the current timestamp on a packet is greater than the value of the absolute deadline, or a buffer overflow, where more packets have been generated than the mote can hold.

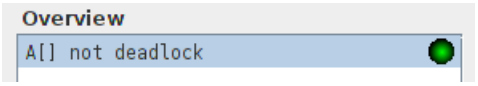

Figure 5.10: The results from checking the presented model for deadlocks.

The results from deadlock-checking the system can be seen in Figure 5.10 and this demonstrates that a deadlock cannot occur. This provides assurance that none of the issues above can arise. To ensure that it is MCW which allows this deadlock-free behaviour to exist MCW was removed from the model and the deadlock check ran again. This second run failed to be deadlock-free, showing that the deadlock may occur by the application entering the $DeadlineMiss$ state.

This section has shown that the application of MCW ensures that under the assumptions stated, the system can never miss a deadline when transitioning between two modes, as the MCW provides an opportunity for old readings to be flushed from the sources and the new mode started, meeting the primary MCW objective 1 for this chapter. As there are three outstanding assumptions from this section these will be analysed in the following section, showing how they can be removed through a combination of feedback from the network and the application of real MAC protocols.

5.3 Evaluation

Section 5.2 demonstrated that the underling MCW approach allows transitions between any two modes whilst ensuring that the deadlines of both modes are met. This section however made a number of simplifications about the properties of the network with respect to the radio. These simplifications are used to ensure that the radio is active for enough time for these events to complete and that the deadline has been met. Within the real environment these values can be estimated, however external factors such as environmental noise and radio collisions may cause messages to be retransmitted, invalidating or making any timing estimates pessimistic. To solve this issue the timing requirements for the transmission and sleep windows could be greatly relaxed, accounting for the worst-case

interference, however obtaining an estimate for these windows is still an issue. Another approach is to use adaptive protocols that measure the interference on-line and use this to inform the network layer, reducing waste.

As in the UPPAAL model, the evaluation will initially use a fixed size for the MCWs to demonstrate that it is not dependent upon any other solutions. Later in the evaluation it will be demonstrated that by using DDC to size the MCWs, further power savings can be achieved. DDC has been chosen due to the low power requirements, and because its one requirement, the provision of application deadlines, is available in this scenario. This protocol ensures that the radio is on for the minimum amount of time possible to meet the deadlines. MCW does not depend on DDC, however as shown in Chapter 4 DDC provides the highest amount of power savings possible when the application's deadline is available. By using DDC it can be ensured that within a specific mode the data is received by the sink before the appropriate deadline, however DDC, being a feedback-based approach cannot, without some modification to the application, be used for sizing the MCWs, as typically no mode changes occur, and thus there is no data to operate the feedback. For this reason an approach similar to DA in Chapter 3 is used, with mode change events being simulated, allowing feedback to occur and DDC to operate correctly.

As both transmission of normal data and the MCWs are periodic, commonly using different periods, at some points these windows may collide. In this circumstance we require that the normal transmissions do not block any required mode changes, as it is assumed that only a single MCW is required for a successful mode change. Secondly collisions between transmission windows of separate applications may also occur (only the same applications use the same window). To alleviate this issue, as all periods and durations are known at run-time, should a collision be due to occur, window-avoidance is performed as outlined in Section 5.1.

As applications can be dynamically added or removed from the network, we cannot rely upon all devices having global information about all the possible applications and their possible modes. For this reason we only have this information on the device which starts the application or initiates the mode change, with this one device knowing the deadline and the data-generation rate. At the earliest opportunity this requesting device informs the sink, which in turn broadcasts the mode across the network to be cached by all the devices. This scenario relies heavily on the dynamic nature of the feedback to quickly adapt to the new mode, to obtain optimal duty cycle settings. Once a mode has been

experienced the appropriate PID settings will be cached, allowing for future mode changes between previously-known modes to be performed quickly. This ensures that no prior information is required, however at the expense that the initial mode change may occur late, missing deadlines, and that the initial time in the new mode may also miss deadlines whilst the network adapts to meet the application requirements. This is not a limitation of our approach, as if the typical approach of providing prior information to the devices is chosen then these deadline misses would not occur.

5.3.1 Cooja Simulations

To verify that the conditions experienced within the model checker still hold when used with real applications, simulations are performed. Simulations were specifically chosen as they provide detailed traces of the radio behaviour, allowing communications at each hop in the network to be analysed in detail.

To assess that the objectives for this chapter are being met, a number of experiments will be conducted:

1. Assess mode changes without MCWs. Based on the original analysis some deadlines may be missed when undertaking the mode changes.

2. Assess mode changes with MCWs. Monitoring of the deadlines should show that deadlines are met at all times.

3. The overheads that MCWs incur should be evaluated. This should be observable in the difference between the effective duty cycle of both the non-MCW and MCW experiments.

4. Demonstrate that the addition of multiple distinct applications does not affect the meeting of the application deadlines.

5. Show how the inclusion of DDC to size the MCWs provides some power savings.

Within these experiments the application that will be tested is the Occupant Tracking application. To begin the experiments a warm-up phase is conducted whereby all modes are cycled though, with 20 transmission rounds occurring in each mode. This is necessary so that the awake and sleep periods for each mode can be learnt by the network. After this warm-up period the results are recorded, with the current active mode switching between the Low and Medium modes at a random time interval between 0 and the deadline, after

every 4th transmission window. This ensures that a large spread of possible interleavings between modes is explored. When a mode switch is initiated the internal PID loops for each of the modes are swapped, ensuring a rapid change between the modes whilst still allowing the network to adapt to external interference unlike the case where fixed values are used.

5.3.1.1 Single Application, No MCW

The first experiment is to demonstrate that there is an issue with deadlines when changing between two modes. Within previous chapters applications have always remained within a singular mode, and therefore the feedback mechanism has been relied upon to provide changes in behaviour. This has ensured that in previous cases, as long as the environmental changes are gradual, the network can adapt. In this scenario mode changes are sudden, with large changes in deadlines, making a feedback approach infeasible.

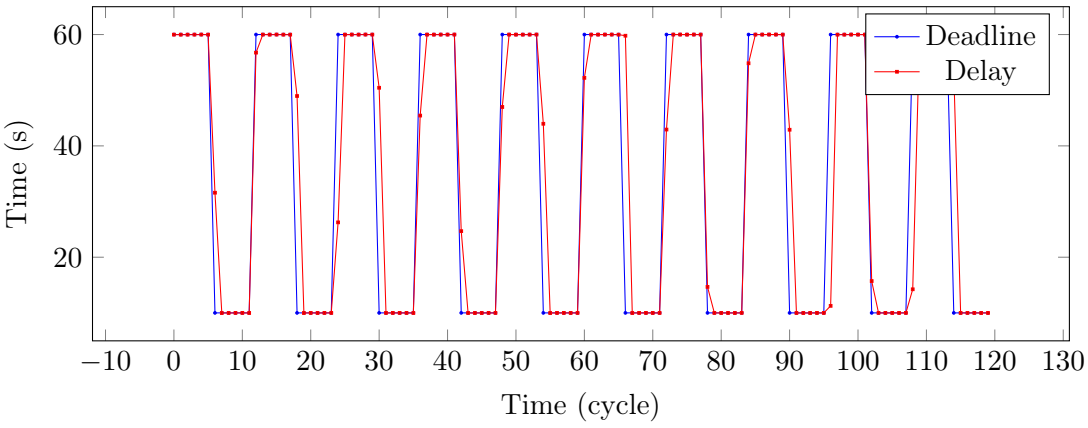

Figure 5.11: Deadline and Delay over time with MCW off

Figure 5.11 shows how an application with no MCWs constantly fails to meet its deadlines when moving from a large sleep mode to a smaller sleep mode. To avoid skewing the figure the X axis is not shown in terms of runtime, as this would cause all the small period modes to be compressed, instead showing the number of awake periods that have elapsed. In this figure the blue line shows the relevant deadline at that point in time, with the red line showing the maximum delay experienced in that transmission window. If at any point the red line is above the blue line then a deadline miss has occurred, which can be clearly seen in the majority of high to low period transitions. The reasoning for these deadline misses is that as the mode is changed from the 60s deadline mode to the 10s deadline mode, the node must still wait for the next alive period from the 60s mode before it can inform the sink of the mode change. This wait is required as all devices use these periods

to turn off their radio, and thus cannot be contacted outside of these periods.

5.3.1.2 Single Application, With MCW

As shown in Section 5.3.1.1 deadlines can be exceeded if the mode changes before the sink can be informed, and the new mode requires a shorter response time from the sink than the previous mode. As outlined in Section 5.1 MCW provides a method for scheduling mode changes which can be used to inform the sink as to changes in the network state, allowing data to be flushed from the sender, and the new mode entered. This has been shown to hold through the use of model checking, however this section verifies that this is correct in the presence of real MAC and Routing layers.

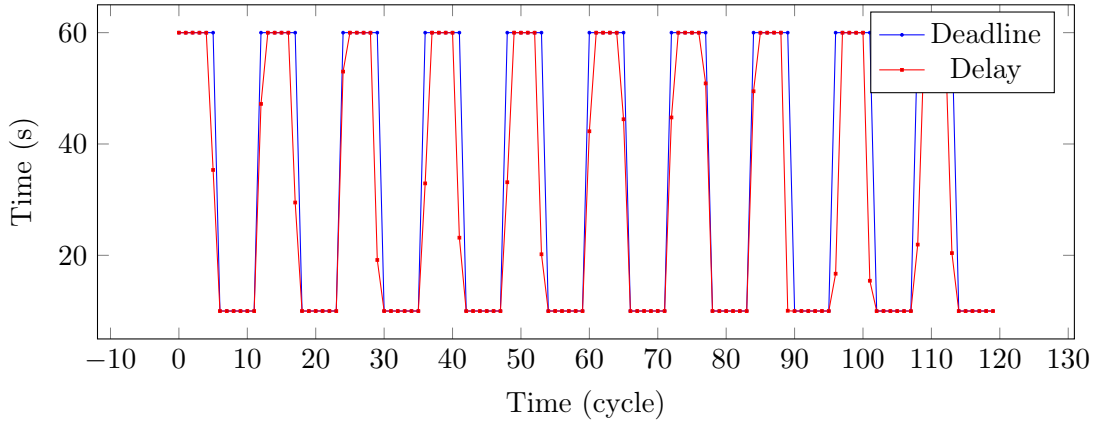

Figure 5.12: Deadline and Delay over time with MCW on

Figure 5.12 shows the results of this experiment. As shown in these results the deadline is never exceeded by the delay, with the delay being reduced before the new deadline is enforced. The reasoning for the reduction in delay before the new deadline is enforced is that MCW allows the sensing node to request a mode change from the sink at the next MCW interval after it has performed a local mode change. At this MCW interval the old packets are flushed from the source node, which by definition is before the normal transmission window, which in turn leads to a reduction in delay, before the new mode is then enacted. One side effect of scheduling MCWs is that less time is spent in the radio-off state, as all devices must be ready to relay messages in this period. For this reason concerns about efficiency are raised which will be addressed in the next section.

5.3.1.3 MCW Overheads

As MCW adds additional transmission windows to mote schedules, the effective duty cycle of the motes is reduced, representing higher power consumption. This section takes the

same application as in Section 5.3.1.1, both without MCW and with MCW, and records the duty cycle of the two approaches.

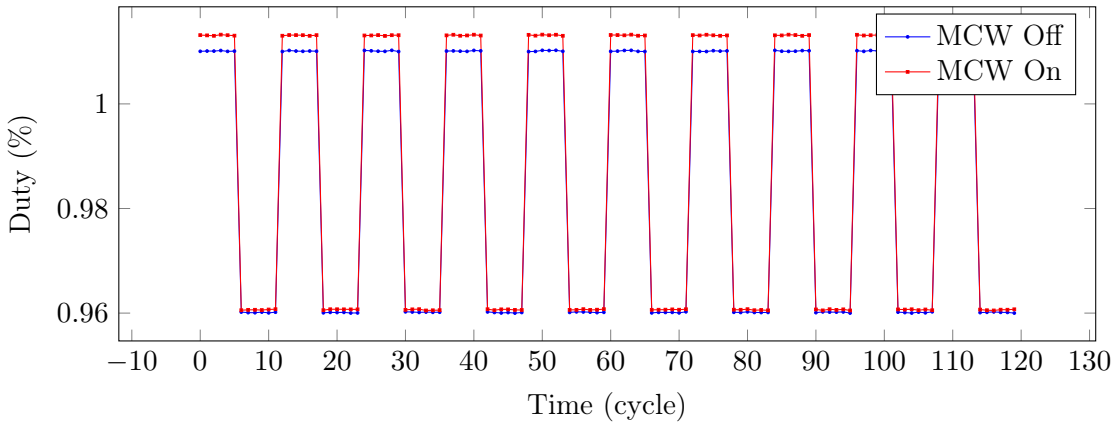

Figure 5.13: Duty cycle with and without MCW

Figure 5.13 shows the results of this experiment. It can be seen that the overheads associated with MCW are larger in larger deadline modes. This is due to larger deadline modes experiencing more MCWs than smaller modes. These overheads are a direct consequence of both the time to inform the sink of a mode change, and the frequency difference between the current mode and the smallest mode. As the time to inform the sink has been statically chosen, any pessimism in the duration will be wasted radio-on time. However should the MCW be set too small, changes may be unable to be performed, allowing deadlines to be missed.

5.3.1.4 Multiple Applications

To demonstrate that MCW works when multiple applications are present further tests need to be performed. As these tests should demonstrate that collision avoidance works effectively to ensure that deadlines of either application are not missed, collisions between transmission windows must occur in the experiment. To ensure that this is the case the periods of the two tested applications should not have any common factors, ensuring that both cycles do not become in-phase with each other, avoiding collisions. For this reason the periods of the applications deviate from those shown in Table 5.1, with the Low mode having a deadline of 61 seconds and the Medium mode having a deadline of 59 seconds. These periods are both prime to ensure no common factors and are both very close to one another so that multiple collisions will occur in successions. Within these experiments no mode changes will occur, instead the two applications will simply co-exist in the network, with MCWs still being scheduled in case a mode change should be required.

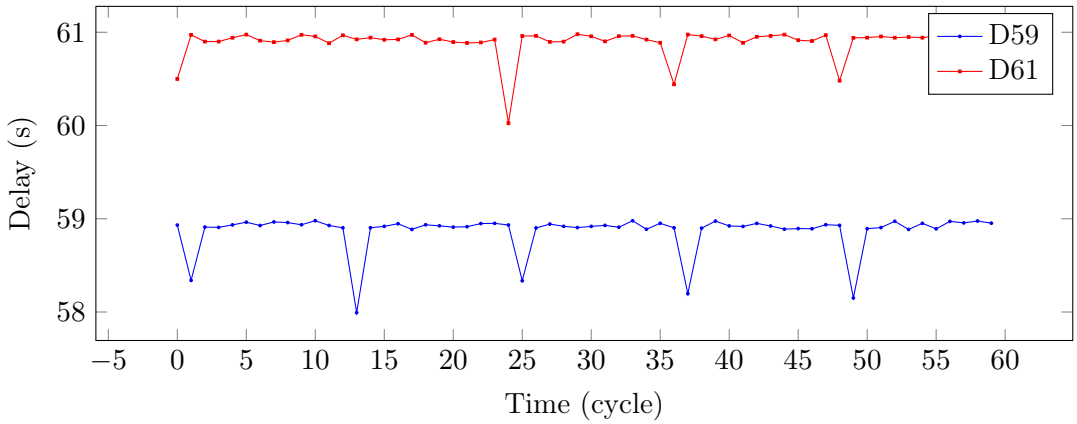

Figure 5.14: Delay of two applications over time with MCW

Figure 5.14 shows the results of the delay measurements for both applications. It can be seen that for the majority of the time the delay is always less than the respective deadline, however at one point in the figure the delay temporarily drops further, first for the larger deadline application, then the smaller. The reasoning for this is that collision avoidance moves transmission windows earlier in time if a collision occurs, causing the delay for that window to be reduced. Initially the frequent window collides with the end of an infrequent window, causing the infrequent window to be moved earlier. Eventually it is the infrequent window that collides with the end of the frequent window, causing the frequent one to move.

To ensure that this is the effect of MCW's collision avoidance, the same experiments were conducted with no collision avoidance.

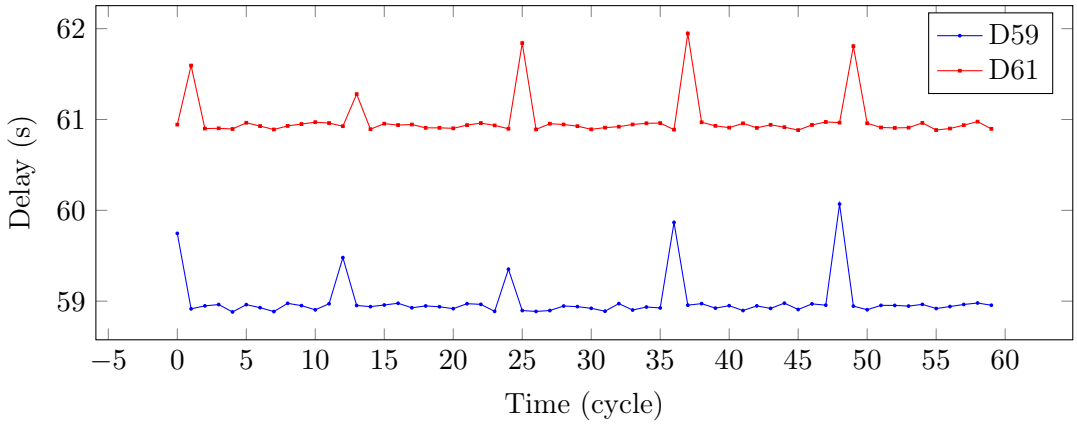

Figure 5.15: Delay of two applications over time without MCW

The results of this experiment can be seen in Figure 5.15. In this figure it can be seen that the delay is periodically exceeded, first in the high frequency application then in the lower frequency application. This is due to the collision causing the other application to delay transmission until the first application has completed its transmissions. As the

applications are out of phase this affects the highest frequency first, followed by the lowest.

5.3.1.5 Dynamic Sizing of MCWs

For these experiments the MCW size has been set to a fixed size of 20ms. This was in order to demonstrate that in the case where the maximum time to send a message to the sink from any node is followed by a subsequent network-wide broadcast of the new mode, then this value can be used. In the majority of cases a pessimistic estimate can be calculated, however this pessimism leads to wasted radio-on time, and thus wasted power. For these reasons this experiment shows that by using a feedback-based approach as shown within DDC, this value can be directly measured, reducing the pessimism and thus increasing efficiency. To measure this value simulated mode changes are randomly sent within MCWs, with the resultant broadcast across the network informing nodes that a simulated event has occurred. When source nodes next send information to the sink the slack at each individual node is reported, with the maximum value being taken as the input to the MCW controller to calculate the MCW size.

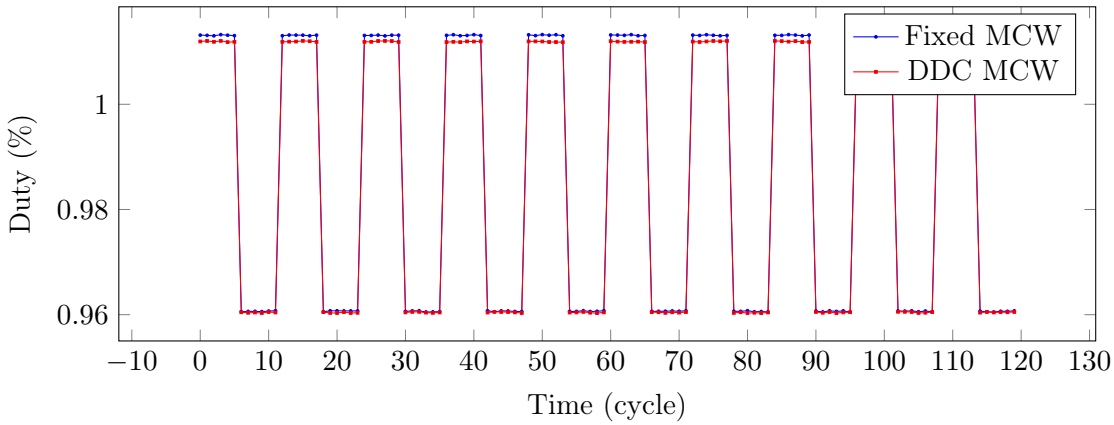

Figure 5.16: Duty cycle with and without DDC sizing the MCW

The results of this are shown in Figure 5.16. This figure shows that a MCW with a dynamically-sized window is more efficient than a fixed size window. This however is only the case as the fixed window size is pessimistic, and should some method exist for calculating a tight worst-case size it may be more efficient, however such a method would be deployment-specific.

5.4 Summary

Previous chapters have assumed that a network has only one application and that this application has a singular mode which specifies the deadline and packet-generation rate. This chapter aims to remove these assumptions and has shown that through the use of MCWs the original objectives for this chapter can be met.

The first objective states that deadlines for an application must be met even in the presence of mode changes. It has been shown through the use of model checking that MCW ensures that deadlines will always be met whenever mode changes occur. Further evaluation of this objective has been performed under the Cooja simulator. This has shown that under real MAC and Routing layers MCW still ensures that deadlines are met, with the removal of MCW showing how delays exceed the specified deadlines. The overheads that MCW places on the network in terms of reduced duty-cycle has also been shown.

The second objective states that multiple applications should be supported. This objective has been met by the inclusion of transmission window collision avoidance to MCW. This ensures that should two transmission windows collide they are rescheduled such that the delays are only reduced, and thus the deadlines are never exceeded. This has been evaluated in the Cooja simulator, showing that the lack of collision avoidance causes deadlines to be exceeded should two application transmission windows collide.

Chapter 6

Conclusion

The performance of WSNs is highly variable as their deployments are typically ad-hoc in nature, their hardware is low cost, and there is limited battery power available. This variable performance, along with proposals that WSNs may be a viable solution within assisted living scenarios, raises concerns about the underlying reliability of such systems. This thesis presents a structured approach to analysing and increasing the super-class of reliability, Dependability, which includes other factors such as availability, safety, integrity, and maintenance. Importantly dependability introduces the concept that some level of failures may be acceptable, acknowledging that failures cannot be completely removed and instead must be mitigated.

This thesis presents three complementary sections, all of which can be either used independently or jointly. These sections provide the following benefits: assurance in the run-time correctness of the system; reduction in the power needed to provide the service; and maintaining correctness when changing service requirements at runtime. They are summarised as follows:

6.1 Contribution 1 - Dependability Assurance

Dependability Assurance provides a systematic method for ensuring that an application is dependable at run-time. This is performed by decomposing an application into a number of Dependability Tests, some of which must be periodically checked at run-time. This approach provides the following benefits:

- It ensures that the application will be supported when required, either under low traffic event-driven applications, or under high traffic sense-and-send approaches.

- It provides confidence that should any failure occur with a WSN mote that affects the application, it will be reported to the operators in a timely manner.

- The operators will be informed if the monitoring system itself has failed, and thus the status of the network is unknown.

- It provides information about the current state of the network which can be used to inform maintenance policies or other algorithms such as DDC.

This contribution meets Objective 1, allowing the dependability of the WSN to be verified over the lifetime of the system. This includes the first application of SHARD to a WSN, and the derivation of tests from these results.

6.2 Contribution 2 - Dynamic Duty Control

Dynamic Duty Control takes any timing requirements from the application, in combination with feedback from the network, and uses these to reduce the duty-cycle accordingly. This approach reduces the power consumption by minimising the radio-on time and additionally batching packets. This provides the following:

- Power is saved by using the application deadlines to minimise radio-on time and to reduce overheads by maximising batching.

- Deadlines are met by using a feedback based approach which monitors the delay at run-time, requiring no external information such as the MAC protocol used or routing topology.

- Fluctuations in the surrounding environment are handled by the introduction of slack, variance in slack, and the use of feedback.

- In event-driven systems, DDC can use the status information provided by DA to support low activity applications.

Dynamic Duty Control meets thesis Objective 2, providing power savings by taking into account the specified application. This has also been shown to work with some typical MAC and Routing protocols.

6.3 Contribution 3 - Mode Change Windows

Mode changes may cause changes in the deadline, which in turn may lead to deadlines being missed. Mode Change Windows are used to ensure that deadlines are not missed in this case. These are small transmission windows inserted into the normal duty-cycle of the mote and are used exclusively for transmitting messages when mode changes are required. MCWs provide the following:

- They allow modes to be switched earlier than normal, sending all messages immediately after the mode change, reducing mode change latency.
- They ensure deadlines are met across all possible mode changes by scheduling the minimal number of MCWs to meet a specific deadline.
- They provide a method for scheduling multiple transmission windows which may occupy the same radio-on time.
- They show how DDC can be used to inform MCW, allowing the mode change window size to be reduced, saving power, including analysis of the overheads that MCWs impose on the system.

The last contribution meets Objective 3 by introducing Mode Change Windows. This ensures that the timing requirements for the applications are met across modes, and with coexisting applications.

These contributions validate the hypothesis presented within section 1.2, that by using a systematic analysis of the application, and that by using the application requirements with feedback, dependability can be increased and further improvements to reliability and availably can be made.

6.4 Summary

As identified in the introduction dependability consists of 5 attributes, Availability, Reliability, Safety, Integrity, and Maintainability. DA focuses mainly on monitoring the availability of the network, which informs maintainability, leading to an increase in the network reliability. DA identified that without larger run-times maintenance would become an issue with large numbers of devices. For this reason DDC was introduced to take the timing requirements identified from DA, and use these to inform the duty-cycling of the network,

increasing availability and reliability. In addition DDC takes into account the variance in the surrounding environment to ensure that even in severe environments deadlines are met, at the expense of power. Finally MCW was introduced to ensure that applications can change mode and coexist with other applications without adversely affecting the timing requirements.

The two dependability attributes which have not been addressed by these solutions are Safety and Integrity. Safety has not been considered as there are no actuators on the devices studied in this thesis, and therefore the WSN itself cannot cause any harm. Integrity has been assumed to be handled by the lower levels of the protocol stack, exposing any loss of integrity as dropped packets at the upper levels. These will be discussed in more detail within the further work in Section 6.5.

6.5 Further Work

To expand on the work presented within this thesis Dependability Assurance should be applied to more scenarios to evaluate if the process can be easily applied to other applications. This would validate that the process is suitably generic to apply to any WSN scenario, yet detailed enough to generate appropriate dependability attributes and the respective dependability tests.

To aid this process it would be ideal to model the application using a more formal specification, such as that performed by Wu et al [163]. This would then allow for more automated processing of the application, possibly allowing for the automatic generation of Derived Safety Requirements and the respective Dependability Tests. The issue with this approach is that the consequences of failure may require human intervention in order to analyse them.

The last improvement to Dependability Assurance would be an analysis into the optimal frequency to execute the run-time tests. Currently the tests run very frequently to ensure that all network changes are detected, however analysis using the failure rate of the devices for this period could be used to inform the frequency of the run-time tests. This could be extended to include the current level of network degradation, with more degraded networks requiring higher levels of monitoring.

Dynamic Duty Control can be extended to provide more recommendations on tuning the PID parameters of the feedback loops. As specific tunings would be deployment-specific then these tunings may need to be performed on-line. This may involve off-line testing

within the simulator if the behaviour is accurate enough.

More experiments could be performed with other MAC and Routing protocols. The behaviour of more statefull protocols such as 6lowpan [101], which require periodic messages to be sent around the network to maintain state information, may be adversely affected by DDC. To solve this issue the Routing messages could be modelled as another application, thus creating dedicated transmission windows for the routing messages.

As mentioned in Section 2.4.2 in-network data aggregation can be used to reduce the volume of data that arrives at the sink. As DDC can optimise the network based upon the amount of data that is received at the sink it would be interesting to measure the benefit this combined approach can provide to the lifetime of the network. This could be combined with investigations into the trade-off between the accuracy of the data, which in turn reduces the data size, and thus allowing DDC to increase the availability though deeper duty cycles.

Finally more research could be conducted into the appropriate size of the slack window (before variation handling), as it may be beneficial to size this based upon the expected dPRR/dT. This derives from the fact that a greater sudden change in PRR would require a greater amount of slack to allow the PID loops time to react, with guarantees that the PRR will never change abruptly allowing for small levels of slack.

Mode change windows could have more extended model checking to ensure that the window collision avoidance is correct under all circumstances, as currently this only covers the mode change windows themselves.

More work needs to be conducted to reduce the worst case period for the mode change windows, as currently the maximum transmission time is used. In practice this value can be reduced as the sleep time is much smaller, thus generating less packets, however the number of packets to transmission time calculation is complex as it involves MAC and Routing issues.

Outside of the protocols presented in this thesis the safety and integrity attributes of dependability need to be investigated. These would involve allowing actuators on the WSN devices, possibly requiring HAZOP analysis in conjunction with the SHARD analysis. Currently the integrity of the WSN has been assumed to be one of two simplified states, either successful or failed sending of packets, however more information in the form of Received Signal Strength Indication and Link Quality Indication values could be used to further inform Dependability Assurance and Dynamic Duty Control.

Chapters 4 and 5 have been evaluated within both low and high fidelity simulations, however detailed evaluation of these systems within a physical environment would be ideal, but would also be difficult to achieve. The difficulty in performing a similar evaluation due to that performed in the simulations is due to the high level of variance in the environment as experienced in the physical experimentation in Section 3.3.3.1. Unlike the DA experiments, the DDC evaluations require a much larger runtime to perform the evaluation, allowing large variations in the environment to occur. A simple example of such variations is the larger number of dropped packets during working hours, most likely due to the increased number of wireless devices operating in the 2.4Ghz band as more people are present within the building along with their associated laptops and mobile phones. For this reason physical experiments should be performed for DDC, however these can only validate that DDC operates correctly within a real environment, but due to the variations in the environment direct comparisons between the simulation results and the physical results cannot be drawn. Primarily this would be shown as a larger awake period, required to handle the dropped packets, which in turn would give a lower than expected energy saving.

Abbreviations and Nomenclature

ALARP	As Low As Reasonably Practicable
BLE	Bluetooth Low Energy
CA	Collision Avoidance
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
DA	Dependability Assurance
DAG	Directed Acyclic Graphs
DAO	Destination Advertisement Object
DDC	Dynamic Duty Control
DSR	Derived Safety Requirements
DT	Dependability Test
FMEA	Failure Mode and Effects Analysis
HAZOP	Hazard And Operability Study
HB	Heartbeat
HM	Health Monitoring
IMU	Inertial Measurement Unit
IP	Internet Protocol
LQI	Link Quality Indication

MAC	Media Access Control
MCW	Mode Change Window
MEMS	Microelectromechanical Systems
NN	Neural Network
NS	Network-Simulator
OS	Operating System
PHY	Physical
PRR	Packet Reception Rate
QOS	Quality Of Service
RAM	Random Access Memory
RF	Radio Frequency
ROM	Read-only Memory
RPL	Routing Protocol for Low-Power and Lossy Networks
RSSI	Received Signal Strength Indication
RTA	Run-Time Assurance
RTT	Round-Trip Time
RX	Receive
SHARD	Software Hazard Analysis and Resolution in Design
SNEDL	Sensor Network Event Description Language
TDMA	Time Division Multiple Access
TTL	Time To Live
TX	Transmit
WPAN	Wireless personal area network
WSN	Wireless Sensor Network

References

[1] Ameer Ahmed Abbasi and Mohamed Younis. A survey on clustering algorithms for wireless sensor networks. *Computer communications*, 30(14):2826–2841, 2007.

[2] N Accettura, LA Grieco, G Boggia, and P Camarda. Performance analysis of the RPL routing protocol. In *International Conference on Mechatronics*, pages 767–772. IEEE, 2011.

[3] Marcos Kawazoe Aguilera, Wei Chen, and Sam Toueg. Heartbeat: A timeout-free failure detector for quiescent reliable communication. In *Distributed Algorithms*, pages 126–140. Springer, 1997.

[4] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349, 2005.

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

[6] Cristina Albaladejo, Fulgencio Soto, Roque Torres, Pedro Sanchez, and Juan A Lopez. A low-cost sensor buoy system for monitoring shallow marine environments. *Sensors*, 12(7):9613–9634, 2012.

[7] Hande Alemdar and Cem Ersoy. Wireless sensor networks for healthcare: A survey. *Computer Networks*, 54(15):2688–2710, 2010.

[8] Asim Ali and Sebastien Tixeuil. Advanced faults patterns for WSN dependability benchmarking. In *13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*, pages 39–48. ACM, 2010.

[9] Cesare Alippi and Giovanni Vanini. A RSSI-based and calibrated centralized localization technique for wireless sensor networks. In *International Conference on Pervasive Computing and Communications Workshops*, pages 301–306, 2006.

[10] Karl Johan Astrom and Tore Hagglund. The future of PID control. *Control engineering practice*, 9(11):1163–1175, 2001.

[11] Algirdas Avizienis, J-C Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[12] Algirdas Avizienis, Jean-Claude Laprie, and Brian Randell. Dependability and its threats: a taxonomy. In *Building the Information Society*, pages 91–120. Springer, 2004.

[13] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, et al. *Fundamental concepts of dependability*. University of Newcastle upon Tyne, Computing Science, 2001.

[14] Antimo Barbato, Luca Borsani, Antonio Capone, and Stefano Melzi. Home energy saving through a user profiling system based on wireless sensors. In *First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 49–54. ACM, 2009.

[15] Paolo Baronti, Prashant Pillai, Vince WC Chook, Stefano Chessa, Alberto Gotta, and Y Fun Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Computer Communications*, 30(7):1655–1695, 2007.

[16] Iain Bate and Mark Louis Fairbairn. Searching for the minimum failures that can cause a hazard in a wireless sensor network. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, pages 1213–1220. ACM, 2013.

[17] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. *UPPAAL - A tool suite for automatic verification of real-time systems*. Springer, 1996.

[18] Carlo Alberto Boano, Thiemo Voigt, Nicolas Tsiftes, Luca Mottola, Kay Romer, and Marco Antonio Zuniga. Making sensornet MAC protocols robust against interference. In *Wireless Sensor Networks*, pages 272–288. Springer, 2010.

[19] Samir Bouabdallah, Andre Noth, and Roland Siegwart. PID vs LQ control techniques applied to an indoor micro quadrotor. In *International Conference on Intelligent Robots and Systems*, volume 3, pages 2451–2456. IEEE, 2004.

[20] Bruno Bougard, Francky Catthoor, Denis C Daly, Anantha Chandrakasan, and Wim Dehaene. Energy efficiency of the IEEE 802.15. 4 standard in dense wireless microsensor networks: Modeling and improvement perspectives. In *Design, Automation, and Test in Europe*, pages 221–234. Springer, 2008.

[21] Josh Broch, David A Maltz, David B Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97. ACM, 1998.

[22] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *4th international conference on Embedded networked sensor systems*, pages 307–320. ACM, 2006.

[23] Jinsung Byun, Boungju Jeon, Junyoung Noh, Youngil Kim, and Sehyun Park. An intelligent self-adjusting sensor for smart home services based on ZigBee communications. *IEEE Transactions on Consumer Electronics*, 58(3):794–802, 2012.

[24] Ed Callaway, Paul Gorday, Lance Hester, Jose A Gutierrez, Marco Naeve, Bob Heile, and Venkat Bahl. Home networking with IEEE 802.15.4: a developing standard for low-rate wireless personal area networks. *IEEE Communications Magazine*, 40(8):70–77, 2002.

[25] Erdal Cayirci and Tolga Coplu. SENDROM: sensor networks for disaster relief operations management. *Wireless Networks*, 13(3):409–423, 2007.

[26] Ian D Chakeres and Elizabeth M Belding-Royer. AODV routing protocol implementation design. In *24th International Conference on Distributed Computing Systems Workshops*, pages 698–703. IEEE, 2004.

[27] Ranveer Chandra, Venugopalan Ramasubramanian, and Kenneth P Birman. Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks. In *21st International Conference on Distributed Computing Systems*, pages 275–283. IEEE, 2001.

[28] Dazhi Chen and Pramod K Varshney. QoS support in wireless sensor networks: A survey. In *International Conference on Wireless Networks*, volume 233, pages 1–7, 2004.

[29] Brendan Cody-Kenny, David Guerin, Desmond Ennis, Ricardo Simon Carbajo, Meriel Huggard, and Ciaran Mc Goldrick. Performance evaluation of the 6LoWPAN protocol on MICAz and TelosB motes. In *4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 25–30. ACM, 2009.

[30] A Coronato and A Testa. Approaches of wireless sensor network dependability assessment. In *Federated Conference on Computer Science and Information Systems*, pages 881–888, Sept 2013.

[31] Brian P Crow, Indra Widjaja, Jeong Geun Kim, and Prescott T Sakai. IEEE 802.11 wireless local area networks. *Communications Magazine, IEEE*, 35(9):116–126, 1997.

[32] Gracon Huttennberg Eliatan Leite de Lima, Pedro FR Neto, et al. WSN as a tool for supporting agriculture in the precision irrigation. In *Sixth International Conference on Networking and Services*, pages 137–142. IEEE, 2010.

[33] Artem Dementyev, Steve Hodges, Stuart Taylor, and Joshua Smith. Power consumption analysis of bluetooth low energy, ZigBee and ANT sensor nodes in a cyclic sleep scenario. In *IEEE International Wireless Symposium*, pages 1–4. IEEE, 2013.

[34] Amol Deshpande, Carlos Guestrin, and Sam Madden. Model-based querying in sensor networks. In *Encyclopedia of Database Systems*, pages 1764–1768. Springer, 2009.

[35] Jordi Dunjo, Vasilis Fthenakis, Juan A Vilchez, and Josep Arnaldos. Hazard and operability (hazop) analysis. a literature review. *Journal of hazardous materials*, 173(1):19–32, 2010.

[36] Adam Dunkels. The ContikiMAC radio duty cycling protocol. `http://www.dunkels.com/adam/dunkels11contikimac.pdf`, 2011. [Online; accessed 12-June-2014].

[37] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462. IEEE, 2004.

[38] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. Protothreads: simplifying event-driven programming of memory-constrained embedded systems. In *4th*

international conference on Embedded networked sensor systems, pages 29–42. Acm, 2006.

[39] David Egan. The emergence of ZigBee in building automation and industrial controls. *Computing and Control Engineering*, 16(2):14–19, 2005.

[40] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *ACM Special Interest Group on Operating Systems Review*, 36(SI):147–163, 2002.

[41] Joakim Eriksson, Fredrik Osterlind, Niclas Finne, Nicolas Tsiftes, Adam Dunkels, Thiemo Voigt, Robert Sauter, and Pedro Jose Marron. COOJA/MSPSim: interoperability testing for wireless sensor networks. In *2nd International Conference on Simulation Tools and Techniques*, page 27. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[42] Melike Erol-Kantarci and Hussein T Mouftah. Wireless sensor networks for domestic energy management in smart grids. In *25th Biennial Symposium on Communications*, pages 63–66. IEEE, 2010.

[43] Mark Louis Fairbairn and Iain Bate. Using feedback control within WSN's to meet application requirements. In *International Conference on Distributed Computing in Sensor Systems*, pages 415–422. IEEE, 2013.

[44] Mark Louis Fairbairn, Iain Bate, and John A. Stankovic. Improving the dependability of sensornets. In *International Conference on Distributed Computing in Sensor Systems*, pages 274–282. IEEE, 2013.

[45] Muhammad Omer Farooq and Thomas Kunz. Operating systems for wireless sensor networks: A survey. *Sensors*, 11(6):5900–5930, 2011.

[46] Konstantinos P Ferentinos and Theodore A Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. *Computer Networks*, 51(4):1031–1051, 2007.

[47] Zwe-Lee Gaing. A particle swarm optimization approach for optimum design of PID controller in avr system. *IEEE Transactions on Energy Conversion*, 19(2):384–391, 2004.

[48] Saurabh Ganeriwal, Ilias Tsigkogiannis, Hohyun Shim, Vlassios Tsiatsis, Mani B Srivastava, and Deepak Ganesan. Estimating clock uncertainty for efficient duty-cycling in sensor networks. *IEEE/ACM Transactions on Networking*, 17(3):843–856, 2009.

[49] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.

[50] Alexandre Guitton, Niki Trigoni, and Sven Helmer. Fault-tolerant compression algorithms for delay-sensitive sensor networks with unreliable links. In SotirisE. Nikoletseas, BogdanS. Chlebus, DavidB. Johnson, and Bhaskar Krishnamachari, editors, *Distributed Computing in Sensor Systems*, volume 5067 of *Lecture Notes in Computer Science*, pages 190–203. Springer Berlin Heidelberg, 2008.

[51] Salem Hadim and Nader Mohamed. Middleware: Middleware challenges and approaches for wireless sensor networks. *IEEE Distributed Systems Online*, 7(3):1, 2006.

[52] Jahan Hassan and Sanjay Jha. Optimising expanding ring search for multi-hop wireless networks. In *Global Telecommunications Conference*, volume 2, pages 1061–1065. IEEE, 2004.

[53] Mark Hempstead, Michael J Lyons, David Brooks, and Gu-Yeon Wei. Survey of hardware systems for wireless sensor networks. *Journal of Low Power Electronics*, 4(1):11–20, 2008.

[54] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. In *ACM Special Interest Group on Operating Systems Review*, volume 34, pages 93–104. ACM, 2000.

[55] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[56] NA Hoult, PRA Fidler, PG Hill, and CR Middleton. Long-term wireless structural health monitoring of the ferriby road bridge. *Journal of Bridge Engineering*, 15(2):153–159, 2010.

[57] NA Hoult, PRA Fidler, IJ Wassell, PG Hill, and CR Middleton. Wireless structural health monitoring at the humber bridge UK. volume 161, pages 189–195. Thomas Telford, 2008.

[58] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *6th annual international conference on Mobile computing and networking*, pages 56–67. ACM, 2000.

[59] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, 2003.

[60] Milos Jevtic, Nikola Zogovic, and Goran Dimic. Evaluation of wireless sensor network simulators. In *17th Telecommunications Forum*, pages 1303–1306, 2009.

[61] Xiaofan Jiang, Joseph Polastre, and David Culler. Perpetual environmentally powered sensor networks. In *Fourth International Symposium on Information Processing in Sensor Networks*, pages 463–468. IEEE, 2005.

[62] Chen Jie, Chen Jiapin, and Li Zhenbo. Energy-efficient AODV for low mobility ad hoc networks. In *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1512–1515, 2007.

[63] Binjia Jiao Sang H Son John and A Stankovic. GEM: Generic event service middleware for wireless sensor networks. `https://cheetah.cs.virginia.edu/~stankovic/psfiles/binjia-inss.pdf`, 2005. [Online; accessed 5-July-2014].

[64] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Springer, 1996.

[65] A Kadri, E. Yaacoub, M. Mushtaha, and A Abu-Dayya. Wireless sensor network for real-time air pollution monitoring. In *1st International Conference on Communications, Signal Processing, and their Applications*, pages 1–5, Feb 2013.

[66] Konstantinos Kalpakis, Koustuv Dasgupta, and Parag Namjoshi. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks*, 42(6):697–716, 2003.

[67] K Khan, Rafi U Zaman, KA Reddy, and TS Harsha. An efficient DSDV routing protocol for wireless mobile ad hoc networks and its performance comparison. In *Second UKSIM European Symposium on Computer Modeling and Simulation*, pages 506–511. IEEE, 2008.

[68] Kavi Khedo, Rubeena Doomun, Sonum Aucharuz, et al. Reada: Redundancy elimination for accurate data aggregation in wireless sensor networks. *Wireless Sensor Network*, 2(04):300, 2010.

[69] JeongGil Ko, Kevin Klues, Christian Richter, Wanja Hofer, Branislav Kusy, Michael Bruenig, Thomas Schmid, Qiang Wang, Prabal Dutta, and Andreas Terzis. Low power or high performance? a tradeoff whose time has come (and nearly gone). In *Wireless Sensor Networks*, pages 98–114. Springer, 2012.

[70] JeongGil Ko, Nicolas Tsiftes, Adam Dunkels, and Andreas Terzis. Pragmatic low-power interoperability: ContikiMAC vs TinyOS LPL. In *9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications Networks*, pages 94–96. IEEE, 2012.

[71] Praveen Kumar, Lohith Reddy, and Shirshu Varma. Distance measurement and error estimation scheme for RSSI based localization in wireless sensor networks. In *Fifth IEEE Conference on Wireless Communication and Sensor Networks (WCSN)*, pages 1–4. IEEE, 2009.

[72] Branislav Kusy, Prabal Dutta, Philip Levis, Miklos Maroti, Akos Ledeczi, and David Culler. Elapsed time on arrival: a simple and versatile primitive for canonical time synchronisation services. *International Journal of Ad Hoc and Ubiquitous Computing*, 1(4):239–251, 2006.

[73] Kanishka Lahiri, Sujit Dey, Debashis Panigrahi, and Anand Raghunathan. Battery-driven system design: A new frontier in low power design. In *2002 Asia and South Pacific Design Automation Conference*, page 261. IEEE Computer Society, 2002.

[74] Chih-Chung Lai, Chuan-Kang Ting, and Ren-Song Ko. An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance applications. In *IEEE Congress on Evolutionary Computation*, pages 3531–3538. IEEE, 2007.

[75] Benoit Latre, Pieter De Mil, Ingrid Moerman, Bart Dhoedt, Piet Demeester, and Niek Van Dierdonck. Throughput and delay analysis of unslotted IEEE 802.15. 4. *Journal of Networks*, 1(1):20–28, 2006.

[76] Sangwon Lee, Dukhee Yoon, and Amitabha Ghosh. Intelligent parking lot application using wireless sensor networks. In *Collaborative Technologies and Systems*, pages 48–57. IEEE, 2008.

[77] Sung-Ju Lee and Mario Gerla. AODV-BR: Backup routing in ad hoc networks. In *Wireless Communications and Networking Conference*, volume 3, pages 1311–1316. IEEE, 2000.

[78] Tomas Lennvall, Stefan Svensson, and Fredrik Hekland. A comparison of WirelessHART and ZigBee for industrial applications. In *IEEE International Workshop on Factory Communication Systems*, volume 2008, pages 85–88, 2008.

[79] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: Accurate and scalable simulation of entire tinyos applications. In *1st international conference on Embedded networked sensor systems*, pages 126–137. ACM, 2003.

[80] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, et al. TinyOS: An operating system for sensor networks. In *Ambient intelligence*, pages 115–148. Springer, 2005.

[81] Philip Levis, Samuel Madden, David Gay, Joseph Polastre, Robert Szewczyk, Alec Woo, Eric A Brewer, and David E Culler. The emergence of networking abstractions and techniques in TinyOS. In *Networked Systems Design and Implementation*, volume 4, pages 1–14, 2004.

[82] Mo Li and Yunhao Liu. Underground coal mine monitoring with wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(2):10, 2009.

[83] Wen-Hwa Liao, Yucheng Kao, and Chien-Ming Fan. Data aggregation in wireless sensor networks using ant colony algorithm. *Journal of Network and Computer Applications*, 31(4):387–401, 2008.

[84] Hyojun Lim and Chongkwon Kim. Multicast tree construction and flooding in wireless ad hoc networks. In *3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 61–68. ACM, 2000.

[85] Jaime Lloret, Miguel Garcia, Diana Bri, and Sandra Sendra. A wireless sensor network deployment for rural and forest fire detection and verification. *Sensors*, 9(11):8722–8747, 2009.

[86] JA Lopez Riquelme, F Soto, J Suardiaz, P Sanchez, A Iborra, and JA Vera. Wireless sensor networks for precision horticulture in southern spain. *Computers and Electronics in Agriculture*, 68(1):25–35, 2009.

[87] Konrad Lorincz, Bor-rong Chen, Geoffrey Werner Challen, Atanu Roy Chowdhury, Shyamal Patel, Paolo Bonato, and Matt Welsh. Mercury: A wearable sensor network platform for high-fidelity motion analysis. In *7th ACM Conference on Embedded Networked Sensor Systems*, pages 183–196, 2009.

[88] Robyn R Lutz. Software engineering for safety: a roadmap. In *Conference on The Future of Software Engineering*, pages 213–226. ACM, 2000.

[89] Jerome P Lynch and Kenneth J Loh. A summary review of wireless sensors and sensor networks for structural health monitoring. *Shock and Vibration Digest*, 38(2):91–130, 2006.

[90] Samuel Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[91] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 30(1):122–173, 2005.

[92] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97. ACM, 2002.

[93] David Malone, Peter Clifford, and Douglas J Leith. On buffer sizing for voice in 802.11 WLANs. *IEEE Communications Letters*, 10(10):701–703, 2006.

[94] Francesco Marcelloni and Massimo Vecchio. Enabling energy-efficient and lossy-aware data compression in wireless sensor networks by multi-objective evolutionary optimization. *Information Sciences*, 180(10):1924–1941, 2010.

[95] Miklos Maroti, Branislav Kusy, Gyula Simon, and Akos Ledeczi. The flooding time synchronization protocol. In *2nd international conference on Embedded networked sensor systems*, pages 39–49. ACM, 2004.

[96] John A McDermid, Mark Nicholson, David J Pumfrey, and P Fenelon. Experience with the application of HAZOP to computer-based systems. In *Tenth Annual Conference on Computer Assurance, Systems Integrity, Software Safety and Process Security*, pages 37–48. IEEE, 1995.

[97] Robert E Melchers. On the alarp approach to risk management. *Reliability Engineering & System Safety*, 71(2):201–208, 2001.

[98] Rajiv Misra and CR Mandal. Performance comparison of AODV/DSR on-demand routing protocols for ad hoc networks in constrained situation. In *IEEE International Conference on Personal Wireless Communications*, pages 86–89. IEEE, 2005.

[99] Yi-Jen Mon, Chih-Min Lin, and Imre J Rudas. Wireless sensor network (WSN) control for indoor temperature monitoring. *Acta Polytechnica Hungarica*, 9(6):17–28, 2012.

[100] David Moss, Jonathan Hui, and Kevin Klues. Low power listening. *TinyOS Core Working Group, TEP*, 105, 2007.

[101] Geoff Mulligan. The 6LoWPAN architecture. In *4th workshop on Embedded networked sensors*, pages 78–82. ACM, 2007.

[102] Razvan Musaloiu-E and Andreas Terzis. Minimising the effect of WiFi interference in 802.15.4 wireless sensor networks. *International Journal of Sensor Networks*, 3(1):43–54, 2008.

[103] Lama Nachman, Jonathan Huang, Junaith Shahabdeen, Robert Adler, and Ralph Kling. Imote2: Serious computation at the edge. In *International Wireless Communications and Mobile Computing Conference*, pages 1118–1123. IEEE, 2008.

[104] Silvia Nittel. A survey of geosensor networks: Advances in dynamic environmental monitoring. *Sensors*, 9(7):5664–5678, 2009.

[105] Ertan Onur, Cem Ersoy, Hakan Deliç, and Lale Akarun. Surveillance wireless sensor networks: deployment quality analysis. *IEEE Network*, 21(6):48–53, 2007.

[106] Fredrik Osterlind and Adam Dunkels. Approaching the maximum 802.15. 4 multi-hop throughput. In *The Fifth Workshop on Embedded Networked Sensors*, pages 6–12, 2008.

[107] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. In *31st IEEE Conference on Local Computer Networks*, pages 641–648. IEEE, 2006.

[108] Paritosh Padhy, Rajdeep K Dash, Kirk Martinez, and Nicholas R Jennings. A utility-based sensing and communication model for a glacial sensor network. In *fifth international joint conference on Autonomous agents and multiagent systems*, pages 1353–1360. ACM, 2006.

[109] Meng-Shiuan Pan, Lun-Wu Yeh, Yen-Ann Chen, Yu-Hsuan Lin, and Yu-Chee Tseng. A WSN-based intelligent light control system considering user activities and profiles. *IEEE Sensors Journal*, 8(10):1710–1721, 2008.

[110] Michael Pan, Sheng-Yan Chuang, and Sheng-De Wang. Local repair mechanisms for on-demand routing in mobile ad hoc networks. In *11th Pacific Rim International Symposium on Dependable Computing*, pages 8–16. IEEE, 2005.

[111] Nikolaos A Pantazis and Dimitrios D Vergados. A survey on power control issues in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 9(4):86–107.

[112] Lilia Paradis and Qi Han. A survey of fault management in wireless sensor networks. *Journal of Network and Systems Management*, 15(2):171–190, 2007.

[113] Youbin Peng, Damir Vrancic, and Raymond Hanus. Anti-windup, bumpless, and conditioned transfer techniques for PID controllers. *Control Systems, IEEE*, 16(4):48–57, 1996.

[114] Haapanen Pentti and Helminen Atte. Failure mode and effects analysis of software-based automation systems. *VTT Industrial Systems, STUK-YTO-TR*, 190:190, 2002.

[115] Charles E Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *SIGCOMM Computer Communication Review*, volume 24, pages 234–244. ACM, 1994.

[116] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *2nd international conference on Embedded networked sensor systems*, pages 95–107. ACM, 2004.

[117] Joseph Polastre, Jonathan Hui, Philip Levis, Jerry Zhao, David Culler, Scott Shenker, and Ion Stoica. A unifying link abstraction for wireless sensor networks. In *3rd international conference on Embedded networked sensor systems*, pages 76–89. ACM, 2005.

[118] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *Fourth International Symposium on Information Processing in Sensor Networks*, pages 364–369. IEEE, 2005.

[119] Joseph Polastre, Robert Szewczyk, Alan Mainwaring, David Culler, and John Anderson. Analysis of wireless sensor networks for habitat monitoring. In *Wireless sensor networks*, pages 399–423. Springer, 2004.

[120] Brian Porter and AH Jones. Genetic tuning of digital PID controllers. *Electronics Letters*, 28(9):843–844, 1992.

[121] David John Pumfrey. The principled design of computer system safety analyses. `http://www.cs.york.ac.uk/~djp/publications/Thesis16.pdf`, 1999. [Online; accessed 9-Jan-2014].

[122] Kay R0mer. Time synchronization in ad hoc networks. In *2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 173–182. ACM, 2001.

[123] Vijay Raghunathan, Curt Schurgers, Sung Park, and Mani B Srivastava. Energy-aware wireless microsensor networks. *Signal Processing Magazine*, 19(2):40–50, 2002.

[124] Mohammad Shaifur Rahman, Youngil Park, and Ki-Doo Kim. Localization of wireless sensor network using artificial neural network. In *9th International Symposium on Communications and Information Technology*, pages 639–642. IEEE, 2009.

[125] Gyan Ranjan and Amit Kumar. A natural disasters management system based on location aware distributed sensor networks. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, pages 182–185. IEEE, 2005.

[126] Luis Ruiz-Garcia, P Barreiro, and JI Robla. Performance of ZigBee-based wireless sensor nodes for real-time monitoring of fruit logistics. *Journal of Food Engineering*, 87(3):405–415, 2008.

[127] Luis Ruiz-Garcia, P Barreiro, Jose Rodríguez-Bermejo, and JI Robla. Review. monitoring the intermodal, refrigerated transport of fruit using sensor networks. *Spanish Journal of Agricultural Research*, 5(2):142–156, 2007.

[128] Luis Ruiz-Garcia, Loredana Lunadei, Pilar Barreiro, and Ignacio Robla. A review of wireless sensor technologies and applications in agriculture and food industry: state of the art and current trends. *Sensors*, 9(6):4728–4750, 2009.

[129] Francoise Sailhan, Thierry Delot, Animesh Pathak, Aymeric Puech, and Matthieu Roy. Fault injection and monitoring for dependability analysis of wireless sensor-actuators networks.

[130] Francoise Sailhan, Thierry Delot, Animesh Pathak, Aymeric Puech, and Matthieu Roy. Dependable wireless sensor networks. `http://cedric.cnam.fr/~sailhanf/publications/gedsip.pdf`, 2009. [Online; accessed 2-May-2014].

[131] Kunal Shah and Mohan Kumar. Distributed independent reinforcement learning (DIRL) approach to resource management in wireless sensor networks. In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems*, pages 1–9. IEEE, 2007.

[132] Kuei-Ping Shih, Sheng-Shih Wang, Hung-Chang Chen, and Pao-Hwa Yang. CollECT: Collaborative event detection and tracking in wireless heterogeneous sensor networks. *Computer Communications*, 31(14):3124–3136, 2008.

[133] Himanshu Singh and Bhaskar Biswas. Comparison of CSMA based MAC protocols of wireless sensor networks. *International Journal on AdHoc Networking Systems*, 2(2):11–20, 2012.

[134] Shio Kumar Singh, MP Singh, DK Singh, et al. Routing protocols in wireless sensor networks: A survey. *International Journal of Computer Science & Engineering Survey*, 1:63–83, 2010.

[135] Vipul Singhvi, Andreas Krause, Carlos Guestrin, James H Garrett Jr, and H Scott Matthews. Intelligent light control using sensor networks. In *3rd international conference on Embedded networked sensor systems*, pages 218–229. ACM, 2005.

[136] Adonis Skordylis, Niki Trigoni, and Alexandre Guitton. A study of approximate data management techniques for sensor networks. In *International Workshop on Intelligent Solutions in Embedded Systems*, pages 1–12. IEEE, 2006.

[137] Sasha Slijepcevic and Miodrag Potkonjak. Power efficient organization of wireless sensor networks. In *IEEE International Conference on Communications*, volume 2, pages 472–476. IEEE, 2001.

[138] Philipp Sommer and Roger Wattenhofer. Gradient clock synchronization in wireless sensor networks. In *2009 International Conference on Information Processing in Sensor Networks*, pages 37–48. IEEE Computer Society, 2009.

[139] Byungrak Son, Yong-sork Her, and J Kim. A design and implementation of forest-fires surveillance system based on wireless sensor networks for south korea mountains. *International Journal of Computer Science and Network Security*, 6(9):124–130, 2006.

[140] Jianping Song, Song Han, Aloysius K Mok, Deji Chen, Mike Lucas, and Mark Nixon. WirelessHART: Applying wireless technology in real-time industrial process control. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 377–386. IEEE, 2008.

[141] SV Srikanth, PJ Pramod, KP Dileep, S Tapas, Mahesh U Patil, and Chandra Babu N Sarat. Design and implementation of a prototype smart parking (SPARK) system using wireless sensor networks. In *Advanced Information Networking and Applications Workshops*, pages 401–406. IEEE, 2009.

[142] Alexandru Stan. Porting the core of the Contiki operating system to the TelosB and MicaZ platforms. `http://www.eecs.iu-bremen.de/archive/bsc-2007/stan.pdf`, 2007. [Online; accessed 24-March-2014].

[143] Bharath Sundararaman, Ugo Buy, and Ajay D Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281–323, 2005.

[144] Dipak Surie, Olivier Laguionie, and Thomas Pederson. Wireless sensor networking of everyday objects in a smart home environment. In *International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 189–194. IEEE, 2008.

[145] Tomasz Surmacz, Mariusz Slabicki, Bartosz Wojciechowski, and Maciej Nikodem. Lessons learned from the deployment of wireless sensor networks. In Andrzej Kwiecien, Piotr Gaj, and Piotr Stera, editors, *Computer Networks*, volume 370 of *Communications in Computer and Information Science*, pages 76–85. Springer Berlin Heidelberg, 2013.

[146] Wen Tan, Jizhen Liu, Tongwen Chen, and Horacio J Marquez. Comparison of some well-known PID tuning formulas. *Computers & chemical engineering*, 30(9):1416–1423, 2006.

[147] Ech-Chaitami Tariq, Radouane Mrabet, and Hassan Berbia. Interoperability of low-pans based on the IEEE 802.15.4 standard through IPv6. *International Journal of Computer Science Issues*, page 315, 2011.

[148] Jonathan Tate and Iain Bate. Maintaining stable node populations in long-lifetime sensornets. In *15th IEEE International Conference on Engineering of Complex Computer Systems*, pages 159–168. IEEE, 2010.

[149] Fan Tiegang, Teng Guifa, and Huo Limin. Deployment strategy of WSN based on minimizing cost per unit area. *Computer Communications*, 38:26–35, 2014.

[150] Nicolas Tsiftes, Joakim Eriksson, and Adam Dunkels. Low-power wireless IPv6 routing with ContikiRPL. In *9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 406–407. ACM, 2010.

[151] Pere Tuset-Peiro. OpenMote CC2538. `http://www.openmote.com/openmote-cc2538/`, 2014. [Online; accessed 30-June-2014].

[152] Asma Tuteja, Rajneesh Gujral, and Sunil Thalia. Comparative performance analysis of DSDV, AODV and DSR routing protocols in MANET using NS-2. In *2010 International Conference on Advances in Computer Engineering (ACE)*, pages 330–333. IEEE, 2010.

[153] Md Borhan Uddin and Claude Castelluccia. Toward clock skew based wireless sensor node services. In *The 5th Annual Wireless Internet Conference*, pages 1–9. IEEE, 2010.

[154] Tijs Van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *1st international conference on Embedded networked sensor systems*, pages 171–180. ACM, 2003.

[155] G Virone, A Wood, L Selavo, Q Cao, L Fang, T Doan, Z He, R Stoleru, S Lin, and JA Stankovic. An assisted living oriented information system based on a residential wireless sensor network. In *1st Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare*, pages 95–100. IEEE, 2006.

[156] Thiemo Voigt, Joakim Eriksson, Fredrik Osterlind, Robert Sauter, Nils Aschenbruck, Pedro J Marron, Vinny Reynolds, Lei Shu, Otto Visser, Anis Koubaa, et al. Towards comparable simulations of cooperating objects and wireless sensor networks. In *Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, page 77. ICST, 2009.

[157] Liping Wang and Viktoria Fodor. Cooperative geographic routing in wireless mesh networks. In *7th International Conference onMobile Adhoc and Sensor Systems*, pages 570–575. IEEE, 2010.

[158] Ning Wang, Naiqian Zhang, and Maohua Wang. Wireless sensors in agriculture and food industryâĂŤrecent development and future perspective. *Computers and electronics in agriculture*, 50(1):1–14, 2006.

[159] Xiaodong Wang, Xiaorui Wang, Liu Liu, and Guoliang Xing. DutyCon: A dynamic duty-cycle control approach to end-to-end delay guarantees in wireless sensor networks. *Transactions on Sensor Networks*, 9(4):42, 2013.

[160] Elias Weingartner, Hendrik Vom Lehn, and Klaus Wehrle. A performance comparison of recent network simulators. In *IEEE International Conference on Communications*, pages 1–5. IEEE, 2009.

[161] Bernard Widrow and Michael A Lehr. 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. *IEEE*, 78(9):1415–1442, 1990.

[162] Anthony Wood, John A Stankovic, Gilles Virone, Leo Selavo, Zhimin He, Qiuhua Cao, Thao Doan, Yafeng Wu, Lei Fang, and Radu Stoleru. Context-aware wireless sensor networks for assisted living and residential monitoring. *IEEE Network*, 22(4):26–33, 2008.

[163] Yafeng Wu, Krasimira Kapitanova, Jingyuan Li, John A Stankovic, Sang H Son, and Kamin Whitehouse. Run time assurance of application-level requirements in wireless sensor networks. In *9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 197–208. ACM, 2010.

[164] W. Xie, M. Goyal, H. Hosseini, J. Martocci, Y. Bashir, E. Baccelli, and A Durresi. A performance analysis of point-to-point routing along a directed acyclic graph in low power and lossy networks. In *13th International Conference on Network-Based Information Systems*, pages 111–116, Sept 2010.

[165] Ning Xiong and Per Svensson. Multi-sensor management for information fusion: issues and approaches. *Information fusion*, 3(2):163–186, 2002.

[166] Yunjiao Xue, Ho Sung Lee, Ming Yang, Priyantha Kumarawadu, Hamada H Ghenniwa, and Weiming Shen. Performance evaluation of NS-2 simulator for wireless sensor networks. In *Canadian Conference on Electrical and Computer Engineering*, pages 1372–1375. IEEE, 2007.

[167] Zhe Yang, Lin Cai, Yu Liu, and Jianping Pan. Environment-aware clock skew estimation and synchronization for wireless sensor networks. In *International Conference on Computer Communications*, pages 1017–1025. IEEE, 2012.

[168] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *Transactions on Networking*, 12(3):493–506, 2004.

[169] Thomas Zahn and Jochen Schiller. Designing structured peer-to-peer overlays as a platform for distributed network applications in mobile ad hoc networks. *Computer communications*, 31(3):643–654, 2008.

[170] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *Twelfth Workshop on Parallel and Distributed Simulation*, pages 154–161. IEEE, 1998.

[171] ZW Zeng, ZG Chen, and AF Liu. Energy-hole avoidance for WSN based on adjust transmission power. *Chinese Journal of Computers*, 33(1):12–22, 2010.

[172] Jianliang Zheng and Myung J Lee. A comprehensive performance study of IEEE 802.15.4. *Sensor Network Operations*, (4):218–237, 2006.

[173] Hubert Zimmermann. OSI reference model - the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1980.

[174] Eustathia Ziouva and Theodore Antonakopoulos. CSMA/CA performance under high traffic conditions: throughput and delay analysis. *Computer communications*, 25(3):313–321, 2002.