ENERGY-EFFICIENT ALGORITHMS FOR AD HOC WIRELESS SENSOR NETWORKS

Kok-Poh Ng

A thesis submitted for the degree of
Doctor of Philosophy

School of Electrical and Electronic Engineering

Faculty of Science, Agriculture and Engineering

November 2017

Abstract

The interest in ad hoc Wireless Sensor Networks (WSN) has been growing rapidly in the past few years due to its wide range of applications in Environment Monitoring and Forecasting, Health and Medical Care, Underwater Communications, Smart Energy, and Building and Home Automation industries. The performance of different network protocols, as well as their sensitivity and the effect of different network parameters, needs to be studied and evaluated for the implementation of WSN with the right protocols and optimal parameters.

With the increasing deployment of unmanned and energy-constrained sensor devices in large-scale wireless sensor networks, energy efficiency and network lifetime have become key considerations in designing WSN routing protocols. In this work, we propose a fully distributed, multi-path load-balancing routing protocol based on Dynamic Source Routing (DSR) to improve network lifetime performance. The new protocol is simulated in ns-2 and compared with the commonly used Destination-Sequenced Distance Vector (DSDV), Ad hoc On-Demand Distance Vector (AODV) and DSR protocols. The simulation results show that the new routing protocol improves network lifetime significantly without sacrificing packet delivery performance.

Another major source of energy wastage is the idle listening of sensor nodes in the MAC layer. Different variants of synchronous duty-cycle MAC protocols have been designed for WSNs to reduce MAC layer energy consumption. However, the synchronisation process of theses protocols remains a significant contributor to the energy consumption. Energy consumption models of duty-cycle MAC protocols in single-hop neighbourhoods are first developed and analysed. A new synchronisation algorithm, 1-Sync, is proposed to address the high energy consumption problem of the existing fixed periodic synchronisation (F-Sync) algorithm, and the Intelligent Network Synchronisation (INS) algorithm. The analysis and simulation results have shown that the proposed 1-Sync algorithm yields better energy performance than the F-Sync and INS algorithms in both low and high density neighbourhoods.

In large multi-neighbourhood networks, the above synchronisation algorithms are inadequate in handling high density, high drift, and low duty-cycle operations. An adaptive energy-efficient synchronisation algorithm referred to as C-Sync, is proposed. C-Sync reduces energy consumption by adaptively regulating the synchronisation traffic and the wakeup period based on the changing network neighbourhood conditions through counter-based and exponential-smoothing algorithms. Extensive simulations of multi-hop multi-neighbourhood network scenarios are performed using ns-2; the simulation results have shown that C-Sync outperforms F-Sync and 1-Sync in energy efficiency, packet delivery ratio, and end-to-end packet delay over a wide range of node densities, drift rates and duty cycles.

Acknowledgement

I am deeply grateful to many people for their help and support, both directly and indirectly, during the period of my Ph.D. programme.

First and foremost, I would like to express my deepest gratitude to my supervisors, Dr. Charalampos Tsimenidis and Dr. Wai Lok Woo for their support, guidance, and many valuable suggestions that have enriched my research experience and made my Ph.D. journey enjoyable and productive.

I would like to thank Nanyang Polytechnic for providing me the opportunity and funding support to pursue my Ph.D. study. I would also like to thank the management and colleagues of Nanyang Polytechnic for making arrangements to cover my work during the periods I was away from the office.

I am also deeply appreciative of the emotional support and encouragement from my parents and my siblings throughout the period of the study.

Last but not least, I would like to thank my wife Yiam Hoon for her love, support, encouragement, and understanding. I would also like to thank our daughter WanQi for spending her precious vacation time to help me edit a large part of this thesis.

My apologies if I have inadvertently omitted anyone to whom acknowledgment is due.

Table of Contents

List of Figures

List of Tables

List of Symbols

c_{P_i}	cost of link *i* along path *P*
e_{P_j}	residual energy of intermediate nodes *j* along path *P*
C_P	path-cost metric of path *P*
E_P	minimum-node-energy metric of path *P*
M_P	EBDSR routing metric
L_P	number of links along path *P*
E_{SW}	total energy consumed in SYNC windows
E_{DW}	total energy consumed in DATA windows
E_{SLP}	total energy consumed in SLEEP periods
P_{tx}	power consumption in transmission state
P_{rx}	power consumption in reception state
P_{idle}	power consumption in idle state
P_{slp}	power consumption in sleep state
N	node density within 1-hop neighbourhood
N_{SP}	number of frames in 1 synchronisation period
N'_{SP}	number of frames in 1 effective synchronisation period (for a high density neighbourhood)
D	duty cycle
n_{sync}	length of *sync* packet in timeslots (TS)
n_{SW}	length of SYNC window in timeslots (TS)

n_{DW}	length of DATA window in timeslots (TS)
n_{SLP}	length of SLEEP period in timeslots (TS)
t_{TS}	duration of a timeslot (s)
T_{SP}	duration of synchronisation period (s)
T'_{SP}	duration of effective synchronisation period (s)
T	total network operation time in steady state (s)
n_{CW}	contention window for *sync* in timeslots (TS)
p_c	probability of *sync* collision
p_s	probability of successful *sync* transmission
$C_i(t)$	clock function of node *i*
θ	clock offset
f_i	clock drift rate of node *i*
c_{sn}	received *sync* packet counter
C_{thres}	threshold value of c_{sn}
N_{RP}	desirable sync inter-arrival time
w_{wk}	number of SYNC windows to sleep
w_a	waiting period to receive a valid *sync*
α	smoothing factor

Abbreviations / Acronyms

ACK	Acknowledgement Frame
ANEC	Average Node Energy Consumption
AODV	Ad hoc On-Demand Distance Vector Routing
AS-MAC	Adaptive Scheduling MAC
AP	Access Point
AWPST	Average Waiting Period for *Sync* Packet Transmission
AvDelay	Average End-to-end Packet Delay
B-MAC	Berkeley MAC
CAP	Contention Access Period
CBR	Constant Bit-Rate
CFP	Contention-Free Period
CMMBCR	Conditional Max-Min Battery Capacity Routing
CoAP	Constrained Application Protocol
CSMA	Carrier Sense Multiple Access
CSS	Chirp Spread Spectrum
C-Sync	Counter-Based Synchronisation Algorithm
CTS	Clear-to-Send Frame
dc	Duty Cycle
DCF	Distributed Coordination Function
DDR	Distributed Dynamic Routing Protocol

DSDV	Destination-sequenced Distance Vector Routing Protocol
DSR	Dynamic Source Routing Protocol
DW-MAC	Demand Wakeup MAC
E-DSR	Energy-efficient DSR
EB-DSR	Energy-balanced DSR
EDDSR	Energy Dependent DSR
EER	End-to-End Retransmissions
FDSIT	Fraction of Desired *Sync* Inter-arrival Time
FFD	Full-Function Device
FSR	Fisheye State Routing Protocol
F-Sync	Fixed, Periodic Synchronisation Packet Broadcast Algorithm
FTSP	Flooding Time Synchronisation Protocol
GPS	Global Positioning System
GSA	Global Schedule Algorithm
GTS	Guaranteed Time Slots
HARP	Hybrid Ad Hoc Routing Protocol
HHR	Hop-by-Hop Retransmissions
IARP	Intra Zone Routing Protocol
IERP	Inter-Zone Routing protocol
INS	Intelligent Network Synchronisation
IoT	Internet-of-Things
LBAR	Load-Balanced Ad hoc Routing
LEACH	Low-Energy Adaptive Clustering Hierarchy
LPL	Low Power Listening

LPWAN	Low Power Wide Area Network
LTS	Lightweight Time Synchronisation
MAC	Media Access Control
MANET	Mobile Ad Hoc Network
MER	Minimum Energy Routing Protocol
MLE	Maximum Likelihood Estimator
MPR	Multipoint Relay
MQTT	Message Queuing Telemetry Transport
MSR	Multipath Routing Protocol
NCE-DSR	Number of times nodes send Constraint Energy DSR
NRO	Normalised Routing Overhead
NTP	Network Time Protocol
OLSR	Optimized Link State Routing Protocol
PDR	Packet Delivery Ratio
PDU	Protocol Data Unit
ppm	Parts Per Million
QoS	Quality of Service
RAT	Resilient Active Time
RBS	Reference Broadcast Synchronisation
RERR	Route Error Packet
RFD	Reduced-Function Device
RI-MAC	Receiver-Initiated MAC
RREP	Route Reply Packet
RREQ	Route Request Packet

RTS	Request-to-Send Frame
RTT	Round Trip Time
SCH	Scheduling Frame
SEA-MAC	Self-Adaptive Duty Cycle MAC
S-MAC	Sensor MAC
SPIN	Sensor Protocol for Information via Negotiation
TC	Topology Control
TS	Timeslot
TDMA	Time Division Multiple Access
TDPR	Traffic Load and Lifetime Deviation Based Power-Aware Routing Protocol
TPSN	Timing-sync Protocol for Sensor Networks
UNB	Ultra-narrowband
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
ZHLS	Zone-based Hierarchical Link State
ZRP	Zone Routing Protocol

List of Publications

- K.-P. Ng and C. Tsimenidis, "Energy-balanced dynamic source routing protocol for wireless sensor network", in IEEE Conference on Wireless Sensors (ICWISE), 2013, pp. 36–41.
- K.-P. Ng, C. C. Tsimenidis, and W. L. Woo, "Energy-efficient synchronization algorithm for duty-cycle MAC protocols", in IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob), 2015, pp. 255–260.
- K.-P. Ng, C. C. Tsimenidis, and W. L. Woo, "C-Sync: Counter-based Synchronization for Duty-cycled Wireless Sensor Networks", *Ad Hoc Networks,* 2017, vol.61, pp. 51–64

Chapter 1

Introduction

1.1 Wireless Sensor Networks

Traditional wireless networking requires a fixed infrastructure for wireless end-nodes that run user applications to communicate with one another. The wireless end-nodes can only send data to their designated Access Points (APs), which are part of the fixed network infrastructure. The APs, on behalf of the wireless end-nodes, then forward the data to routers that in turn route the data to different parts of the network.

In recent years, the emergence of small, low cost, low power, multi-functional sensor nodes with wireless communication capabilities has accelerated the research and development of Wireless Sensor Networks (WSNs) [1]. Wireless sensors are small, with limited memory and computing resources, and they are inexpensive compared to traditional sensors. These sensor nodes can sense, measure, and gather information from the environment and transmit the sensed data to the user to meet the application needs.

There are two types of WSNs: structured and unstructured [2]. An unstructured WSN contains a dense deployment of sensor nodes. Sensor nodes may be deployed in an ad hoc manner into the field, and is left unattended to perform monitoring and reporting functions. In an unstructured WSN, network maintenance such as managing connectivity and detecting failures is difficult since there are many unattended nodes. On the other hand, all or some of the sensor nodes in a structured WSN are deployed with pre-determined fixed locations. The advantage of a structured network is that fewer nodes can be deployed with lower network maintenance and management cost. Fewer ad hoc nodes can now be deployed to provide coverage for the uncovered regions.

Wireless Sensor Network has the potential for many applications [3][4]: e.g. for military purpose, it can be used for monitoring, tracking and surveillance of military targets; in industry for factory instrumentation; in large cities to monitor traffic density and road conditions; in construction to monitor buildings structures; in environment to monitor forest, oceans, volcanoes, etc. The sensor nodes, which are often deployed in large numbers, and are typically deployed in difficult-to-access locations, sensed data are transferred to a base station via wireless communication. Battery is the main power source in a sensor node. Secondary power supply that harvests power from the environment may be added to the sensor node if the deployment environment is appropriate. Energy harvesting involves nodes replenishing its energy from an energy source such as solar cells, vibration, RF, acoustic noise, etc. However, power supply sources often exhibit a non-continuous behaviour so that an energy buffer (a battery) is needed as well. In any case, energy is a very critical resource and therefore, energy conservation is a key issue in the design of systems based on wireless sensor networks.

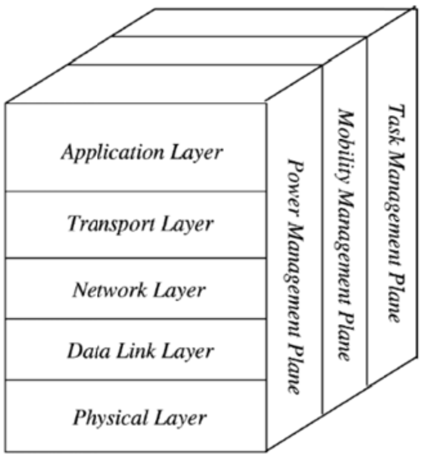
Different applications used different architecture models to achieve their intended objectives. Home control, industrial and building automation applications typically use single hop network architecture. In the single hop architecture, sensor nodes do not support communications on behalf of other sensor nodes. They are directly connected with a cluster head or forwarding node which will forward the data to the terrestrial network. Military and environmental monitoring applications typically use multi-hop network architecture to extend the coverage area of the applications. In the multi-hop architecture, sensor nodes have an additional role of forwarding data from other sensor nodes towards their final destinations, i.e. performing a routing function [5].

WSNs are also considered as one of the key enablers for the Internet-of-Things (IoT) paradigm. Diverse WSN and IoT devices, including sensors and actuators, smart meters and industrial machines are increasingly being interconnected and integrated with the Internet to form a converged network infrastructure. With such convergence, the massive amount of data generated from these heterogeneous devices can be harnessed for new business applications such as monitoring and tracking, smart grid energy management, supply chain management, surveillance, etc. [6].

The operational environment of WSNs poses additional constraints and challenges to its researchers and developers, as compared to traditional wireless networks. As sensor nodes are typically deployed in large quantities in unmanned areas, and in most cases are non-retrievable, the cost and the operational considerations limit the physical form factor, processing capability, memory size, and battery power of the sensor nodes in terms of hardware.

In terms of communications, one of the key design considerations for sensor network communication protocols is the low power consumption requirement. Therefore, while traditional networks aim to achieve high throughput and high quality of service (QoS), WSN protocols focus on efficient energy consumption and a long network lifetime at the cost of lower throughput and longer network delays.

1.2 WSN Protocol Stack



Source: I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," Computer Networks, vol. 38, pp. 393–422, 2002. [4]

Fig. 1.1 Generic protocol stack for sensor networks

In general, a WSN comprises a large number of sensor nodes that are scattered in the area of operation to collect location specific data and route it back to a central station for processing, analysing and decision making. Due to the absence of an infrastructure, the data is routed back via the sensor nodes themselves in a multi-hop operation.

Fig. 1.1 shows the generic protocol stack used by sensor nodes with five protocol layers. Different types of application software can be built and used at the application layer depending on the WSN applications. The Transport layer maintains the flow of data if the sensor networks application requires it. The Network layer determines the optimal paths to be taken by the data. The Media Access Control (MAC) layer protocol provides error control and recovery schemes for data transmission in a noisy wireless medium. The Physical layer takes care of the modulation and coding, transmission and receiving techniques. The power management, mobility management, and task management planes work together to help the sensor nodes coordinate the sensing task and lower the overall power consumption [4].

1.3 Energy Efficient WSN Protocols

There are three major subsystems in a sensor node, the computation subsystem, the communication subsystem and the sensing subsystem. In general, the communication subsystem has much higher energy consumption than the computation subsystem [7]. Depending on the specific application, the sensing subsystem might be another significant source of energy consumption, so its power consumption has to be reduced as well. There is a plethora of research studies conducted on improving the energy efficiency in the communication sub-system of WSNs, ranging from optimizations of the physical layer radio to techniques for the application layer data reduction.

In the Physical layer, the coding and modulation schemes, transmission power control and antenna configurations are some of the radio parameters that can be optimised for energy consumption efficiency. In the MAC layer, the most common approach is to put the nodes into a Sleep mode when possible to save energy. This includes duty-cycling schemes, separate low power wake-up radio, and topology control, to minimize the number of active nodes. Network layer routing is another source that drains the energy reserves of the sensor nodes. In particular, in multi-hop networks, energy consumption is uneven among the different nodes. Nodes that need to route more packets will have their energy depleted in a shorter amount of time. Energy-aware routing schemes include clustering techniques, energy-aware routing metrics, multi-path routing, etc. In the Application layer, data reduction techniques are generally used. These include data aggregation, adaptive sampling, network coding, and data compression. These solutions effectively reduce the data traffic in the sensor network and hence prolong the network lifetime [1].

Over the years, there have also been an increasing number of papers that focus on the cross-layer development of WSN protocols. The cross-layer design typically focuses on pairing two or more layers, and exploits the dependencies and interactions across these layers for joint optimisation. Among them, cross-layer interactions between the MAC and the Network layers are most commonly exploited in multi-hop WSNs. However, while most of these cross-layer solutions may yield performance improvements, these results are often obtained at the expense of decreasing the architectural modularity, which restricts further development and improvements [8][9].

1.4 Objectives and Methodology

The main objective of this work is to design and develop energy-efficient algorithms for multi-hop WSNs that are capable of supporting a wide range of application scenarios. Energy consumption in the Network and MAC layers are the key contributors to the total communication energy consumptions of the sensor nodes and will be the key focus of this work.

In this work, existing Network and MAC layer energy-efficient protocols are first studied and analysed to identify the areas where energy efficiency can be improved. Network models for the proposed new protocols and algorithms are then developed, and their performances are evaluated against the existing protocols through mathematical models and simulations.

Network Simulator ns-2 [10] developed through the VINT project by University of Southern California and its collaborative partners, is used extensively in this study. ns-2 is an open-source event driven simulator designed specifically for research in computer communication networks, and is by far the most popular simulator used in ad-hoc and sensor network simulations due to its flexibility and modularity [11]. It provides substantial support for the simulation of TCP, routing, and multicast protocols over both wired and wireless networks and is widely used by the researchers to simulate a wide variety of networking protocols.

In the Network layer, routing algorithms are responsible for delivering the data from the source nodes to the destination node along selected paths. In this work, a new routing algorithm is proposed to optimize the network lifetime of WSNs by distributing the data load among different intermediate sensor nodes so that the energy consumption is more evenly distributed in these nodes. The distribution of energy consumption based on the energy levels of each node prevents early exhaustion of some sensor nodes, which might otherwise segment the network and lower the data delivery performance.

In the MAC layer, sensor nodes spend most of the time idling and sensing the radio channel for data signals. This idle sensing is the dominant source of energy consumption in the MAC layer. Duty cycling is one of the key mechanisms used in the MAC layer to reduce energy wastage stemming from idle listening and thus improve network lifetime. On the other hand, energy consumed for the synchronisation process in duty-cycle MAC protocols is substantial; therefore, an energy-efficient synchronisation algorithm can improve energy performance significantly. In this work, new synchronisation algorithms are proposed to optimize the energy performance of duty-cycle MAC layer protocols. Performance evaluation criteria include packet delivery ratio, end-to-end network delay, and algorithm stability, in addition to energy consumption.

To test the proposed protocols and algorithms, new modules in the Network and MAC layers are developed and incorporated into the existing protocols in ns-2. This integration enables the simulations of the different protocols of interest using the same network environment, topology, and parameters, so that the performance of these protocols can be compared fairly. A wide range of network scenarios with different network sizes, data loads, traffic patterns, network densities, and duty cycles are simulated and their performance analysed.

1.5 Contributions

The main contributions of this work are summarised as follows:

- A new energy-balanced routing protocol is proposed. The proposed routing protocol, which is termed Energy-balanced Dynamic Source Routing (EB-DSR), has the ability to perform multi-path routing dynamically based on the node energy status to maximize the network lifetime of WSNs [12].
- An energy monitoring and distribution mechanism is needed for the communication of node energy status among different sensor nodes in energy-aware routing protocols. This mechanism will consume additional energy and bandwidth but has not been considered in the proposals of many existing energy-aware routing protocols. On the other hand, we have designed and integrated the energy monitoring and distribution mechanism into the proposed routing protocol without generating substantial bandwidth and energy overheads [12].
- A new and energy-efficient synchronisation algorithm that integrates well with synchronous duty-cycle MAC protocols is proposed. This approach offers the synchronous duty-cycle MAC protocols a new dimension in reducing energy consumption, in addition to the existing energy-efficient data algorithms in the MAC layer [13][14].
- Analytical models for energy consumption behaviour, as well as synchronisation performance of the existing and proposed synchronisation algorithms in single-hop neighbourhoods, are developed for both unsaturated and saturated neighbourhoods. The analytical models are validated via network simulations [13].
- To address the different synchronisation challenges posed by the wide range of network densities in WSNs, an adaptive synchronisation algorithm is developed. This algorithm effectively reduces congestion and collisions when synchronisation (*sync*) packet traffic is high, and maintains synchronisation performance when *sync* packet traffic is low. The algorithm enables the duty-cycle MAC protocols to support a wide range of WSN networks and applications. Even within a single large multi-hop WSN, different

neighbourhoods will have different densities; an adaptive synchronisation algorithm will thus be desirable to deliver better energy performance [14].

- For the first time, end-to-end multi-hop network simulations are performed against sensor nodes with different clock drifts and duty cycles to analyse the sensitivity and stability of the synchronisation algorithms. Energy performance and data performance including packet delivery ratio and packet delay are evaluated and analysed [14].
- The new synchronisation algorithm successfully lowers the duty-cycle limit of synchronous MAC protocols, extending the effectiveness of synchronous MAC protocols to a wider range of applications [14].

1.6 Thesis Overview

The remainder of this thesis is organised into the following chapters.

Chapter 2 introduces the different energy efficient WSN protocol implementations and provides a brief description of related work in each of these implementations.

Chapter 3 is dedicated to the Network layer of WSNs. The challenges of extending network lifetime in heterogeneous, multi-hop WSNs are identified and a new energy-balanced routing metric is proposed to improve network lifetime performance. A new dynamic source routing protocol incorporating an energy-balanced routing metric is implemented in ns-2, and the simulation results are examined and compared with the performance of existing routing protocols.

Chapter 4 focuses on the MAC layer of WSNs and addresses time synchronisation issues in duty-cycle WSNs. Synchronisation needs and processes in duty-cycle MAC protocols are discussed. Energy consumption models for the existing and proposed duty-cycle MAC synchronisation algorithms in single-hop neighbourhoods are also developed and analysed. The analytical models are then validated using ns-2 simulations and the results are discussed in detail.

In Chapter 5, the study of network synchronisation is extended to multi-hop multi-neighbourhood WSNs of different densities, clock drifts, and duty cycles. An adaptive synchronisation algorithm is designed to address the high energy consumption issues in high density networks. In addition to energy consumption, effects of the synchronisation algorithms on performance such as packet delivery ratio and end-to-end packet delay are also studied.

Finally, Chapter 6 concludes this thesis, and describes the directions for future research work in this area.

Chapter 2

Background and Related Work

2.1 Introduction

This chapter introduces a brief summary of previous studies that are directly related to the proposed work in this thesis. First the background of Wireless Sensor Networks (WSNs) and their design and implementation challenges are discussed. The two different approaches of addressing limited energy supply to maximise network lifetime through energy harvesting and efficiency in node energy consumption are also briefly discussed.

Different literatures in energy-efficient WSN communications in the application (data), network and MAC layers are reviewed. Section 2.3 discusses the various WSN routing protocols including proactive, reactive and hybrid routing protocols. Proactive protocols maintain routing information for every node in the network independent of data requirements (section 2.3.1). Reactive protocols discover routes to the destination nodes only when they are needed by the application (section 2.3.2). Hybrid protocols are combinations of proactive and reactive protocols (section 2.3.3).

Section 2.4 reviews the different algorithms that are incorporated into the routing protocols to make them energy-efficient. These include energy-aware, maximum lifetime and load balancing algorithms. Energy-aware routing algorithms make use of the energy information to select the lowest energy cost path (section 2.4.1), whereas maximum lifetime routing algorithms make use of the node residual energy information to avoid paths that contain nodes with low residual energies (section 2.4.2). Load balancing algorithms are traditionally employed to address network congestion problems; however, the effect of distributing data transmission along multiple paths improves the network lifetime to some extent (section 2.4.3).

Section 2.5 discusses duty-cycle MAC protocols, in which sensor nodes alternate between active and sleep periods to conserve energy. There are two types of duty-cycle MAC protocols: synchronous and asynchronous. Synchronous duty-cycle MAC protocols (section 2.5.3) make use of a MAC layer synchronisation algorithm (section 2.5.2) to synchronise sensor nodes in the same neighbourhood, so that they can wake up at the same time to exchange sensor data. On the other hand, asynchronous duty-cycle MAC protocols (section 2.5.4) do not use a synchronisation algorithm. Depending on the protocol, acknowledgement from either the transmitter or the receiver is needed to start the data transmission.

Section 2.6 discusses data-centric protocols, which are commonly used to remove data duplication and hence reduce data transmission in query-response based WSNs.

Finally a summary of this chapter is provided in section 2.7.

2.2 Background

WSNs are one of the first real-world examples of integrating the digital and physical world. The combination of distributed sensing, computing, and wireless communications enables a broad range of applications that are not seen in a purely digital world. WSN is also considered as one of the key enablers for the Internet-of-Things (IoT) paradigm which has garnered significant media attention in recent years. According to a report by Gartner, 20.8 billion heterogeneous devices embedded with electronics, software and sensors will be connected by 2020, up from 6.4 billion devices in 2016 [15].

There are three key areas in the studies of energy efficiency in IoT, namely IoT communication protocols, low power wide area networks (LPWANs) and WSNs.

i. **IoT communication protocol**: This is the application layer protocol used by the IoT software applications. Due to the constraints of limited bandwidth and energy capacities of IoT network devices, lightweight communication protocols such as Constrained Application Protocol (CoAP) [16], Message Queuing Telemetry Transport (MQTT) [17], and others have been designed [18]. Experimental results in [19] have shown that CoAP is more efficient in terms of energy consumption and bandwidth usage while MQTT provides high reliability. Both protocols possess the low overhead for header parsing; however optimized encoding for payload compression is a further challenge to be resolved.

ii. **LPWANs:** LPWANs provide long range, low data rate, and energy efficient wireless communication to complement traditional cellular and short range wireless technologies in addressing diverse requirements of IoT applications. A very long range of LPWAN technologies enables devices to spread and move over large geographical areas. IoT devices connected by LPWAN can be turned on anywhere and anytime to sense and interact with their environment instantly.

   Two leading examples in this area are SigFox, which uses an ultra-narrowband (UNB) solution and LoRa, which uses a chirp spread spectrum (CSS) solution [20][21]. By using UNB, SigFox utilizes bandwidth efficiently and experiences very low noise levels, resulting in high receiver sensitivity, ultra-low power consumption, and inexpensive antenna design. However, these benefits come at the expense of maximum throughput of only 100 bps. LoRa supports multiple spreading factors for the trade-off between range and data rate based on application needs. The data rate ranges from 300 bps to 37.5 kbps depending on spreading factor and channel bandwidth [21]. In addition to SigFox and LoRa, a number of other LPWAN technologies are also surveyed in [22]. To achieve low energy consumption, the communication bandwidth of the LPWAN solutions is usually narrow and hence the data rate of the current LPWAN solutions is low. The trade-off among energy efficiency, data rate and communication range remain a key challenge in this area [22].

iii. **WSNs:** Design and implementation of WSN applications have to address different dimensions of challenges which include sensor node computation and storage capabilities, cost and size of each node, wireless communications, sources of energy, protocols for data dissemination and communication, applications and management tools, etc. A typical and widely deployed application category is one that uses battery-powered sensor nodes [3], untethered sensor nodes are commonly used in these deployments to facilitate mobility and deployment in hard-to-reach locations. A major limitation of these sensor nodes is finite battery capacity. This implies finite lifetime of the applications or additional cost and complexity to change batteries regularly.

A comparative review of several commonly used wireless sensor network motes is presented in [23]. The power consumption of the radio modules in these sensor motes is in the range of 10 – 60mA for transmission mode, 74µA – 40mA for idle and reception mode, and 20nA – 1.4mA for sleep mode. These motes typically use AAA batteries, AA batteries or 750 mAh rechargeable lithium ion batteries. It is clear that the batteries will not last for a very long time if the sensor motes are operating in the active modes continuously.

There are two fundamental approaches to address the issue of limited battery capacity. The first approach is the use of energy harvesting. Energy harvesting refers to harnessing energy from the environment and converting it to electrical energy. If the harvested energy source is large and continuously available, a sensor node can be powered perpetually. The second is to reduce the energy consumption of WSN through the use of low-power hardware and energy-efficient communications.

2.2.1 Energy Harvesting in WSNs

In general, energy harvesting can be divided into two architectures: Harvest-Use and Harvest-Store-Use architectures [24]. In the Harvest-Use architecture, the harvesting system directly powers the sensor node. For the sensor node to be operational, the power output of the harvesting system has to be continuously higher than the minimum operating power. The node will be disabled if sufficient energy is not available. The Harvest-Store-Use architecture consists of a storage component that stores harvested energy and also powers the sensor node. Energy storage is useful when the harvested energy available is more than its current usage. Energy stored can then be used later when either there is no harvesting opportunity or energy usage of the sensor node has to be increased to improve capability and performance parameters.

A critical component of energy harvesting architecture is the energy source. Energy sources that can be harvested from the ambient environment can be broadly classified into mechanical energy, radiant energy, thermal energy and fluid flow [25]. Mechanical energy is based on kinetic energy or motion of an object, which include vibrations, pressure, and human activity. The kinetic energy arises from these sources can be converted to electrical energy using a converter such as piezoelectric, electrostatic, or electromagnetic converters. Radiant energy comes from electromagnetic waves such as sunlight and radio frequency (RF) signal. Solar energy has gained popularity in energy harvesting since solar energy is readily available and can be harvested using photovoltaic or solar cells. An RF energy harvesting device can harvest the energy from the signal emitted by a dedicated RF transmitter or from the ambient source such as the base station antenna, radio and TV signal, WiFi [26], and mobile devices. Thermal energy is based on the temperature gradient of the environment. Thermoelectric and pyroelectric transducers are typically used to convert thermal energy to electrical [27]. The Wind and water flow energy can be classified under fluid dynamic or fluid flow. Energy from these sources can be harvested using turbine or piezoelectric converters [28][29].

Energy sources have different characteristics such as controllability, predictability and magnitude [30]. A controllable energy source can harvest sufficient energy whenever it is needed, energy availability need not be predicted before harvesting. For non-controllable energy sources, energy must be simply harvested whenever available. In this case, if the availability of the energy source is predictable, a prediction model can be used to indicate the time of next recharge cycle.

A WSN with the capability of energy harvesting sensor nodes to supplement the battery energy supply can potentially operate in the energy neutral mode in which the system uses only as much energy as is available from the environment to sustain its operation. In the case that this is not achievable, energy harvesting can be used to improve the lifetime of the WSN by incorporating a prediction model and power management techniques into the design [25][30].

2.2.2 Energy-efficient Communications in WSN

In general, energy efficiency in WSN communication can be accomplished in three ways:

i. **Multi-hop Communication**: Since the energy required for wireless communications increases as a power law over distance, multi-hop communication is adopted to reduce the transmission range without sacrificing network reachability [31]. Routing is an essential component to support self-organising, multi-hop communications.

ii. **Low duty-cycle operation**: The basic idea of the low duty-cycle operation is to reduce power consumption by putting a sensor node to sleep when there is no data to transmit or receive [32]. Duty-cycle is measured as the percentage of active period in a complete cycle which includes the sleep period. A small duty-cycle means that a node is asleep most of the time; however, it also increases end-to-end delay and a balanced approach is needed to meet application specific requirements.

iii. **Data-centric protocols**: Data-centric or data aggregation protocols are commonly used in query-response WSNs, where multiple source nodes are available to provide responses to a single query. When the source nodes send their data to the sink, intermediate sensor nodes can perform some form of aggregation on the data originating from multiple sources and send the aggregated data toward the sink. The aggregation process helps to eliminate redundancy, minimise the number of transmissions and therefore reduce energy consumption [33]. This is different from the traditional address-centric approach of finding short routes between pairs of two end nodes.

The data-delivery model of WSN from source to sink can be continuous, event-driven, query-driven, or a hybrid of the latter two, depending on the application. The choice of routing and MAC protocols is highly influenced by the data-delivery model, especially with regard to energy efficiency and route stability. For an environmental monitoring application where data is periodically transmitted to the sink, a hierarchical routing protocol with data-processing capability is the most efficient approach. This is because such an application generates a significant amount of redundant data that can be aggregated along the routed path, thus reducing traffic and energy. For event-based data-delivery models, a contention based duty-cycle protocol is a good fit [34], since data is generated infrequently.

2.3 Multi-hop Routing in WSNs

Routing is the process of determining an optimal path to transport data information, in the form of packets, to traverse a network between a source and a destination. Routing protocols use metrics, a form of measurement, to determine the best paths among multiple alternatives. To aid the process of path determination, routing tables, which contain network route information, are built and maintained by routing protocols. To keep the route information up to date, routing update messages are sent either periodically or when a change in the network topology is detected, depending on the routing protocols.

The key design goals of routing protocols/algorithms can be summarised as follows:

i. **Optimisation**: This is the capability of the algorithm to select the best route based on the selected metrics used in the calculations.

ii. **Simplicity:** An efficient routing algorithm should have minimum CPU time, memory and bandwidth overhead. This is important so that it can be scaled to large networks.

iii. **Rapid convergence**: When there is a change in network topology due to some network events, convergence measures how fast the routing protocols can obtain updated route information to re-establish network connectivity. Routing protocols that converge slowly can cause data loss and affect packet delivery ratio.

iv. **Robustness and Stability**: A routing algorithm should perform correctly in unusual or unforeseen circumstances, such as overload conditions and node failures.

The development of routing in wireless networks led to the emergence of Mobile Ad Hoc Networks (MANETs). MANETs share many properties with WSNs and have substantially influenced the development of WSN routing protocols. There are three major groups of multi-hop MANET routing protocols: proactive routing protocols, reactive routing protocols and hybrid routing protocols [35][36].

2.3.1 Proactive Routing Protocols

A proactive routing protocol is also known as table-driven routing protocol. Each node in a proactive routing network maintains routing information to every other node in the network. The routes to the other nodes are usually determined at the start up, and maintained using a periodic route update process. Alternatively, the routes are updated when the network topology changes. In general, proactive routing protocols come with higher overhead in most scenarios because of frequent updates. However, they have lower latency of packet forwarding compared to reactive protocols because the route is available when it is needed. A comprehensive survey of proactive routing protocols can be found in [36].

2.3.1.1 Destination-sequenced Distance Vector Routing Protocol (DSDV)

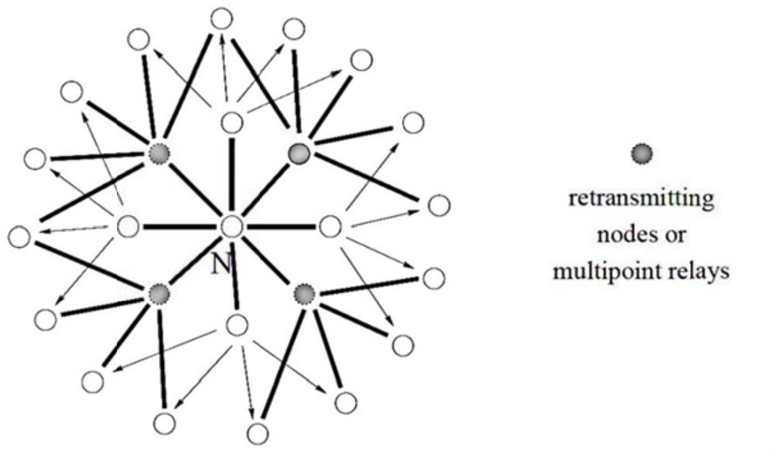
DSDV [37] is a hop-by-hop distance routing protocol based on the Bellman-Ford algorithm. Every node periodically transmits routing updates to maintain the routing table consistency. The key difference between DSDV and traditional distance-vector routing protocols is that the route entries are tagged by a sequence number assigned by the destination nodes in order to guarantee loop-free routing in the wireless environment. The sequence number indicates the freshness of routes with the same destination; a higher sequence number is more favourable compared to a lower sequence number. In the event that two routes have the same sequence number, the route with the smaller metric is used.

A routing table update generates a lot of overhead traffic; DSDV addresses this problem by using two types of routing update packets. The first is known as a full dump, which carries all routing table information. These packets are large and transmitted relatively infrequently if the network is stable. The second type of updates uses smaller incremental packets that consume less bandwidth to relay only the information that has changed since the last full dump.

2.3.1.2 Optimized Link State Routing Protocol (OLSR)

OLSR[38][39] is a table-driven, proactive routing protocol, which uses Hello and Topology Control (TC) messages to discover and then disseminate link state information throughout the mobile ad-hoc network. The protocol inherits the stability of a link state algorithm and being a proactive protocol, it has the advantage of having routes readily available when needed. Being a link-state protocol, OLSR consumes a reasonably large amount of bandwidth and CPU power for optimal path computation.

OLSR uses the concept of multipoint relays (MPRs) to minimise the overhead of flooding messages in the network by reducing redundant retransmissions in the same region. OLSR makes use of "Hello" messages to find its 1-hop and 2-hop neighbours through their responses. A subset of the 1-hop neighbours is selected as MPRs to retransmit the messages. The MPR set of a node N is selected in such a way that all the 2-hop neighbours of N are within 1-hop distance of the MPRs, as shown in **Fig. 2.1**. The smaller the MPR set is, the more optimal the routing protocol. The neighbours of node N which are not in its MPR set receive and process broadcast messages received from node N, but do not retransmit them.



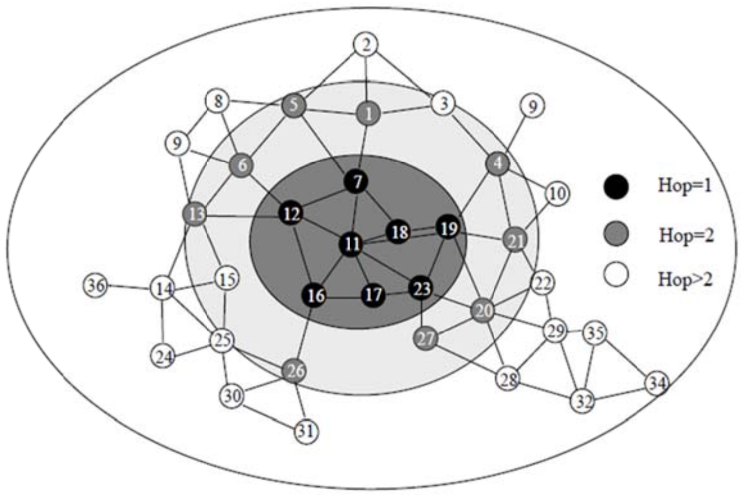
Source: Optimized link state routing protocol for ad hoc networks [39]

Fig. 2.1 Selection of Multipoint relays in OLSR to cover all 2-hop neighbours of node N

OLSR, like any link-state routing protocol, makes use of link-state information to build a topology table; which is subsequently used by a node to build its routing table. OLSR nodes broadcast specific TC messages, which are retransmitted throughout the entire network for the advertisement of link-state information. However, OLSR takes advantage of MPRs, which reduces control traffic overhead and improves scalability.

2.3.1.3 Fisheye State Routing Protocol (FSR)

FSR [40][41] is a table-driven link state routing protocol. FSR takes inspiration from "fisheye" where pixels near the focus are captured with more details; the details reduce as distance from the focal point increases. In terms of routing, the fisheye approach maintains accurate distance and path quality information about nodes in the immediate neighbourhood, and gives fewer details progressively when the distance increases. This is achieved by propagating routing entries that correspond to nodes within the smallest distance to the neighbours with the highest frequency; the rest of the entries are sent out with lower frequencies. **Fig. 2.2** illustrates the application of fisheye technique in a MANET. The nodes in the different zones are under different fisheye scopes with respect to the central node (node 11). The scope is defined as the set of nodes that can be reached within a given number of hops, each with a different routing update frequency. In this way, the routing update overhead is reduced by using different exchange frequencies for different entries in the routing table.



Source: Scalable routing strategies for ad hoc wireless networks [40]

Fig. 2.2 Different Fisheye scopes with respect to node 11 shown as different coloured regions.

FSR exhibits very good scalability because it does not attempt to maintain the same knowledge level of link states for all nodes in the network. Although the identification of the optimal path to the distant node is imperfect, the packets will still be routed correctly because the route information becomes more accurate as the packets get closer to the destination.

2.3.2 Reactive Routing Protocols

A reactive routing protocol is also known as an on-demand routing protocol. Route information is discovered when needed. This means that routes are determined and maintained only for nodes that need to send data to a particular destination. The control traffic overhead is thus reduced compared to a proactive routing protocol.

When a source node attempts to transmit a data packet to a destination where a route is not found, it begins a route discovery process by flooding a route request packet towards the destination through the network. When a node with a route to the destination (or the destination itself) receives this route request, it sends a route reply to the source node using the reversed link, and thus a routing path is established.

Reactive routing protocols have been the protocols of choice in mobile ad hoc networks due to frequent node mobility. They have also been the predominant design choice in wireless sensor networks due to their simplicity and support for data on-demand.

2.3.2.1 Dynamic Source Routing Protocol (DSR)

DSR [42] is an efficient reactive routing protocol designed specifically for use in multi-hop wireless ad hoc networks. "Route Discovery" and "Route Maintenance" are two main processes in the protocol that work together in order to allow nodes to discover and maintain routes to arbitrary destinations in the ad hoc network. An advantage of DSR is that nodes can store multiple routes in their route cache. If a valid route is found in a node's route cache, there is no need for route discovery and this saves a considerable amount of bandwidth in the network, especially in a low mobility network.

DSR uses source routing; when a new data packet is generated, the source node determines the complete route to the destination. It places the hop-by-hop information in the packet header and the intermediate nodes simply forward the packet based on this routing information. With multiple routes in the route cache, the source node is able to select and control the routes used in routing its packets, an ability which can be used in load balancing or for increased robustness. Other advantages of DSR include the support of unidirectional links, as well as rapid discovery when routes in the network change.

2.3.2.2 Ad hoc On-Demand Distance Vector Routing (AODV)

AODV [43] is an on-demand routing protocol which uses a similar route discovery procedure as that in DSR. However, AODV has a very different mechanism to maintain routing information. It uses traditional routing tables, one entry per destination, and uses hop-by-hop routing. In DSR, however, multiple route cache entries for each destination are maintained and end-to-end routes are determined by the source. Similar to DSDV, AODV uses sequence numbers maintained at each destination to determine the freshness of the routes and to prevent routing loops.

The advantage of AODV is that it is adaptable to high mobility networks. However, large delays may be experienced by sensor nodes during route construction and link failure may trigger another route discovery, which introduces extra delays and consumes more bandwidth.

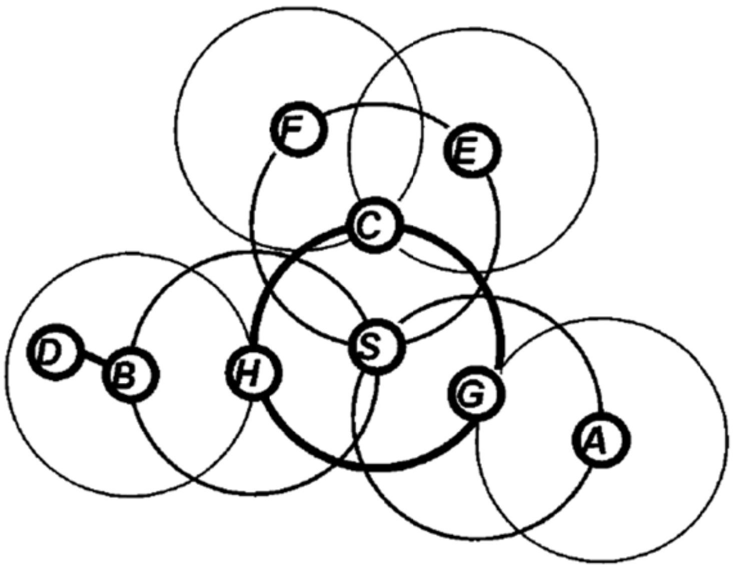
2.3.3 Hybrid Routing Protocols

Hybrid routing protocols constitute both proactive and reactive routing processes. These protocols are designed to increase scalability by allowing nodes in close proximity to work together to form some sort of a backbone. Nodes that are nearby maintain routes proactively to reduce the route discovery overheads. Routes to nodes that are far away are determined by a route discovery process, similar to a reactive protocol.

2.3.3.1 Zone Routing Protocol (ZRP)

In ZRP [44], each node has a routing zone, which defines a range (in hops) in which each node is required to proactively maintain network connectivity. For nodes within the routing zone, the Intra Zone Routing Protocol (IARP) implements link-state routing that provides a complete view of network connectivity. Therefore, routes are immediately available within the routing zone. Outside the routing zone, routes are determined on-demand, and any on-demand routing protocol can be used to determine a route to the required destination.

The advantage of ZRP is that the amount of routing overhead is significantly reduced compared to pure proactive protocols. By enabling routes to be discovered faster, the delays are also reduced in the ZRP in contrast with pure reactive protocols. To determine a route to a node outside the routing zone, ZRP uses the bordercasting process for its inter-zone routing protocol (IERP). Bordercasting is a process that allows a node to send packets to its peripheral nodes (nodes on the routing zone boundary). Route discovery is efficiently done by bordercasting a route query to all the peripheral nodes of the source node, which in turn bordercast the query to their own peripheral nodes and so on. Once the destination is found, a route reply is sent back to the source node. The routing path, which consists of a list of peripheral nodes between the source and the destination, will be stored in the packet header or cached in the queried peripheral nodes [45]. **Fig. 2.3** shows an example of IERP from the source node S to the destination node D.



Source: *The performance of query control schemes for the zone routing protocol* [45]

Fig. 2.3 An example of ZRP inter-zone routing from S to D via peripheral nodes H and B using bordercasting

2.3.3.2 Zone-based Hierarchical Link State (ZHLS)

ZHLS [46] is a zone-based, hierarchical routing protocol incorporating location information. The network is divided into non-overlapping zones and each node has a two level hierarchical address, a node ID and a zone ID. Each node knows its own position and therefore its zone ID through global positioning system (GPS). Unlike other hierarchical protocols, there are no cluster heads. High level topological information is transmitted in a peer-to-peer manner to all nodes. This means that a single point of failure, as well as traffic bottlenecks, can be avoided.

ZHLS is proactive for destinations within the same zone as a source, and reactive for remote destinations. When there is a need to route to a destination node in another zone, the source node broadcasts a zone-level location request to all other zones, which generates significantly lower overhead compared to the flooding approach used in reactive protocols. The disadvantage of ZHLS is that a GPS is required in all nodes which may not be feasible in many WSN applications.

2.3.3.3 Distributed Dynamic Routing Protocol (DDR) [47]

Similar to ZRP and ZHLS, DDR is a hybrid hierarchical routing protocol based on zones. Unlike ZRP, the zones in DDR do not overlap, and each node needs to know only the next hop to all the nodes within its zone. Unlike ZHLS, DDR does not need location information for routing, and each node keeps only the zone connectivity of its neighbouring zones whereas in ZHLS, each node maintains the zone connectivity of the whole network.

DDR executes a six-phase process to build a two-level forest and tree structure: preferred neighbour election, forest construction, intra-tree clustering, inter-tree clustering, zone naming and zone partitioning. During the initial phase, each node carries out the preferred neighbour election algorithm, choosing a neighbour that has the most number of neighbours as its preferred neighbour. Next, a forest is constructed by connecting each node to its preferred neighbour. After that, the intra-tree clustering algorithm is executed to form a tree structure within a zone and build the intra-zone routing table. The inter-tree clustering algorithm is then executed to determine the connectivity with the neighbouring zones. Finally to complete the process, each tree is assigned with a name by executing the zone naming algorithm. After the structure is completed, hybrid ad hoc routing protocols (HARP) [48] will use the intra-zone and inter-zone routing tables created by DDR to determine a stable path between the source and the destination.

2.4 Energy-efficient Wireless Routing Protocols

A WSN is a network of sensor nodes connected via wireless communications. These sensor nodes often have limited energy capacities; therefore one of the most important considerations of a routing protocol in WSN is the energy consumption efficiency and the extension of the network's lifetime. Most commonly used ad hoc routing protocols such as DSDV, ADOV, DSR and other routing protocols described in the previous section use hop-count as the metric for route selection which does not take into account energy consumption. With the rapid increase in the demand of WSN applications, many energy-efficient routing protocols have since been proposed for WSNs.

2.4.1 Energy-aware Routing

Energy-aware routing is a class of routing protocols that makes use of energy information to make routing decisions, in order to improve the energy performance of the network.

An on-demand Minimum Energy Routing Protocol (MER) [49] uses a transmission power control approach to determine the minimum energy route. MER implements the transmission power control mechanism in DSR and modifies the header of the Route Request packet to include the power used by the sender to transmit the packet. The receiving node uses this information, as well as the received power level, to compute the minimum power required for successful transmission in this link. The power information (per hop) is appended at each intermediate node towards the destination. This power information is sent back to the source node in the Route Reply packet along the reversed links. In this way, the source node and all the intermediate nodes along the path are able to transmit data packets with the minimum transmit power.

To differentiate between reliable and unreliable transmission links, a new link cost metric that is a function of both the energy required for a single transmission attempt across the link as well as the link error rate is defined in [50]. The link error rate factors in the potential retransmission cost needed for reliable data delivery. There are two operating models in this retransmission-aware algorithm, end-to-end retransmissions (EER) and hop-by-hop retransmissions (HHR). The EER model applies in the scenarios where the individual links do not provide link-layer retransmissions, and retransmissions due to errors are only initiated by the source node. The HHR model caters for networks where each individual link provides reliable forwarding via localised packet retransmissions.

The retransmission-aware algorithm proposed in [50] has only considered the energy cost of exchanging data packets, although common wireless protocols also require control packets for reliable data delivery. A more accurate energy model that accounts for total energy consumption of data packets, control packets and retransmissions is proposed in [51], and simulation results show that the energy performance is better using this more accurate energy model.

Energy efficiency in these variations of energy-aware routing protocols has improved. However, for a network with stationary nodes, these routing protocols almost always select the same path, which is the lowest cost path, for the same source-destination pair. The static behaviour of the path selection means that wireless nodes along the selected path have to work harder and consume more energy than those that are not on the selected path. The energies of these overworked nodes will naturally be depleted much faster than the other nodes. When this happens, there is a high possibility that the network will be partitioned, making some destination nodes unreachable. In wireless sensor network (WSN) applications, the exhausted sensor nodes are no longer able to perform their sensing function even if the network is not segmented. This reduces the effectiveness of the WSN.

An ant-based energy-aware routing (ABEAR) protocol based on the Ant Colony Optimization (ACO) is proposed in [52]. ABEAR is a multi-path protocol that considers the link-quality, congestion metrics and the remaining energy of the next hop to compute the routing path. Simulations results show that ABEAR performs better than AODV in network lifetime. However, ABEAR requires a proactive neighbour maintenance process which will increase its control overhead rapidly as the network density increases.

2.4.2 Maximum Lifetime Routing

There is another class of routing algorithms that aims to maximise the network lifetime. In [53][54], maximum network lifetime routing is modelled as a linear programming problem with the objective of maximising network lifetime, which can also be interpreted as maximizing the amount of information transfer between the origin and destination nodes given the limited energy. This translates into a shortest cost path routing whose link cost is a combination of transmission and reception energy consumption and the residual energy levels at the two end nodes. In [55], the model is extended to take into the consideration of energy-harvesting capability of the WSNs.

In [56], a heuristic *max-min* zP_{min} algorithm that combines path power consumption and path minimal residual energy is offered. Intuitively, to maximise network lifetime, routes should avoid nodes whose energy level is low because overuse of these nodes will deplete their battery reserves. However, the strategy of routing data along the path with the maximum minimum residual energy (known as max-min path) could lead to very poor results as shown in the example given in [56]. Also, routing along max-min path could be expensive compared to the path with the minimum power consumption. The proposed *max-min* zP_{min} algorithm relaxes the power consumption requirement along the selected path to be zP_{min}, where P_{min} is the power consumption along the minimum power consumption path and z is to be computed adaptively based on the algorithm. The author has simulated the algorithm with mathematical networks and has obtained good empirical results.

In [57], NCE-DSR(Number of times nodes send Constraint Energy DSR) protocol is proposed to prolong its network lifetime. NCE-DSR uses the number of messages that has been transmitted by a node i (N_i) as a proxy for the node residual energy based on the inverse relationship between the two. The routing metric used in this protocol is a combination of maximum (*N_max*) and average (*N_ave*) values of N_i along the routing path. Simulation results have shown that NCE-DSR has improved network lifetime compared to DSR.

Energy Dependent DSR (EDDSR) protocol proposed in [58] attempts to prolong its network lifetime by discouraging nodes with short lifetime from participating in the route discovery process. When a node has enough residual energy, it participates in the network activities behaving exactly as a DSR node. When the node’s residual battery capacity falls below a threshold, it delays rebroadcasting of a received RREQ by a time period that is inversely proportional to its predicted lifetime. The protocol has obtained some success in the scenarios simulated.

An Energy-efficient DSR (E-DSR) protocol proposed in [59] uses a metric that is based on a combination of node transmission power cost, path energy cost and link availability for route selection. Simulation results have shown that E-DSR has a better lifetime performance than DSR in networks with mobile nodes. However, the assumption that each node knows it coordinates and mobility is a major limitation of this protocol.

A conditional max-min battery capacity routing (CMMBCR) algorithm, using a different combination of power consumption and node residual energy, is proposed in [60]. Selection of routes in this scheme is conditioned on the minimum battery capacity of the nodes along the paths between a source and a destination. A path with minimum total transmission power is selected if routes exist that have the minimum battery capacity above some threshold value; otherwise, a path with maximum-minimum battery capacity will be selected. The algorithm is simulated with a simplified routing scheme to study the effects of the threshold value to network lifetime. [61] and [62] provide other variations of maximum lifetime routing algorithms.

Most of the maximum lifetime routing algorithms require accurate residual power information of all nodes in the network for route selection. However, the communication of such information could generate substantial overhead that consumes additional energy, and is not addressed in these algorithms.

2.4.3 Load Balancing Algorithms

Load balancing algorithms are traditionally employed to address network congestion problems to improve packet delivery ratio and reduce packet delay. There are single path and multi-path load balancing routing algorithms. Single path routing may discover multiple paths from a source to a destination but will only use the best path, according to the metric, for data forwarding. Load balancing is achieved over multiple flows that avoid using the same paths or nodes. Multi-path routing, on the other hand, distributes data packets over different paths for a single flow.

An example of single path load-balancing is the Load-Balanced Ad hoc Routing (LBAR) algorithm proposed in [63], which uses node traffic activity as a metric to distribute the load and to avoid routing via heavily loaded nodes. In LBAR, routing information on all paths from source to destination is forwarded through setup messages to the destination. Setup messages include nodal activity information of all nodes on the traversed path. With the collection of all nodal activity information, the destination makes a selection of the lowest cost path and sends an acknowledgement back to the source node. Simulation results in the paper have shown that LBAR has better packet delivery performance under high traffic conditions. However, as efficient energy performance is not its priority, LBAR utilises a substantial amount of control messages to achieve its objective.

Load Balanced Congestion Adaptive Routing (LBCAR) [64] is proposed to avoid congestion and increase the throughput of the network. LBCAR is a route selection algorithm using traffic load intensity and link cost as the routing metric. Simulation results in the paper have shown that LBCAR is able to reduce the end-to-end delay and enhance the throughput through balancing the load in the network.

The Multipath Routing Protocol (MSR) [65] is based on DSR and uses Round Trip Time (RTT) to measure delays for different paths, which form the basis of the routing metric. A source node then employs a weighted-round-robin scheduling to distribute the load. Using this scheme, both average RTT and throughput has shown improvements over the original DSR protocol.

A load balancing algorithm can be used to improve the network lifetime of WSN if an energy-aware metric is included as part of the route selection criteria. The idea of this approach is that while the total energy consumed in the network may not be reduced, total energy consumption is spread over a larger number of sensor nodes and thus the energy draining rate for individual nodes is slowed down. Load balancing algorithms for WSNs typically use combinations of new parameters such as transmit and receive energy costs, residual node energy, and link reliability to the routing metric in addition to the traditional hop count or link cost.

Traffic Load and Lifetime Deviation Based Power-Aware Routing Protocol (TDPR) proposed in [66] is a single path load balancing algorithm to prolong network lifetime. Path selection in TDPR is done by the destination node only during the route discovery process. The absence of an energy monitoring mechanism in this protocol during the data transmission phase limits its effectiveness in balancing node residual energy in the network.

2.5 Duty-cycle MAC Protocols

The main objective of the MAC protocol is to coordinate access to and transmission over a medium common to several nodes. In the wireless context the common medium is the wireless channel which is broadcast in nature. Any ongoing transmission will interfere with any other transmission within the communication range. Interference may lead to packet losses and retransmission mechanisms need to be catered for. To minimize interference and packet collisions, appropriate MAC rules have to be put in place.

To reach the agreement as to which node can access the communication channel at any given time, the nodes must exchange some amount of coordinating information. However, the exchange of information requires the use of the communication channel itself. This recursive aspect of the multi-access medium problem increases the complexity and overhead of the MAC protocol.

The key design goals of MAC protocols can be summarised as follows:

i. **Delay:** Delay refers to the amount of time spent by a data packet in the MAC layer before it is transmitted successfully.

ii. **Throughput**: Throughput is defined as the rate at which messages are serviced by a communication system. It is usually measured either in messages per second or bits per second.

iii. **Robustness:** Robustness, defined as a combination of reliability and availability requirements. It reflects the degree of the protocol insensitivity to errors.

iv. **Stability**: Stability refers to the ability of a communications system to handle fluctuations of the traffic load over sustained periods of time.

v. **Energy efficiency**: This is one of the most important issues in the design of MAC protocol for wireless sensor nodes. Energy-efficient MAC protocols achieve energy savings by controlling the radio to eliminate or reduce energy wastage such as collisions, idle listening, management and control packet overhead, etc.

Although a variety of MAC protocols have been developed for wireless networks, many are not suitable for WSNs because they were not designed with energy conservation as a primary goal [67].

A typical WSN generates very light traffic and sensor nodes spend most of the time listening to the radio channel and idling. This idle listening is the dominant source of energy consumption in WSNs. Duty-cycling is a common approach used in the MAC layer to reduce energy consumption due to idling, where sensor nodes alternate between active and sleep periods. Sensor nodes schedule the transmission and reception of data during active periods, and switch the radio off completely during sleep periods to conserve energy. References [68]–[70] provide comprehensive reviews of duty-cycle MAC protocols for WSN.

In addition to idle listening, overhearing of uninteresting packets, management packet overheads and collisions also waste power. It is important to identify these causes because while attempting to reduce idle listening, duty-cycling can increase the collision rates and introduce more management traffic, hence increasing energy consumption and reducing the effectiveness of the duty-cycle mechanism designed to reduce energy consumption.

2.5.1 Challenges of Duty-cycle MAC Protocols

Many different duty-cycle MAC protocols have been proposed during the last decade and new ones are still being published. Researchers aim to achieve very low duty cycles for energy conservation. However, this goal is achieved with the trade-off of other network performance parameters such as end-to-end delay, throughput or robustness. As WSN applications have diverse network performance requirements, one duty-cycling mechanism designed for a specific application may not work well with another. Some of the challenges of duty-cycle MAC are summarised as follows [32]:

i. **End-to-end Delay**: Data traversing a duty-cycle multi-hop network will potentially encounter a situation in which the next hop is sleeping, and will have to wait for it to wake up. This may add significant latency to the packet delivery time which is not tolerated by some applications such as surveillance, where a given event needs to be communicated in a timely fashion.

ii. **Collision Rate**: With duty-cycling, transmission and reception windows are shortened. If a contention-based MAC is used, these smaller time windows will increase the probability of collisions, reducing throughput and increasing latency further. If TDMA is used, a more precise clock synchronisation process will be needed, which means an increase in control traffic and an increase in energy consumption for transmitting and receiving such traffic.

iii. **Management Traffic Overhead**: Duty-cycling may require additional management traffic. The most common source of this overhead is synchronisation. Fine-grained synchronisation requires frequent resynchronisation to deal with clock skews. Protocol designers need to ensure that the energy saved from duty-cycling is not drained by the additional management traffic overhead.

There are two main categories of duty-cycle MAC protocols. The synchronous duty-cycle [71] approach makes use of a MAC layer synchronisation algorithm to synchronise sensor nodes in the same neighbourhood, so that they can wake up at the same time to exchange sensor data. On the other hand, the asynchronous or preamble sampling approach [72] does not use a synchronisation algorithm, but places the burden of data delivery on the senders. When a sensor node has data to send, it has to first transmit a preamble that is longer than the sleep period of the receiver so that the receiver will be able to detect it. Once the preamble is detected, the receiver will stay awake to receive the data. This approach may also increase the delay significantly as the sender has to meet the receiver's active schedule. B-MAC [73], X-MAC [74], and WiseMAC [75] are some examples of asynchronous duty-cycle MAC protocols.

Synchronous duty-cycle MAC protocols reduce idle listening time, but the required synchronization introduces extra overhead and complexity, and a node may need to wake up multiple times if its neighbours are on different schedules. Asynchronous duty-cycle MAC protocols remove the synchronization overhead, which also means that they could support applications that require very low duty cycle (<0.1%) [76]. However, they are mainly optimized for light traffic loads and become less efficient in latency, power efficiency, and packet delivery ratio as traffic load increases.

2.5.2 Synchronisation in WSNs

There are several different needs for synchronisation in WSNs. First, there are many application areas where sensor nodes need to collaborate to achieve a complex sensing task. Data fusion is an example in which data sensed at different nodes is collected and aggregated into meaningful results. For example, in a vehicle tracking application, different sensor nodes report the location and time at which they sense the vehicle to a base station where different pieces of information are processed to estimate the location and velocity of the vehicle. If the sensor nodes lack a common timescale (i.e., are not synchronised), the estimate will be inaccurate.

Synchronisation is also used by energy-saving mechanisms to increase network lifetimes, such as in duty-cycle protocols, in which sensor nodes may sleep by turning off their sensors and/or transceivers at appropriate times, only waking up when necessary. The nodes need to sleep and wake up at coordinated times, so that the receiver of a sensor node is not turned off when there is data directed to it. This requires precise timing between sensor nodes within the same neighbourhood.

Scheduling algorithms such as TDMA, which enable multiple sensor nodes to share the transmission medium in the time domain to eliminate collisions and conserve energy, have a very stringent requirement on synchronisation for their operations.

Clock synchronisation has been studied thoroughly in the areas of Internet and Local Area Networks (LANs) [77][78]. Many existing synchronisation algorithms rely on the Global Positioning System (GPS) to work. However, GPS is not widely available in many WSN application areas, such as those underwater, indoors and underground. It also requires a relatively high-power receiver, which is not possible in tiny and low cost sensor nodes.

Network Time Protocol (NTP), a software-based protocol, is the default protocol used for maintaining synchronisation in computer networks due to its ubiquitous deployment, scalability, and robustness related to failures. However, NTP is not suitable for WSNs due to many challenges, such as limited energy and bandwidth, latency, dynamic topology and multi-hopping. Therefore, clock synchronisation algorithms which are different from the conventional protocols are needed to deal with the challenges specific to WSNs.

Different WSN applications have different synchronisation requirements. The requirements can be broadly classified into three categories. The first is event ordering;

many applications of sensor networks rely on the chronological order of event occurrences to obtain useful information from sensed data. For such applications, clocks are normally unsynchronised; time reference normalisation is only performed when an event of interest occurs [79]. The second is the maintenance of relative clock synchronisation. In this category, each node records the relative offset between its clock and the clocks of other nodes in the network. Relative clock synchronisation can be further divided based on the scope of synchronisation needs. For some applications, it is enough to synchronise only a subset of the network at a time (cluster-based synchronisation), whereas for others, network-wide synchronisation might be required [80]. The third category is that every node maintains a clock that is synchronised to a reference node. A global time scale throughout the network is maintained.

Time synchronisation in WSN is typically achieved by exchanging timing messages among the sensor nodes. There are broadly three approaches for time synchronisation in WSNs. They are one-way message dissemination (or unidirectional reference broadcast), sender-receiver synchronisation and receiver-receiver synchronisation and [80]–[83].

In the unidirectional reference broadcast approach [84][85], a single message broadcast carrying a reference clock signal is used to achieve local synchronisation with the participating nodes in the sender neighbourhood. Due to its simplicity, unidirectional reference broadcast approach is suitable for applications that requires high energy efficiency but less stringent synchronisation. On the other hand, both sender-receiver and receiver-receiver synchronisations use multiple message exchanges to achieve pair-wise synchronisation with high accuracy, however, to achieve this they need higher bandwidth and higher energy consumption. A comprehensive comparison and review of different synchronisation algorithms in WSNs can be found in [80].

2.5.2.1 Unidirectional Reference Broadcast

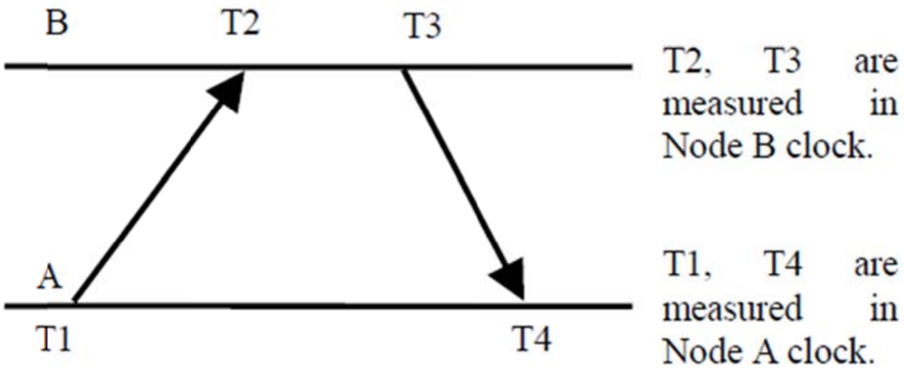
Unidirectional reference broadcast is a simple synchronisation mechanism in which a reference node simply broadcasts a reference clock signal to other nodes. The receiving nodes will then synchronise their times with the reference clock.

Flooding Time Synchronisation Protocol (FTSP) [84] is a unidirectional reference broadcast algorithm that uses low communication bandwidth and is robust against node and link failures. The goal of the FTSP is to achieve a network-wide synchronisation of the local clocks of the participating nodes. FTSP synchronises the time of a sender to multiple receivers by periodically sending reference messages that are time-stamped at both the sender and the receiver sides. FTSP uses MAC layer time-stamping to eliminate many of the errors [86] and achieve accurate time-synchronisation when the reference messages are received. In addition, linear regression is used in between synchronisation points to compensate for clock drifts.

2.5.2.2 Sender-receiver Synchronisation

Sender-receiver Synchronisation is a classic mechanism for exchanging timing information between two neighbouring nodes, and uses the timing information received to estimate the clock drift (skew) and clock offset.

The Timing-sync Protocol for Sensor Networks (TPSN) [86] works on the approach of sender-receiver synchronisation to provide network-wide time synchronisation in a WSN. The algorithm works in two phases. During the initial phase, a hierarchical structure is established in the network and in the second phase, a pair-wise synchronisation is performed along the edges of this structure to establish a global timescale throughout the network.


B
T2
T3
T2, T3 are measured in Node B clock.
A
T1
T4
T1, T4 are measured in Node A clock.

Source: *Timing-sync Protocol for Sensor Networks* [86]

Fig. 2.4 Two-way message exchange between pair of nodes *A* and *B*

As shown in **Fig. 2.4**, TPSN relies on the two-way message exchange scheme to acquire the synchronisation between two nodes *A* and *B*. Assuming that the clock drift and the propagation delay do not change in this short span of time, node *A* can calculate the clock drift θ and propagation delay *d* as:

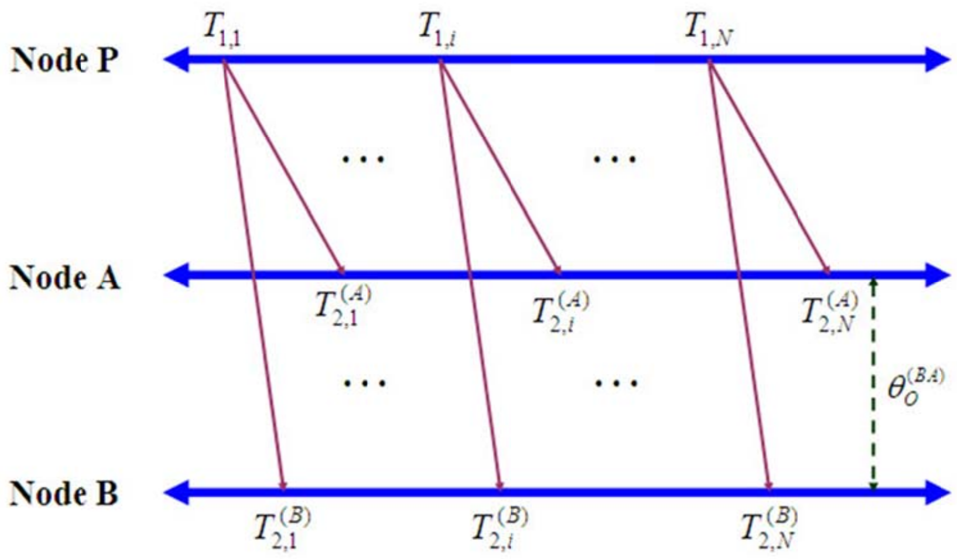
$$\theta = \frac{(T2 - T1) - (T4 - T3)}{2}, \quad d = \frac{(T2 - T1) + (T4 - T3)}{2}, \tag{2.1}$$

TPSN is scalable and the synchronisation precision does not deteriorate significantly as the size of the network increases. However, it is not energy efficient and does not support dynamic topologies since it requires a hierarchical infrastructure.

Other examples of sender-receiver synchronisation algorithms include *mini-sync* and *tiny-sync* [87], Maximum Likelihood Estimator (MLE) [88], [89], and Lightweight Time Synchronisation (LTS) [90].

2.5.2.3 Receiver-receiver Synchronisation

The main idea of receiver-receiver synchronisation algorithms is that when a node broadcasts a timing reference beacon to its neighbours, the receivers will get the message at approximately the same time. Instead of the traditional synchronisation algorithms that try to synchronise between the sender and receiver, the receivers will try to synchronise with one another.



Source: *Clock synchronisation in wireless sensor networks: An overview* [91]

Fig. 2.5 Receiver-receiver synchronisation approach where nodes *A* and *B* exchange message arrival timestamps

Reference Broadcast Synchronisation (RBS) [92] is a representative receiver-receiver synchronisation algorithm. As shown in **Fig. 2.5**, a master node *P* broadcasts a reference message with no timestamp, and the receivers (nodes *A* & *B*) stamp the arrival times of the message according to their local clocks. The receivers exchange information with each other and compute their clock offsets by averaging all the observed timestamps of neighbour nodes. Clock drift is estimated over the time by least-squares linear regression. In case of master node failure, there should be an election algorithm to re-elect the master node and this is not robust as additional computation load and convergence time are required. In terms of computational load, for a single-hop network of *n* nodes, RBS requires $O(n^2)$ message exchanges, which is computationally expensive in large networks. Similarly, the convergence time can be high due to the large number of message exchanges [91].

Other examples of receiver-receiver synchronisation algorithms include the Time-diffusion Synchronisation Protocol (TDP) [93] and Coefficient Exchange Synchronisation Protocol (CESP) [94].

Unidirectional reference broadcast such as FTSP utilizes less network resources than both RBS (receiver-receiver synchronisation) and TPSN (sender-receiver synchronisation). Assuming all three algorithms use the same synchronisation period, each node sends 1 message in FTSP, 2 messages in TPSN (1 message to parent and 1 response), and 1.5 messages in RBS (0.5 for a reference broadcast and 1 for a time-stamp exchange message) in 1 synchronisation period.

2.5.2.4 Synchronisation in synchronous duty-cycle MAC

The classic synchronous duty-cycle MAC protocol S-MAC follows a SYNC-DATA-SLEEP 3-phase operational cycle. During SYNC windows, sensor nodes broadcast synchronisation packets (*sync*) periodically to synchronise the clocks of the neighbouring nodes. During DATA windows, sensor nodes send out data packets from the higher layers based on some contention mechanisms to avoid collisions. Later developments of synchronous MAC protocols such as DW-MAC, AS-MAC and SEA-MAC focus on improving the energy efficiency, throughput and delay performance by implementing changes in the scheduling and transmission of data packets, leaving the synchronisation algorithm largely unchanged.

The synchronisation algorithm adopted by the above synchronous MAC protocols is based on fixed, periodic synchronisation packet broadcast algorithms [95] in SYNC windows. This algorithm works fine when the network is sparse. When the network is dense however, there are many unnecessary transmissions that cause collisions and consume excess energy. Energy consumption for the synchronisation process in SYNC windows is not insignificant as for most of the synchronous MAC protocols, the ratio of SYNC window to DATA window is about 1:2. Energy efficiency for the synchronisation process will be examined in Chapters 4 and 5.

2.5.2.5 Effect of clock drift on synchronisation

Every sensor node has a local clock that is based on a crystal oscillator which provides a local time for each node. The time reference in a sensor node is just a counter that gets incremented with interrupts from the oscillator. Due to the imperfections of crystal oscillators, the time maintained by each sensor node will drift away from the ideal time, as well as from one another, over time. As discussed in [81], the clock function of a node *i* can be generally modelled as

$$C_i(t) = \theta + f_i \cdot t, \tag{2.2}$$

where θ and f_i are the clock offset (phase difference) and clock drift rate (frequency difference) respectively. Therefore if two nodes A and B are initially synchronised with each other, the time difference between the two clocks in time *t* can be shown as

$$C_A(t) - C_B(t) = (f_A - f_B) \cdot t, \tag{2.3}$$

which is proportional to the time elapsed since the last synchronisation. Periodic re-synchronisation is thus required to prevent the continuing increase of clock offsets that will affect communication reliability and energy consumption efficiency.

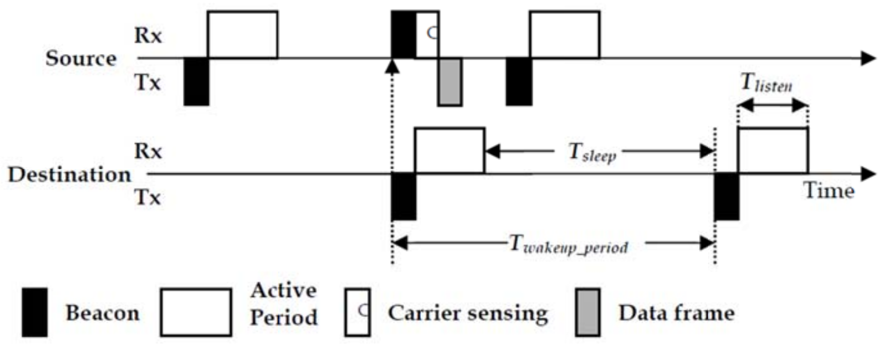
A typical crystal-quartz oscillator commonly used in sensor networks has a drift rate of up to 40 parts per million (±40 ppm) [81]. In addition, external factors such as temperature, voltage changes and hardware aging also add to the clock drift. Therefore a duty-cycle MAC protocol and its synchronisation algorithm must be able to handle different levels of clock drifts and still provide good energy and data performance.

2.5.2.6 Effect of duty cycle on synchronisation

Duty cycling is one of the key mechanisms in WSNs to reduce energy wastage in idle listening and improve network lifetime. In general, lower duty-cycle networks, with longer sleep time, have lower energy consumption albeit at the expense of longer packet delivery times. With longer sleep periods and longer frame times in low duty-cycle operations, *sync* packet inter-arrival times are longer even when the number of frames in the synchronisation period is kept constant, which makes clock synchronisation a greater challenge. For example, a 20 kbps S-MAC frame is about 8.0 seconds and 1.6 seconds in 2% and 10% duty-cycle (dc) networks respectively. With a 10-frame synchronisation period, a sensor node in a 2% dc network will schedule a *sync* packet every 80 seconds compared to just 16 seconds in a 10% dc network. Therefore in comparing different synchronisation algorithms, it is important to evaluate the stability of their performance in different duty-cycle operations.

2.5.3 Synchronous Duty-cycle MAC Protocols

Synchronous duty-cycle MAC protocols are typically designed with periodic sleep/wakeup schedules for data exchange which consists of a sleep period T_{sleep} and an active period, T_{active} repeated at T_{wakeup} intervals [96]. The typical operation of a synchronous duty-cycle MAC protocol is shown in **Fig. 2.6**, where beacon frames are transmitted frequently to achieve synchronisation. At the start of the active periods, a node broadcasts its beacon frames and shares its sleep/wakeup schedule with its neighbours. This way, all the nodes in the neighbourhood can use the same schedule for data communication.



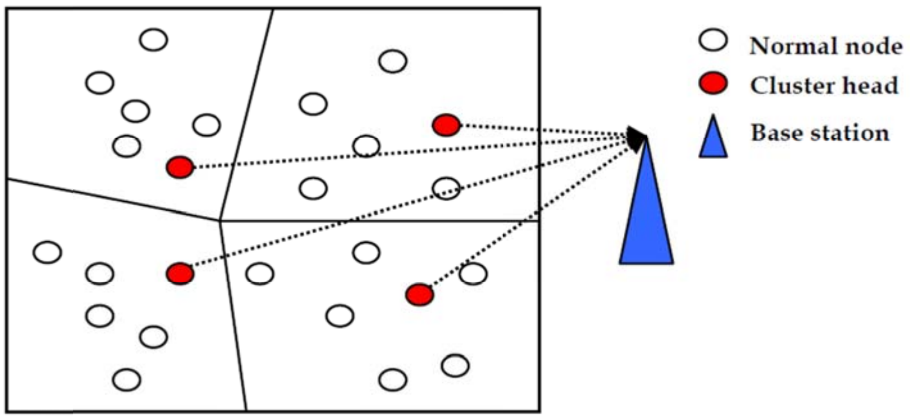
Source: A Survey of low duty cycle MAC protocols in wireless sensor networks [69]

Fig. 2.6 A synchronous duty-cycle MAC sleep/wakeup scheme

The seminal synchronous duty-cycle MAC protocol S-MAC [97] further divides the active periods into SYNC and DATA windows. During SYNC windows, sensor nodes broadcast synchronisation (*sync*) packets periodically to synchronise the clocks of the neighbouring nodes. During DATA windows, sensor nodes send out data packets from the higher layers based on a contention mechanism to avoid collisions. Later developments of synchronous MAC protocols such as DW-MAC [98], AS-MAC [99], SEA-MAC [100] and LO-MAC [101] focus on improving the energy efficiency, throughput and delay performance by implementing changes either in the scheduling and transmission of data packets, or tuning the active/sleep cycles for different data traffic behaviour and specific applications. Little attention has been given to the energy consumption of the sensor nodes in the synchronisation process.

2.5.3.1 Low-Energy Adaptive Clustering Hierarchy (LEACH)

LEACH [102] is a Time Division Multiple Access (TDMA) MAC protocol for low-energy operation. Using a TDMA approach allows end nodes to sleep for a long time outside their allocated timeslots to save energy. As shown in **Fig. 2.7**, a LEACH network is organised into clusters, each cluster consisting of a cluster head that communicates directly with all the end nodes within the cluster. The cluster head receives the data from all the end nodes, processes the data and transmits them directly to the remote base station.



Source: *A Survey of low duty cycle MAC protocols in wireless sensor networks* [69]

Fig. 2.7 LEACH hierarchical cluster network architecture

The operation of LEACH is organised into phases, consisting of a set-up phase and a steady-state phase. In the set-up phase, a distributed cluster formation technique is used to select cluster heads. The cluster heads then broadcast their services using the Carrier Sense Multiple Access (CSMA) mechanism to the end nodes. The end nodes select a cluster head based on the received signal strength. The cluster formation is complete once all the cluster members are synchronised to the TDMA schedules. Data transmission can then take place during the steady-state phase.

As a cluster head needs to be active all the time, its energy consumption is much higher than the end nodes. LEACH incorporates randomised rotation of cluster heads in order to balance energy consumption among the sensor nodes. However, the fixed clustering structure and the need for tight, global synchronisation make the network not scalable. In addition, the high transmission power required for direct communication between cluster heads and the base station may dominate the total energy consumption since every sensor node must have enough power to reach the base station if selected as a cluster head.

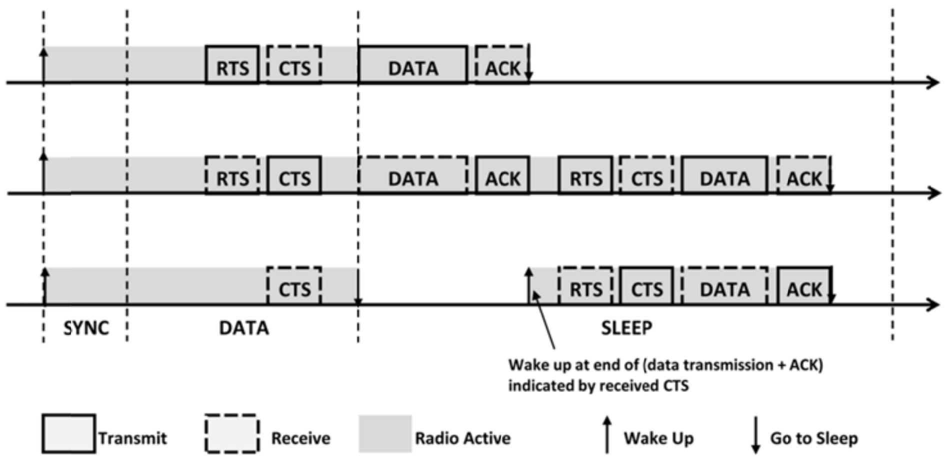
2.5.3.2 Sensor MAC (S-MAC)

S-MAC [97][95] is a synchronous duty-cycle MAC protocol primarily designed for energy conservation and self-configuration. Unlike LEACH, S-MAC adopts a virtual-cluster approach to support a flat, multi-hop network topology. Neighbouring nodes form virtual clusters based on common sleep/wakeup schedules to reduce latency and control overhead.

S-MAC introduces several novel features for energy-efficient operation. The first feature is a periodic sleep and listen schedule. In the listen period, sensor nodes wake up to listen and communicate with other nodes. The listen period is further divided into SYNC and DATA periods. Only synchronisation frames are allowed in SYNC periods for the purpose of synchronising the neighbourhood, and Data frames follow a contention procedure to access the media during the DATA periods. In a sleep period, the nodes will try to sleep by turning off their radios. In this way, the time spent on idle listening can be significantly reduced.

S-MAC is a contention-based protocol. To avoid collisions, S-MAC has adopted both physical and virtual carrier sensing, which is similar to the Distributed Coordination Function (DCF) protocol in IEEE 802.11 standards. The sequence of RTS/CTS/DATA/ACK is used to avoid collisions due to hidden nodes.

The periodic sleep and listen scheme, however, increases latency in multi-hop networks. S-MAC implements an adaptive-listening technique [97] to reduce the latency that could be caused by the periodic sleep of intermediate nodes.



Source: *A survey and projection on medium access control protocols for wireless sensor networks* [67]

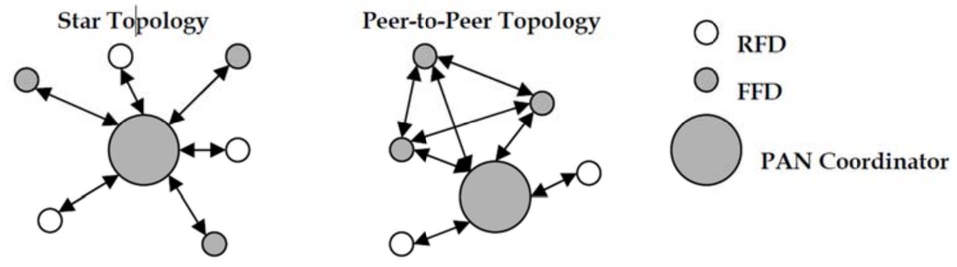
Fig. 2.8 Adaptive listening in S-MAC

To reduce energy wastage due to overhearing, S-MAC allows all interfering nodes, which are immediate neighbours of the sender and receiver, go to sleep after they "hear" an RTS or CTS packet (**Fig. 2.8**).

In large multi-hop networks, multiple virtual clusters are formed with multiple schedules. Nodes that share the same schedule form a virtual cluster in S-MAC, and nodes with neighbours in two or more clusters are border nodes. Border nodes consume more energy as they spend more time listening to or sending data. Therefore, these border nodes will be exhausted of energy sooner than the other nodes, which could possibly cause network partitioning. The Global Schedule Algorithm (GSA) [103] was introduced to eliminate inefficient multiple-schedules at border nodes and improve the network lifetime of S-MAC. GSA uses schedule age as a unique parameter for all nodes to converge to the global schedule.

2.5.3.3 IEEE 802.15.4 MAC

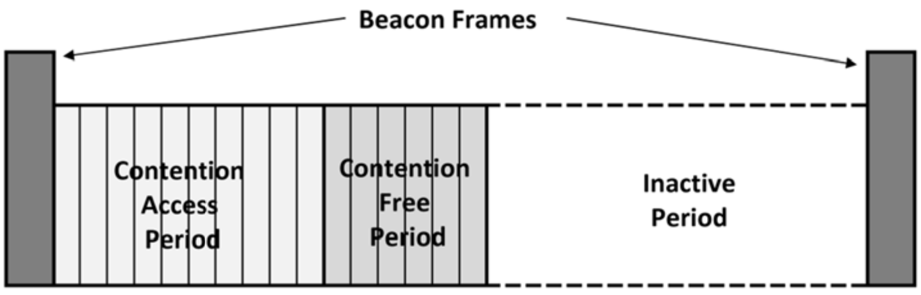
The IEEE 802.15.4 MAC standard was released in 2006 to support Wireless Personal Area Networks (WPANs) with a duty-cycle mechanism in which the size of active and sleep periods are adjustable during the WPAN formation. The standard defines two types of devices: a Full-Function device (FFD) and a Reduced-Function device (RFD). FFD can play the role of a coordinator whereas RFD can only form star topologies by connecting to the network coordinator [104][105]. Multiple coordinators can either operate in a peer-to-peer topology or a star topology with a coordinator which later becomes the PAN coordinator as shown in **Fig. 2.9**.



Source: A Survey of low duty cycle MAC protocols in wireless sensor networks [69]

Fig. 2.9 Topology configurations formed by FFD and RFD in IEEE 802.15.4 standard

The key features of IEEE 802.15.4 are in the beacon mode where a superframe structure is maintained to organise the channel access and data exchanges, which is shown in **Fig. 2.10**. Typically the coordinator broadcasts a beacon frame in the first time slot which is then followed a Contention Access Period (CAP) and a Contention-Free Period (CFP). CFP is made up of Guaranteed Time Slots (GTS) allocated by the coordinator to allow specified end nodes to transmit data without contention. A node therefore listens to the beacon first to determine whether a GTS has been reserved for itself. If there has, it remains powered off until its scheduled GTS to transmit the data. If no GTS is reserved, it will use CSMA/CA during the CAP with typical back-off procedures for its data transmission.



Source: *Overview of the IEEE 802.15. 4 standards family for low rate wireless personal area networks* [105]

Fig. 2.10 Superframe structure in IEEE 802.15.4 beacon mode

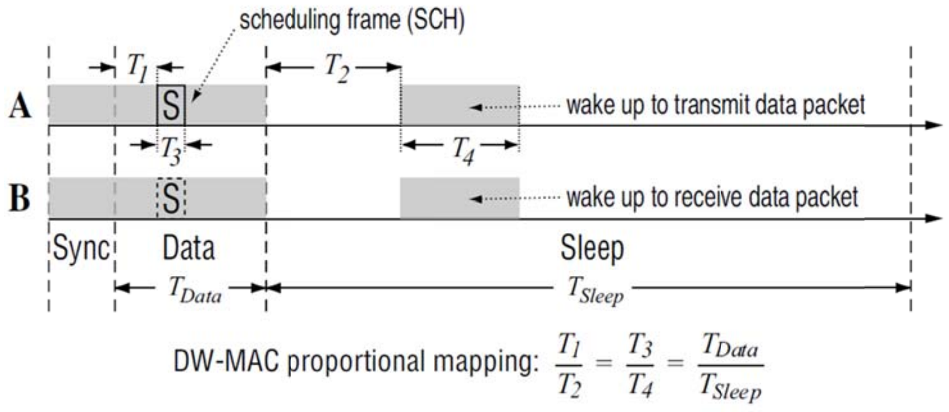
However, although energy efficiency is achieved for end nodes by putting the nodes to sleep during inactive periods and when there is no data to transmit or receive, the energy burden is put on the coordinator where it has to be active during the entire active period. Also, as with other scheduled protocols, the key issue is the synchronisation of the entire network as well as the maintenance of this synchronisation.

2.5.3.4 Demand Wakeup MAC (DW-MAC)

DW-MAC [98] is an enhancement of S-MAC that improves both energy efficiency and latency performance. It introduces a new low-overhead scheduling algorithm that allows nodes to wake up on demand during the sleep period of an operational cycle, ensuring that data transmissions do not collide at their intended receivers. This demand wakeup technique adaptively increases effective channel capacity as the traffic load increases, as transmission is now made possible during sleep periods.

In DW-MAC, medium access control and data scheduling are integrated. A node with data to transmit during a DATA period broadcasts a special frame called a *scheduling frame* (SCH). SCH is a replacement of RTS/CTS and it sets up one-to-one mapping between a DATA period and the following Sleep period, which uniquely reserves the proportional interval of time for the transmission of the pending data frame in the following Sleep period (**Fig. 2.11**). As in RTS, SCH contains the destination address, so that only the intended receiver will wake up during the specified time, minimising the energy consumed due to unnecessary wake-ups. From the simulations performed in

[98], DW-MAC is 50% and 15% more efficient in energy performance for unicast and broadcast traffic respectively.


scheduling frame (SCH)
T_1
T_2
A
S
wake up to transmit data packet
T_3
T_4
B
S
wake up to receive data packet
Sync
Data
Sleep
T_{Data}
T_{Sleep}
DW-MAC proportional mapping: $\frac{T_1}{T_2} = \frac{T_3}{T_4} = \frac{T_{Data}}{T_{Sleep}}$

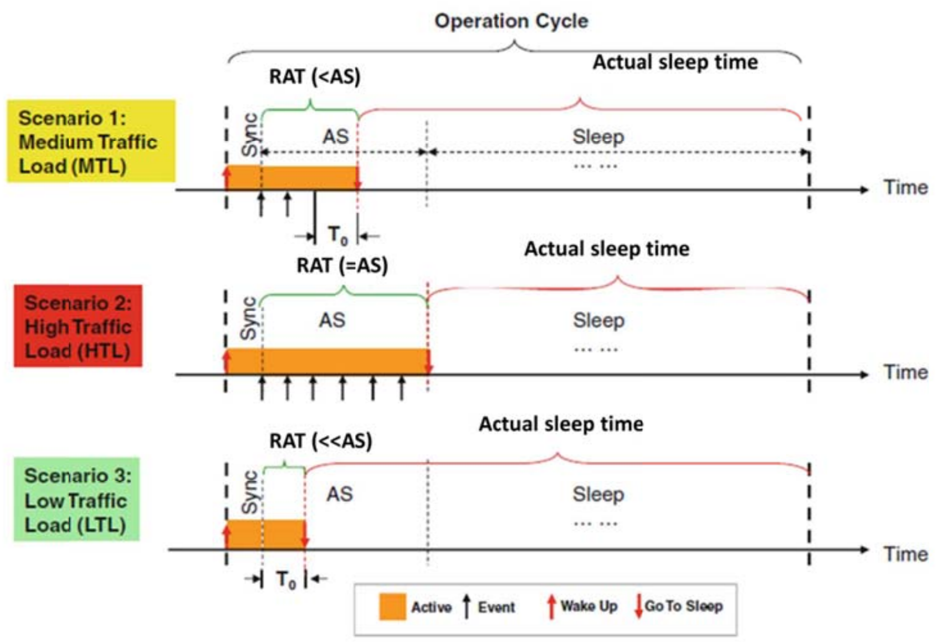
Source: *DW-MAC: a low latency, energy efficient demand-wakeup mac protocol for wireless sensor networks* [98]

Fig. 2.11 Overview of data frame scheduling by SCH in DW-MAC

2.5.3.5 Adaptive Scheduling MAC (AS-MAC)

AS-MAC [99] is an evolution of DW-MAC that introduces an adaptive scheduling mechanism to make the protocol adaptive to varying traffic load. Similar to DW-MAC, AS-MAC allows sensor nodes to wake up on demand during the sleep period to transmit or receive packets. However, a major disadvantage of DW-MAC is that the durations of the DATA period and Sleep period are fixed, and therefore the system is not able to dynamically adapt to different traffic loads.

AS-MAC replaces the DATA period with Adaptive Scheduling (AS) period. Within the AS period, the time duration for a node to stay awake is defined as the Resilient Active Time (RAT). The length of the RAT can vary in each operational cycle and is adaptive to the variable traffic load. This mechanism allows nodes to schedule more data transmissions within one operational cycle when the traffic load is high, as well as go to sleep earlier when the traffic load is low. **Fig. 2.12** illustrates the adaptive change of the RAT with respect to the change in traffic conditions in three operating cycles.



Source: *An Energy-Efficient Self-Adaptive Duty Cycle MAC Protocol for Traffic-Dynamic Wireless Sensor Networks* [100]

Fig. 2.12 Adaptive change of resilient active time (RAT) for: Scenario 1 (average traffic), Scenario 2 (high traffic), and Scenario 3 (low traffic)

2.5.3.6 Self-Adaptive Duty Cycle MAC (SEA-MAC)

The duty-cycle MAC such as S-MAC, DW-MAC, and AS-MAC all adopt fixed duty-cycles, which are unable to adapt to variations in traffic. A low duty-cycle operation is designated for low traffic load to conserve energy, but it results in high end-to-end delay and low packet delivery ratio (PDR) for high traffic loads. On the other hand, a high duty cycle operation might help in reducing end-to-end delay and increasing PDR, but it will result in significant energy wastage when traffic is low.

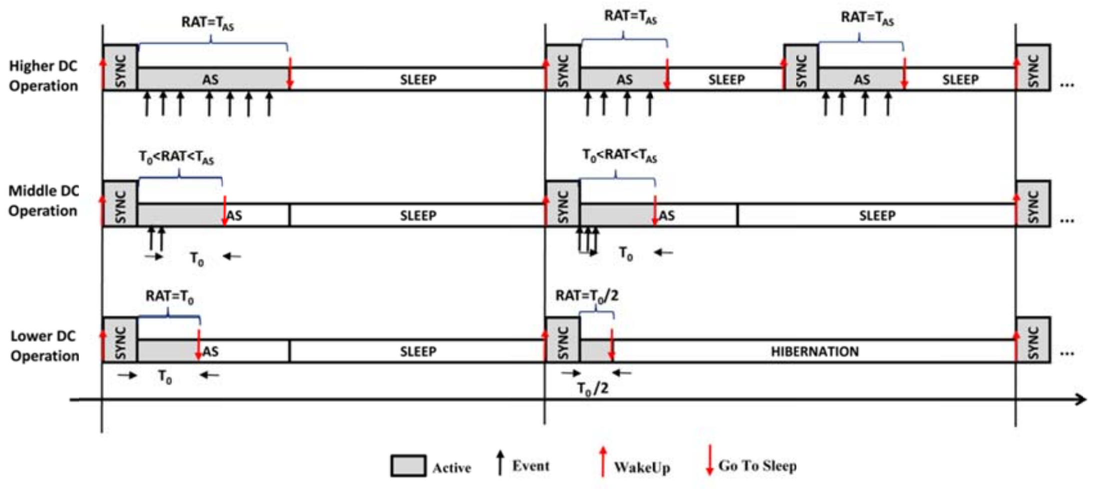
SEA-MAC [100] is similar to AS-MAC in the following aspects:

- It follows a SYNC-AS-SLEEP 3-phase operational cycle.
- It adopts the timeout-based Resilient Active Time (RAT) mechanism.
- It inherits the enhanced proportional mapping function for scheduling data transmission.

- It retains the SCH frame with tracked retry numbers for efficient multi-hop forwarding.

Different from AS-MAC, SEA-MAC computes the duty-cycle efficiency of the preceding operation cycle in SYNC periods, and shares it with the neighbouring sensor nodes. The duty-cycle of the succeeding operation cycle is then adaptively determined based on the computed results. SEA-MAC defines duty-cycle efficiency as the ratio of RAT over T_{AS} since this ratio reflects the effective duty cycle based on the proportional mapping function for data transmissions.

As shown in **Fig. 2.13**, if the duty-cycle efficiency has reached the pre-determined lowest efficiency, the succeeding operation cycle for this node will be reduced in order to comparatively increase the duty-cycle efficiency. The opposite is true if the duty-cycle efficiency has reached the pre-determined highest efficiency. Otherwise the operation cycle will remain unchanged.



Source: *An Energy-Efficient Self-Adaptive Duty Cycle MAC Protocol for Traffic-Dynamic Wireless Sensor Networks* [100]

Fig. 2.13 Dynamic changes of duty cycles in SEA-MAC for three different duty-cycle efficiencies

Experimental results from [100] show that SEA-MAC outperforms AS-MAC substantially in energy consumption and end-to-end delay, especially under heavy unicast traffic load scenarios.

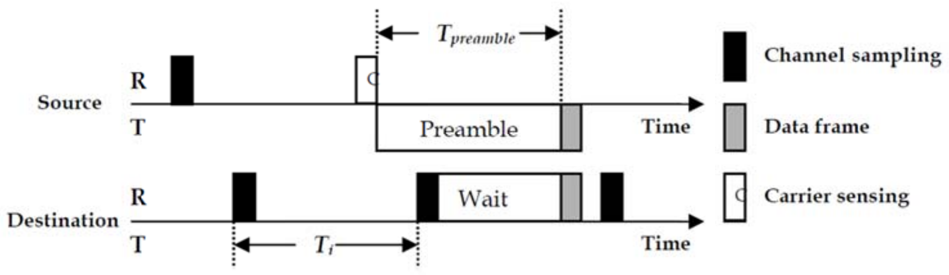
2.5.4 Asynchronous Duty-cycle MAC Protocols

Unlike the synchronous protocols, asynchronous duty-cycle MAC protocols do not require prior knowledge on the timing information and schedules of neighbouring nodes for data communication. They are not impacted by clock drift and hence do not require a synchronisation protocol to operate.

Asynchronous duty-cycle MAC utilizes a frequent channel sampling mechanism known as low power listening (LPL) to detect possible starting transmissions in the network. The sender first transmits a preamble packet to signal that there is data to be transmitted. Upon receiving the preamble packet, the receivers wake up to wait for the arrival of data packets. However, the long preamble packet size of this transmitter-initiated approach in asynchronous WSNs contributes to the higher transmission energy used in the network. Other approaches such as receiver-initiated and redundant transmission of preamble packets are explored to reduce the burden on the transmitter [72]. In addition, frequent channel sampling also contributes to higher start-up costs; proper measures must be taken to ensure the optimal wake-up period is implemented.

2.5.4.1 Berkeley MAC (B-MAC)

B-MAC [73] is a variant of CSMA with a preamble sampling mechanism. To achieve a low power operation, B-MAC employs an adaptive preamble sampling scheme to reduce the duty cycle and minimise idle listening. Upon waking up, the sensor nodes use Low Power Listening (LPL) to check for activity above the estimated noise floor. The nodes will go into a full active state and wait for data packets only if activity is detected. If it is a false-positive and no packet is received, the nodes will go back to sleep after a timer time-out. The basic operation of B-MAC is shown in **Fig. 2.14**.



Source: *A Survey of low duty cycle MAC protocols in wireless sensor networks* [69]

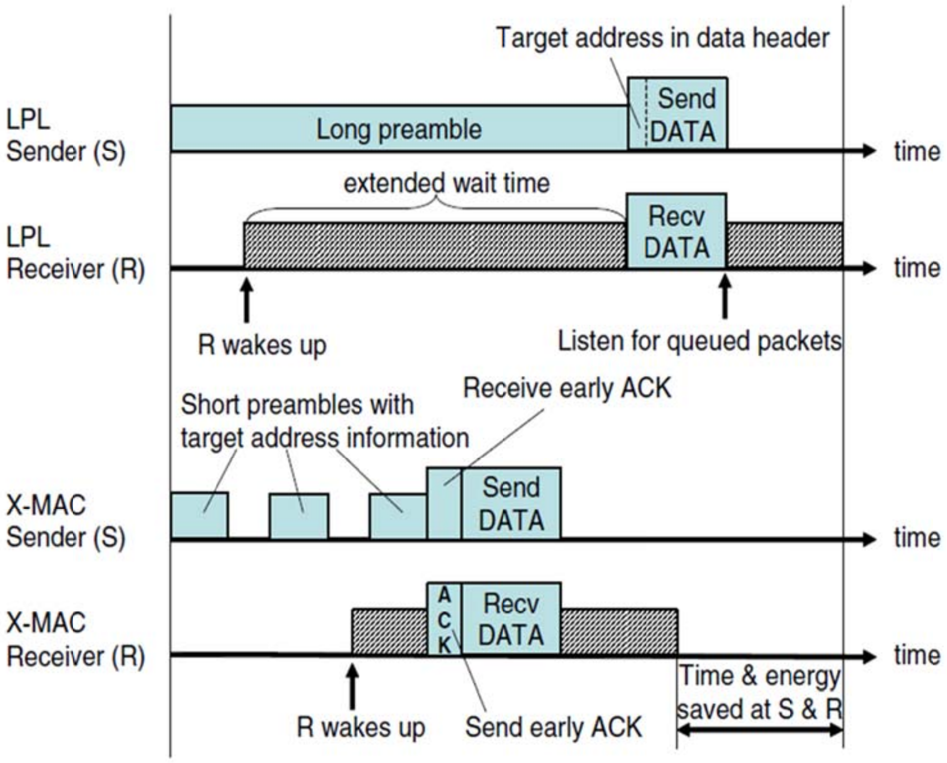
Fig. 2.14 Basic operation of B-MAC with preamble sampling

The CSMA mechanism is implemented at the transmitter. Upon detection of a clear channel, a long preamble is transmitted preceding a data packet. Acknowledgement can be used to enhance reliability. RTS-CTS can also be implemented in high traffic networks to reduce collisions.

B-MAC provides some flexibility to the higher layer protocols to adjust channel sampling interval T_i, based on the changing network conditions. In a high load condition, sampling rates can be increased to increase the traffic capacity of the network; however, this will also increase the energy consumption of the sensor nodes.

2.5.4.2 X-MAC

The long preamble in B-MAC introduces excess latency at each hop and suffers from excess energy consumption at non-target receivers. X-MAC [74] solves the long preamble problem by dividing it into a stream of short preamble packets, each containing the ID of the target node as shown in **Fig. 2.15**. The series of short preamble packets effectively constitutes a single long preamble. When a node wakes up and receives a short preamble packet, it inspects the target ID. If it is the intended recipient, it stays awake to receive the subsequent data packet; otherwise, it goes back to sleep, thus avoiding the overhearing problem.



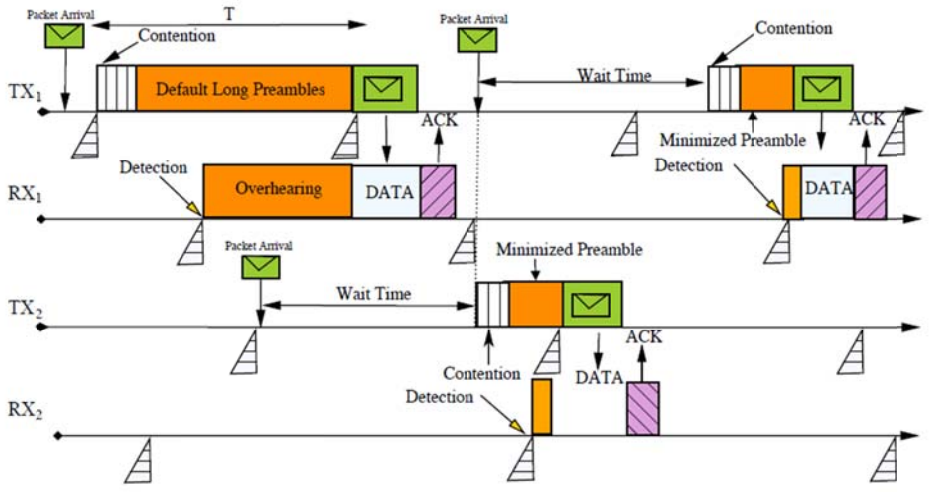
Source: *X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks* [72]

Fig. 2.15 The basic operations of X-MAC's short preamble approach compared to LPL approach

The second improvement of X-MAC is the introduction of early acknowledgment packets by the intended receiver to the sender through the gap between the short preambles. When the sender receives the acknowledgement packet, it stops sending preambles and sends the data packet instead. This reduces per-hop latency and avoids unnecessary energy consumption in waiting and transmitting.

2.5.4.3 WiseMAC

WiseMAC [75] is a preamble sampling duty-cycle MAC protocol. It solves the problem of the long preamble by learning the sampling schedule of its neighbours so that a wake-up preamble of minimised length can be used.



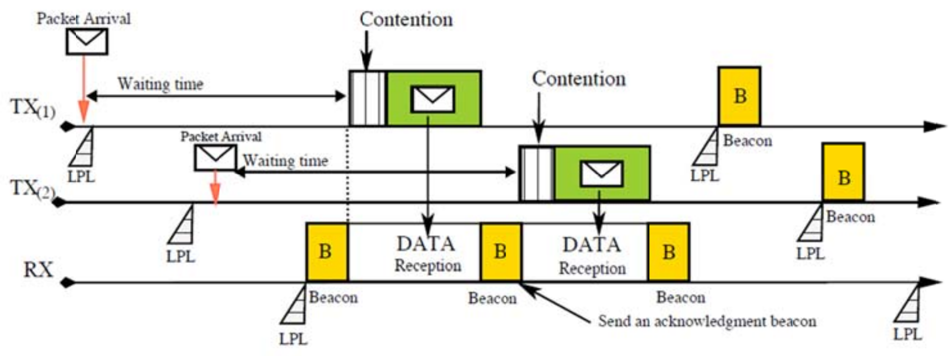
Source: *Survey on latency issues of asynchronous mac protocols in delay-sensitive wireless sensor networks* [74]

Fig. 2.16 Adaptive wake-up preamble mechanism through learning of sampling schedule in WiseMAC

As shown in **Fig. 2.16**, when the receiver's wakeup pattern is still unknown, *TX$_1$* sends a preamble that is equal to the full basic cycle duration *T*. Each receiver adds its own schedule into the ACK frame when a DATA frame is successfully received. Sampling schedule offsets of all neighbouring nodes are subsequently learnt and kept in a table. The schedules are dynamically updated whenever frames and schedules are exchanged or possibly overheard. Based on the schedule table, a node can determine the wakeup times of its neighbours so that it can wake up at the right time to send data with minimum preamble length. This feature also allows *TX$_2$* to send its data quickly without additional waiting time as shown in **Fig. 2.16**.

2.5.4.4 Receiver-Initiated MAC (RI-MAC)

RI-MAC [106] aims to minimise the time a sender and its intended receiver occupy the wireless medium by using receiver-initiated data transmission. In contrast to B-MAC and X-MAC, there is no preamble transmission from senders in RI-MAC. In RI-MAC, each node wakes up periodically based on its own schedule (determined by its duty-cycle) and broadcasts a short beacon frame if the medium is idle. A node with queued data remains active and waits silently until it receives a beacon frame sent by the intended receiver. As only beacons and data transmissions occupy the medium in RI-MAC with no preambles, occupancy of the medium is significantly decreased, which frees up more capacity for data exchange among other nodes.



Source: *Survey on latency issues of asynchronous mac protocols in delay-sensitive wireless sensor networks* [74]

Fig. 2.17 The basic operations of receiver initiated beacon broadcast and contentions among the senders in RI-MAC

As shown in **Fig. 2.17**, node $TX_{(1)}$ starts DATA transmission upon receiving a beacon from *RX*, which is acknowledged by *RX* with another beacon. This second beacon has two functions; first, it acknowledges the correct receipt of the sent DATA frame; and second, it invites a new DATA frame transmission. $TX_{(2)}$ then has the opportunity to send its DATA frame to the same receiver.

2.6 Data-centric Protocols

Data-centric protocols are typically used in data gathering sensor networks. In data gathering applications, the sink requests data by sending queries to the sensor nodes near a region of interest. In response, the sensor nodes in the region then collect and transmit the data back to the sink. Due to the large number of sensor nodes in a WSN and their random positions, there is no global identification of the nodes; making it difficult in querying a particular set of sensors. The traditional flooding approach of data dissemination also leads to implosion and data redundancy, which is bandwidth and energy inefficient. In addition, sensor nodes often cover overlapping geographic areas which leads to overlapping data sending to the sink [33][107].

2.6.1 Sensor Protocol for Information via Negotiation (SPIN)

SPIN [107] is a negotiation-based information dissemination protocol suitable for WSNs. Negotiation and resource-adaptation are two key mechanisms used in SPIN that contribute to its energy efficiency.

To overcome the problems of implosion, SPIN nodes negotiate with the neighbouring nodes to eliminate the transmission of redundant data messages. To eliminate overlap, SPIN uses meta-data as the descriptors of the data for negotiation, allowing the nodes to name the portion of the data that they are interested in obtaining. In this way, nodes do not waste energy for the unnecessary data transmission.

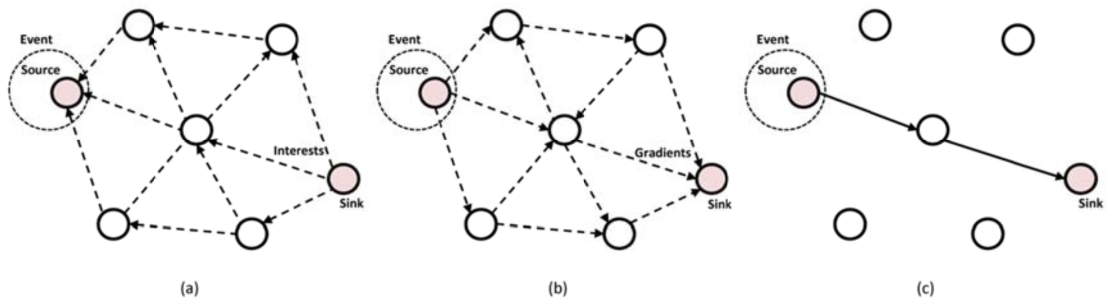
The SPIN protocols are resource aware and resource adaptive. Each sensor node has its own resource manager to compute the energy required to process, send, and receive data over the network. The resource manager also keeps track of the energy consumption, which helps the sensors to monitor and adapt to any change in their own resources. Whenever their resources are low, nodes are able to cut back on their activities to increase their lifespan.

However, SPIN does not specify a format for meta-data; therefore, SPIN applications must define a meta-data format that takes into account the costs of storing, retrieving, and managing the data in order to be effective. The cross-layer dependency of SPIN also makes it less flexible to support different WSN implementations.

2.6.2 Directed Diffusion

Directed Diffusion [108] is a data-centric protocol designed for sensor query, data dissemination and processing. The protocol uses attribute-value pairs for naming the data and diffuses the named data through sensor nodes. The main reason behind using such a scheme is to get rid of unnecessary operations of network layer routing in order to save energy.

A sink node using directed diffusion creates a query by broadcasting an *interest* message with a list of attribute-value pairs. The dissemination of *interest* message sets up *gradients* along multiple paths within the network. Specifically, the gradient direction is set towards the neighbouring node that sends the *interest*. Initially, when a source detects a matching target, it sends exploratory responses along multiple paths towards the sink. When the sink receives these exploratory responses, it reinforces one particular neighbour as a preferred path based on some local rules. For example, choosing the neighbour from which the most events have been received, or the neighbour which consistently sends events before other neighbours. This reinforcement process repeats towards the source and a reinforced, preferred path is established. **Fig. 2.18** illustrates a simplified schematic of directed diffusion.



Source: *Directed Diffusion for Wireless Sensor Networking* [108]

Fig. 2.18 Simplified directed diffusion process. (a) Interest dissemination, (b) Initial gradients setup, (c) Data transmission along reinforced path

There are several key differences between directed diffusion and traditional networking: First, it is data-centric; sensor nodes use *interest* messages to specify required data. Second, in contrast with the end-to-end communication in traditional WSN, all communication in diffusion based networks is neighbour-to-neighbour. Third, every node can cache, aggregate, and process messages.

As the diffusion technique uses strictly local communication, none of the nodes have a global topology view, and thus the resulting communication paths may be suboptimal. However, the energy inefficiency due to suboptimal paths is compensated with data aggregation techniques and an increase in robustness.

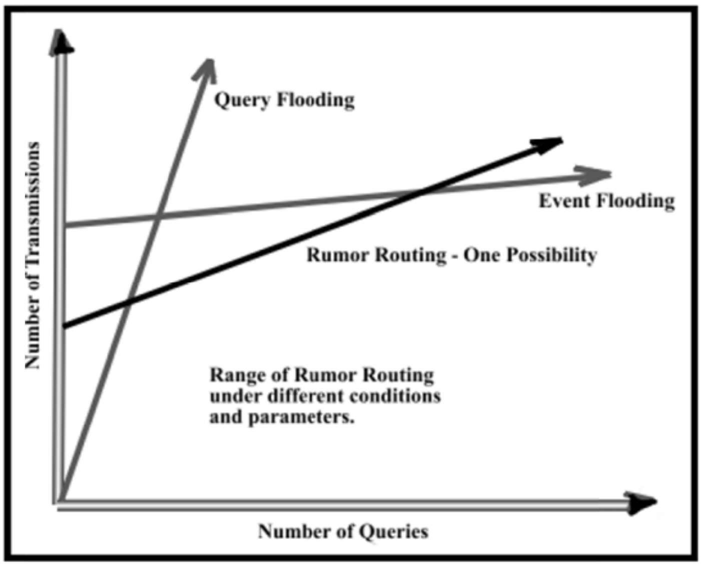
2.6.3 Rumour Routing

Directed diffusion generally floods the query throughout the entire network in order to discover the best path. However, in many applications, the queries could be for a small amount of data, and therefore flooding the queries is not efficient. Rumour Routing

[109] provides an alternative to flood the events instead of queries if the number of events is small and the number of queries is large.

The rumour routing algorithm employs long-lived packets, termed as agents, for flooding events through the network. When a node detects an event, it probabilistically generates an agent which travels through the network, propagating the event information to distant nodes.

Any node may generate a query, which should be routed to a particular event. If it has a route to the event, the query will be routed to the event node. Otherwise, the query will be forwarded in a random direction until it reaches a node that has observed the target event, or until it expires. If the query did not reach a destination; the query node can either retransmit or flood the query.



Source: Rumour routing algorithm for sensor networks [109]

Fig. 2.19 Rumour routing applies in the region below query and event flooding

As shown in **Fig. 2.19**, the Rumour Routing algorithm is intended to fill the region between query flooding and event flooding. It is useful only if the number of queries compared to the number of events lies between the two intersection points. An application with the knowledge of this ratio can use a hybrid of Rumour Routing and flooding for optimal energy utilisation.

2.7 Summary

The market potential and rapid development of IoT applications has generated strong research interests in the related component areas such as IoT communication protocols, LPWANs and WSNs, and energy efficiency is always a key consideration in the design of these components. Within the area of WSNs, node to node communications contributes to a significant amount of sensor node energy consumption. To minimise energy consumption and maximise network lifetime, various techniques are used in the design of sensor nodes which include energy harvesting, data centric protocols, routing algorithms and duty-cycle operations. In this chapter, we have reviewed and discussed the various state of the art ad-hoc routing and duty-cycle MAC protocols in detail. We have identified the gaps in the areas of energy monitoring and distribution mechanism in routing and synchronisation algorithm in duty-cycle MAC and will present our proposals in the next few chapters.

Chapter 3

Energy-balanced Routing Algorithm in WSN

3.1 Introduction

Network lifetime is one the most important metrics for the evaluation of WSNs. In a resource-constrained environment, the consumption of every limited resource must be taken into consideration. This is especially so in WSNs, where other metrics such as packet delivery ratio (PDR), quality of service (QoS), as well as sensor coverage and connectivity, are strongly dependent on network lifetime [110]. Network lifetime, in turn, depends on the lifetimes of the individual nodes that constitute the network, which is finally reduced to how much energy it consumes over time, and how much energy is available for its use.

Traditional ad hoc routing protocols such as DSDV, AODV and DSR use path cost, which considers parameters such as distance, bandwidth, and link quality, as the metric for route selection. This selection criterion does not take into account the energy cost for utilizing a particular link. As sensor devices are energy-constrained, energy consumption has become a key consideration and these routing protocols are deemed to be inadequate.

To improve energy performance, various energy-aware routing protocols that incorporate energy cost for individual links as described in Chapter 2 have been proposed [49]–[51]. Energy efficiency in these variations of energy-aware routing protocols has improved. However, the static behaviour of the path selection means that wireless nodes along the selected path have to work harder and consume more energy than those that are not on the selected path. The energies of these overworked nodes will naturally be depleted much faster than the other nodes. When this happens, there is a high possibility that the network will be partitioned, making some destination nodes unreachable.

There is another class of routing algorithms that aims to maximize network lifetime as proposed in [56]–[62]. Most of these maximum lifetime routing algorithms require accurate residual power information of all nodes in the network for route selection. However, the communication of such information could generate substantial overhead, which consumes additional energy and is not addressed in these algorithms.

In recent years, Internet of Things (IoT) application scenarios are rapidly gaining traction. The majority of these application scenarios consist of interconnected heterogeneous devices such as wireless sensors, smart-phones, as well as network-enabled embedded systems such as controllers, actuators and RFID devices. The heterogeneity of WSN nodes further increases the complexity of optimising the network lifetime of a WSN.

In this chapter, we describe a routing protocol that is computationally efficient, fully distributed, with energy-aware multi-path balancing for heterogeneous WSNs. The new routing protocol enhances the existing stable Dynamic Source Routing (DSR) protocol with an energy-balancing feature to improve network lifetime which we refer to as Energy-balanced Dynamic Source Routing (EB-DSR).

3.2 DSR Overview

DSR [42][111] is an efficient routing protocol designed specifically for multi-hop routing in a wireless ad hoc network. A wireless ad hoc network is built spontaneously when a collection of wireless mobile hosts connects to one another for data communications without the aid of any established network infrastructure or centralised administration.

DSR is a reactive routing protocol, which means routing activities are only initiated in the presence of data packets in need of a route. The key benefit of reactive or on-demand protocols is the reduction of routing overhead and energy consumption. For sensor network applications, high routing overhead consumes additional network capacity and energy, which could significantly impact the data performance and lifetime of the WSN.

DSR uses *source routing*, i.e. the source node determines the complete route to the destination when a new data packet is generated. It places the hop-by-hop information in the packet header so that the intermediate nodes can simply forward the packet based on the routing information in the packet header. Normally, the source node obtains the routing information by searching for routes it previously learned from its *route cache*. If no route is found in its cache, a *route discovery* process is initiated to find a new route to the destination node.

While a host is using any source route, it continues to monitor the correctness of the route. If any node along the route moves out of transmission range of its next or previous hop neighbour along the route, the route can no longer be used to reach the destination and is considered invalid. A route will also become invalid if any of the nodes along the route is powered off or fails due to other reasons. A *route maintenance* process is used to monitor of the validity of a route.

3.2.1 Route Discovery Process

To initiate route discovery, a source node *S* broadcasts a Route Request (RREQ) packet, which is received by all the neighbouring nodes within wireless transmission range of *S*. Each RREQ identifies the source node *S* and the destination node *D*, and also contains a unique request identification (ID). When an intermediate node receives the RREQ for the first time, it rebroadcasts the RREQ packet after adding its address to the source route. The intermediate node discards the RREQ if the message contains the same ID that it has received before, or if its address is already in the source route of the message. This process continues until the RREQ reaches the destination node *D*.

When the destination node *D* receives the RREQ, it will simply reverse the sequence of hops in the route record and use this as the source route for the Route Reply (RREP) message back to node *S* if the MAC protocol requires bidirectional frame exchange for unicast packets. Otherwise, it will examine its own *route cache* for a route back to *S*. It will use this route as the source route for delivery of the Route Reply (RREP) message back to node *S*. If no such route is found, it will piggyback the RREP on its own RREQ to discover a route to *S*. To improve the efficiency of the route discovery process, all intermediate nodes will also cache the routes learned during the process for future use. The building of source route record during route discovery process is shown as **Fig. 3.1**.

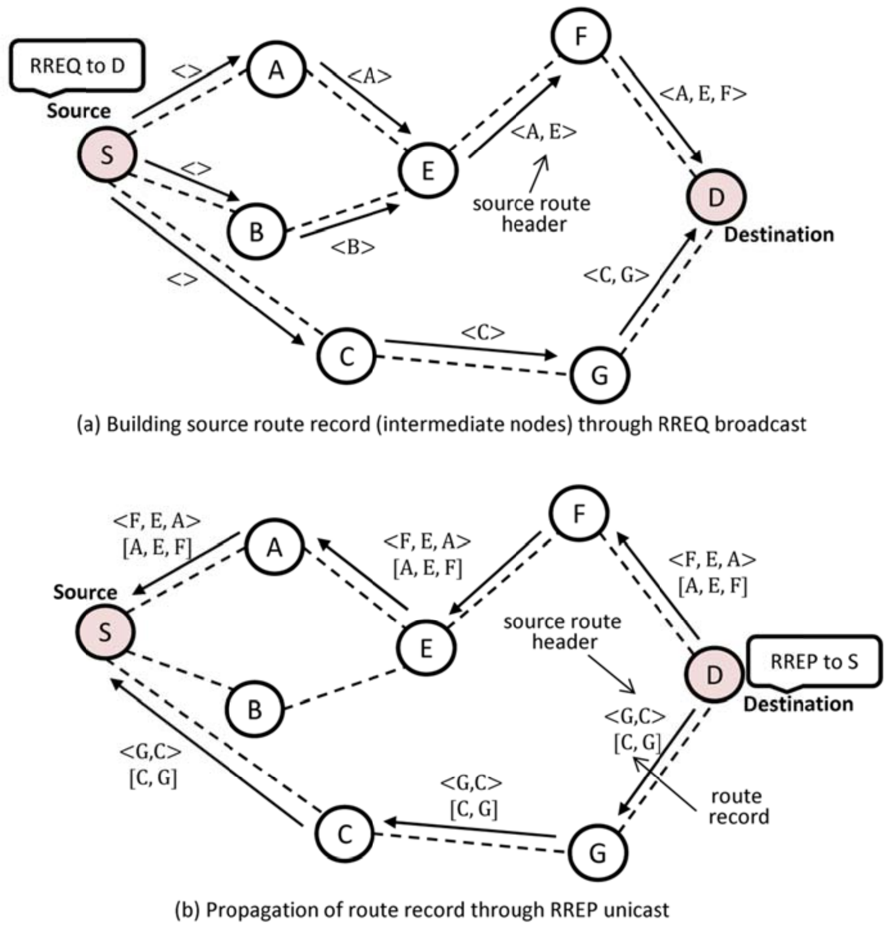


Fig. 3.1 DSR Route discovery process through broadcast of RREQ and unicast of RREP packets

3.2.2 Route Maintenance Process

The routing information for all nodes in the network needs to be updated at regular intervals. Proactive routing protocols integrate route discovery with route maintenance by continuously sending periodic routing updates. If the status of a link changes, the periodic updates will eventually reflect the changes to all other routing nodes.

In the absence of periodic updates, DSR provides two mechanisms for maintaining the correctness of the routes. The first is the acknowledgement, which provides confirmation that a link is capable of carrying data. In wireless networks, acknowledgements are often provided at no cost to the Network layer, either through the lower layer wireless MAC protocol acknowledgement, or by a "passive acknowledgement" by overhearing downstream nodes forwarding their transmitted packets.

The second is the Route Error (RERR) packet. When a node determines that a link is broken due to the lack of acknowledgement received after a maximum number of retransmissions, it returns a RERR to the original sender of the packet encountering the error as shown in **Fig. 3.2**. The RERR packet contains the addresses of the nodes at both ends of the hop in error. When a node receives the RERR, it will remove all routes that contain the broken link from its route cache. A new route discovery process is then initiated by the node to obtain new and updated routes.

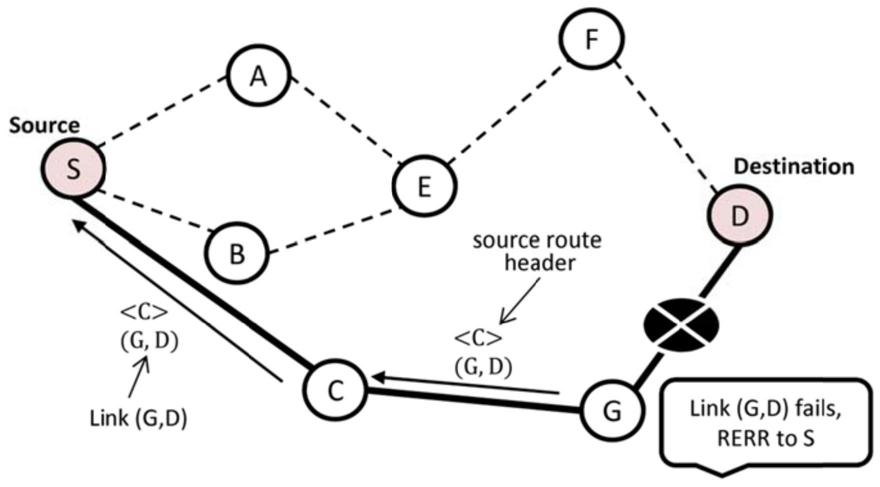


Fig. 3.2 Propagation of RERR to the source node in the event of link failure

3.3 Proposed EB-DSR Protocol

In many large-scale applications, a single WSN may consist of multiple types of sensor nodes with different sensing functions working together. In such a heterogeneous network, sensor nodes could have different energy capacities. Special care needs to be taken to avoid overloading the weaker sensor nodes; otherwise the weaker nodes could be exhausted of their energies in a very short period of time, which shortens network lifetime unnecessarily. A new routing protocol that is aware of the energy status of the sensor nodes in the WSN and a new routing metric that uses the updated node energy information is needed to improve network lifetime performance.

The proposed EB-DSR protocol has the following characteristics:

- It is reactive; a route discovery process is initiated by the source node only when there is data to be routed and no existing route is found in its route cache.
- The forwarding path is determined at the source (i.e. Source Routing Protocol).
- Within a single flow between the same source and destination pair, forwarding paths can vary from time to time dynamically based on the residual energies of the intermediate nodes along the paths.
- Multi-path routing results in balancing node remedial energies and hence extends the network lifetime.

EB-DSR protocol uses DSR as a base for source routing and adds 3 new features and enhancements for multi-path energy-aware routing.

3.3.1 New Energy-balanced Metric

One key feature of EB-DSR is the introduction of a new energy-balancing metric *M*. The new metric incorporates the traditional link cost elements (distance, bandwidth, QoS, etc.) as well as the residual node energies along a given path. In this section, we discuss the derivation and the properties of this new metric.

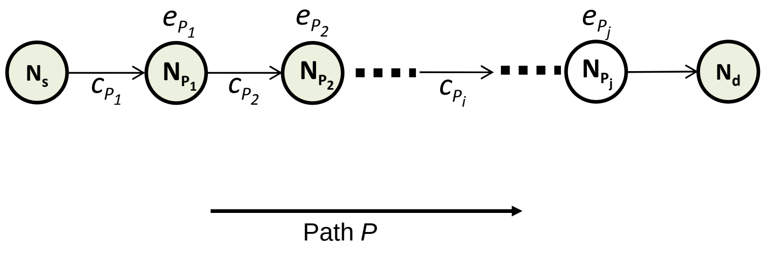


Fig. 3.3 Cost and remedial node energy information along a network path *P*.

With reference to **Fig. 3.3**, for any path *P* from a source to a destination, the path-cost metric is defined as:

$$C_P = \sum_{\forall i} (c_{P_i}), \tag{3.1}$$

where c_{P_i} denotes the cost of link *i* along path *P*. Using only the path-cost metric, between two paths *P* and *Q*, path *P* is preferred to path *Q* if:

$$C_p < C_Q \text{ , or } {}^{C_p}\!\Big/_{C_Q} < 1. \tag{3.2}$$

Similarly, the minimum-node-energy metric of path *P* is defined as:

$$E_P = \min(\{e_{P_j}\}) \text{ ,} \tag{3.3}$$

where e_{P_j} denotes the residual energy of intermediate nodes *j* along path *P*. Using only the minimum-node-energy metric, between two paths *P* and *Q*, path *P* is preferred to path *Q* if:

$$E_p > E_Q \text{ , or } {}^{E_p}\!\Big/_{E_Q} > 1. \tag{3.4}$$

We can combine the two metrics, giving equal weight to each of them. The resulting parameter C_P / E_P satisfies the property of a path metric such that path *P* is preferred to *Q* if:

$$ {}^{C_p}\!\Big/_{C_Q} < {}^{E_p}\!\Big/_{E_Q} \text{, or } {}^{C_p}\!\Big/_{E_P} < {}^{C_Q}\!\Big/_{E_Q} . \tag{3.5}$$

Thus, C_P / E_P, which considers both path cost and node energy, can be used as a new metric for route selection. Among all possible paths from a source to a destination, a path P_i with minimum C_{p_i} / E_{p_i} will be selected as the best path.

Various path energy metrics have been explored for minimum energy routing [49]–[51]. However, for such metrics to be effective there is a need for frequent updates of node- and link-related information from the neighbours, which will consume additional bandwidth and energy. We will leave the cost-benefit analysis of implementing such metrics for future work.

In this work, we focus on a metric that is practical and easily implementable in a real energy-constrained WSN. We simplify the path cost component of the metric to the number of links (hop count) along the path to be similar to the three other established routing protocols DSDV, AODV, and DSR, and investigate the effectiveness of the path minimum-node-energy component. The new routing metric along a path *P, M_P,* used in the simulation is given as:

$$M_p = {}^{L_P}\!\big/_{E_P} \tag{3.6}$$

where L_P is the number of links along path *P*.

Route selection between candidate paths based on this simplified metric is illustrated in the following four cases:

i. Directly Connected Path:
For a direct connected (single-hop) path *P* between source and destination, there is no intermediate node, which means E_p is not defined. Provision is made in the algorithm to select the direct path since the direct path is the more desirable path where no intermediate node is needed.

ii. Equal minimum-node-energy paths *P* and *Q* ($E_P = E_Q$):
In this case, the metric *M* reduces to the simple hop-count metric for shortest path routing. If $L_P < L_Q$, the shorter path *P* will be selected. However, it should be noted that the shortest path routing can only happen for a short duration. After some data packets are routed, node energies along the selected path are consumed, increasing the value of M_P. If the increase results in $M_P > M_Q$, then path *Q* will be selected next.

iii. Equal hop count paths *P* and *Q* ($L_P = L_Q$):
In this case, the metric *M* reduces to the minimum-node-energy metric. If $E_P > E_Q$, which implies that $M_P < M_Q$, then path *P* with higher minimum-node-energy will be selected. After some data packets are routed, node energies along the selected path are consumed, decreasing the value of E_P. If the decrease results in $M_P > M_Q$, then path *Q* will be selected next.

iv. Paths *P* and *Q* with different hop count and minimum-node-energy:

This is most general and the scenario for most cases. Based on the metric, path *P* will be selected if $E_P > (L_P / L_Q) * E_Q$, i.e. the minimum-node-energy for path P must be greater than that of path *Q* by a ratio of (L_P / L_Q). Data will be sent along path *P* and all nodes along this path will consume their energies for transmitting and receiving data packets. Path *P* will continue to be the preferred path until the energy consumption along path *P* results in $E_P < (L_P / L_Q) * E_Q$, and path *Q* will be selected next. Similarly, the switching of the preferred path from Q to P will happen when $E_P > (L_P / L_Q) * E_Q$. The alternate selection of path *P* and *Q* by this metric has the energy-balancing effect of maintaining a constant minimum-node-energy ratio between the two paths, i.e.

$$E_P / E_Q = L_P / L_Q . \qquad (3.7)$$

3.3.2 Transport and Storage of Node Energy Information

To incorporate the proposed energy-balancing metric effectively, the sensor nodes that make the routing decision must be provided with the updated residual node energy information of all nodes along the possible paths. Efficient transport and storage of the residual node energy information must be considered in the new protocol.

In the DSR protocol, a source node selects the best end-to-end route to the destination node. It provides an address list of the intermediate nodes through which the packets are forwarded in order to reach the destination. In EB-DSR, additional node energy fields associated with the corresponding intermediate nodes are added to the source route headers of RREQ and RREP packets. Each intermediate node, upon receiving these control packets, will first update this field with its current node energy level before it forwards the packets downstream. By piggybacking energy information on the source route headers, node energy information is communicated with minimum added overhead.

During the route discovery process, RREQ carries the energy information from data source to data destination, and RREP carries the energy information in the other direction. All nodes receiving these messages directly or indirectly (by overhearing) will keep the new energy information for future use. After the route discovery process is performed, the source node will have sufficient updated node residual energy information along the candidate paths for route selection based on the proposed metric as shown in **Fig. 3.4**.

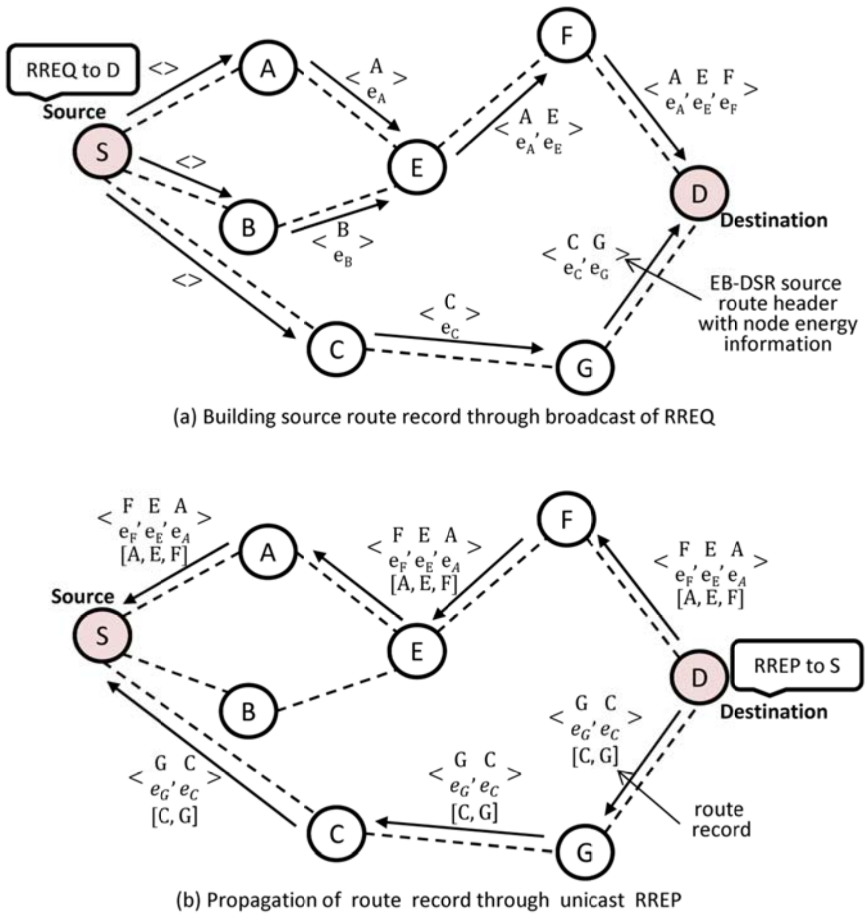

RREQ to D
Source
Destination
EB-DSR source route header with node energy information
(a) Building source route record through broadcast of RREQ
RREP to S
route record
(b) Propagation of route record through unicast RREP

Fig. 3.4 EB-DSR route discovery process with node energy information in the source route header

The pseudo-code of the EB-DSR route discovery process is shown in **Fig. 3.5**.

Route Discovery Process

```
received DATA from application :
    if (route_to_destination exists) {
        route_selection()              // perform route selection using the routing metric
        send_DATA()                    // send DATA packet out using the selected route
    }
    else {
        buffer_DATA()                  // put DATA packet in the data buffer
        construct_RREQ()               // construct RREQ packet
        broadcast_RREQ()               // broadcast RREQ packet with empty source route header
    }

received RREQ (S → D):
    cache_route (SRHeader)             // copy route from source route header into route cache
    cache_node_energy (SRHeader)       // copy node energy info into node energy table
    if (myNodeId == destination) {     // RREQ targeted at me
        process_RREQ()                 // process the RREQ packet
        construct_RREP()               // construct RREP packet
        send_RREP()                    // send RREP packet back to source
    }
    else if (RREQ is new) {                            // this is  a new RREQ for the intermediate node
        append_SRH (myNodeID, myNodeEnergy)     // append myNodeEnergy to source route header
        set_next_alert (energy_level, D)        // set next alert energy level to node D in alert table
        broadcast_RREQ()                        // broadcast RREQ with updated source route header
    }
    else {
        discard(RREQ)                  // procedure to discard RREQ
    }

received RREP (D → S):
    cache_route (SRHeader)             // copy route from source route header into route cache
    cache_node_energy (SRHeader)       // copy node energy info into node energy table
    if (myNodeId == destination) {     // RREP targeted at me
        process_RREP()                 // process the RREP  packet
        route_selection()              // perform route selection using the routing metric
        send_bufferedDATA()            // send out DATA packet in the data buffer using selected route
    else {
        set_next_alert (energy_level, S)  // set next alert energy level to node S in alert table
        forward_RREP()                 // forward RREP packet using the source route header
    }
```

Fig. 3.5 Pseudo-code of EB-DSR route discovery process

Each EB-DSR node maintains a *node-energy table* that stores the residual node energy level of other nodes in the network by reading them from the source route headers it receives. When data packets need to be routed, the table is referenced by the routing algorithm in order to compute the energy-balanced metric *M* for all available candidate paths before the path with the lowest *M* is selected.

Energy entries in the node energy table are updated with the following considerations:

i. An intermediate node may receive multiple packets with node energy information for the same nodes, and the sequence of packet arrival may not be the same as the time sequence of the node energy updated by the originating nodes. In our simulations, we have assumed that there is no energy harvesting capability in the network and node energy is a strictly decreasing function with time, a node-energy table entry will only be updated if the energy state in the source route header for that particular node is lower. In the case of an energy harvesting network where nodes may be recharged, the assumption could be removed and node energy could be updated accordingly.

ii. To reduce the memory requirements of the sensor node, the size of the node-energy table may not be able to accommodate all the nodes in the network. When the table is full and there is energy information for a new node, the highest energy entry in the table is discarded. This is because the lower energy nodes are more important in the computation of energy-balancing metric *M*.

3.3.3 Energy Alert Mechanism

Obtaining node energy status from the initial Route Discovery process is not sufficient. Energy entries in the table become outdated after some time and need to be updated. To reduce the routing and energy overhead, we use an energy-efficient algorithm that keeps the sending of energy alert packets to the minimum.

As described in section 3.2.2, RERR packets are used for route maintenance in DSR. RERR packets in DSR are sent when a node detects a link failure. The RERR packet header contains an “Error Source Address” and an “Error Destination Address” which specify the nodes at the two ends of the broken link. A new type of RERR is defined in EB-DSR to carry out the energy alert function. To differentiate Energy Alert packets from normal RERR packets, the Energy Alert packet header contains the address of the alerting node for both the Error Source and Destination addresses. **Fig. 3.6** illustrates the propagation of the Energy Alert packet from a low energy node to the source node that is actively transmitting data through it.

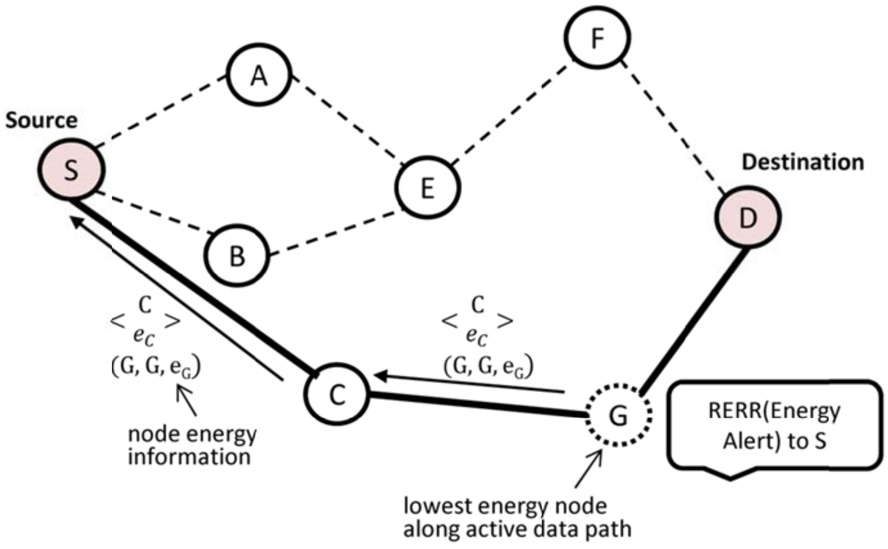


Fig. 3.6 Propagation of RERR(Energy Alert) to the source node

An important parameter to consider in the energy alert mechanism is the energy *alert interval*. In our simulations, the *alert interval* is set to 5% of the initial energies of the nodes. To minimize the overhead, Energy Alert packets will only be triggered and sent when the following three conditions are met:

i. The node is an active intermediate node;

ii. It is the minimum energy node along the active data path; and

iii. Its node energy has dropped below a pre-defined interval (alert interval) from its previous update to the same source node. The next-alert information is kept in the node-energy table to keep track of the alert status.

The pseudo-code of the EB-DSR energy alert process is shown in **Fig. 3.7**.

Energy Alert Process

```
received DATA (S → D) from neighbour :
    cache_route (SRHeader)                // copy route from source route header into route cache
    cache_node_energy (SRHeader)          // copy node energy info into node energy table
    if (myNodeId == destination) {        // DATA packet is for me
        process_DATA()
    }
    else {
        update_SRH (myNodeID, myNodeEnergy)     //update myNodeEnergy to source route header
        forward_DATA()                          // forward DATA packet using the source route header
        alert_level = alert_look_up (S)         // look up alert level to S  from the alert table

        if ((myNodeEnergy < alert_level ) && (minEnergy_in_SRHeader == true)) {
            send_energy_alert (myNodeEnergy, S)       // send Energy Alert packet to source node S
        }
    }

received RERR(Energy Alert):
    cache_route (SRHeader)                // copy route from source route header into route cache
    cache_node_energy (SRHeader)          // copy node energy info into node energy table
    update_EA()                           // update alert_node energy info into node energy table
    if ((myNodeId == destination) && (Databuffered)) {    // energy alert for me and more data to send
        route_selection()           // re-compute route using the routing metric
        send_bufferedDATA()         // send out DATA packet in the data buffer using selected route
    }
```

Fig. 3.7 Pseudo-code of EB-DSR energy alert process

3.4 Simulation Setup

3.4.1 Network Model

An ad hoc network is set up with 50 nodes randomly located within a square grid of 500m x 500m. These 50 nodes are divided into two groups, with 25 high-energy nodes and 25 low-energy nodes. The high-energy nodes have sufficient energy to last the whole simulation while the low-energy nodes will be exhausted before the end of simulation. The heterogeneous network models a large scale WSN that consists of multiple types of sensor nodes with different sensing functions and energy capacities working together. The source and destination nodes are located at equal distance at both ends of the grid as shown in **Fig. 3.8**. The source and destination nodes are also high-energy nodes so that no packets are lost due to their exhaustion.

We have used constant bit-rate (CBR) traffic of 1 packet per second with fixed length packets of 512 bytes. For physical layer, we have used the two-ray ground reflection propagation model which considers both the direct path and a ground reflection path, with ns-2 (version 2.35) default carrier sensing range of 550m and packet reception range 250m (ns-2 uses binary decision for packet reception). We have also used a simple energy model, in which node energy is only consumed when transmitting, receiving and overhearing packets.

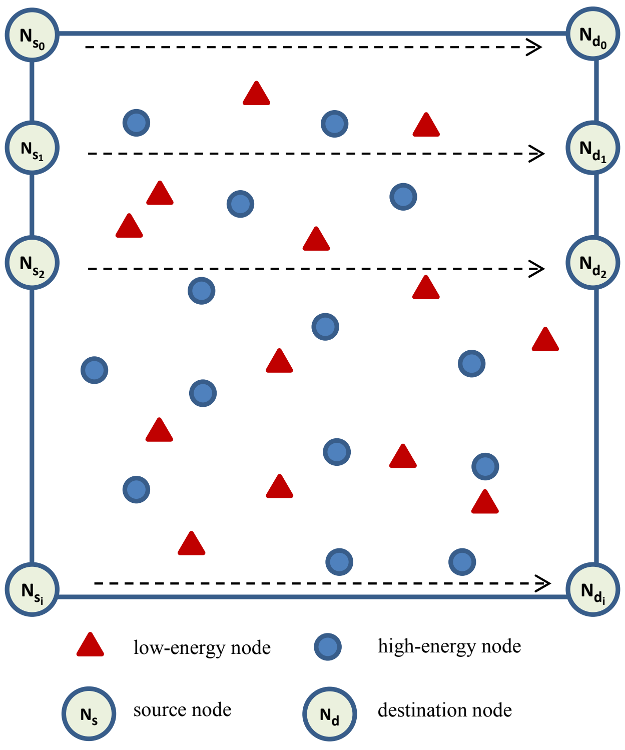


Fig. 3.8 Network model for ns2 simulation

As the power consumption ratio for transmitting and receiving packets (*tx_rx_ratio*) varies across different hardware and applications [112][113], and experimental measurements [114] have shown that actual energy consumptions are very different from the hardware specifications, the performance of the four routing protocols DSDV, AODV, DSR, EB-DSR are evaluated with different values of *tx_rx_ratio* in the static network simulations. It has to be emphasized that the relative results are more important than the absolute numbers. **Table 3.1** shows the parameters used in the simulations of the 4 routing protocols under study.

The ability to respond to topology changes due to node mobility is also one of the most important design criteria of WSN routing protocols. In the second part of our simulations, the performance of the routing protocols are compared in mobile network scenarios, in which each intermediate node moves independently with a random velocity, which changes the network topology over time.

Table 3.1 WSN simulation parameters for ad-hoc routing protocols

Parameter	**Value**
Grid size	500m x 500m
Number of intermediate nodes	25 (high-energy), 25 (low-energy)
Number of source-destination pairs	5
Data rate (CBR)	1 packet/s
Packet size	512 bytes
Interface queue length	50 packets
MAC protocol	IEEE 802.11
Simulation time	500s
Simulated routing protocols	DSDV, AODV, DSR, EB-DSR
Propagation model	Two-ray ground reflection
Carrier sensing range	550m
Transmission range	250m

3.4.2 Performance Metrics

3.4.2.1 Network Lifetime

The primary performance metric of interest to us is network lifetime. There are different definitions of network lifetime available in the literature. In [115], the definitions can be categorised as follows:

i. Time to which a pre-defined fraction of nodes is exhausted;

ii. Time to which emergence of first partition in the network occurs; and

iii. Time to which packet delivery rate drops below a pre-defined value.

In our simulations, the 25 high-energy nodes remain alive until the end of the simulations. They keep the network connected and the packet delivery rate remains high. Therefore, the first category of the definitions is appropriate. For a simple comparison among different routing protocols, we adopt the most common definition for network lifetime within the first category, which is the duration from the beginning of the network operation to the first node failure.

3.4.2.2 Data Load Ratio on Low-energy Nodes (*DLR$_{le}$*)

This metric is defined as the ratio of the number of CBR data packets transmitted and received by the low-energy nodes to the total number of data packets transmitted and received by all the intermediate nodes. A lower ratio indicates that the routing protocol is more effective in diverting data packets away from the low-energy nodes and is therefore more energy-balanced:

$$DLR_{le} = \frac{\sum_{le\_nodes}(CBR_{tx} + CBR_{rx})}{\sum_{all\_nodes}(CBR_{tx} + CBR_{rx})} \times 100\% \tag{3.8}$$

3.4.2.3 Packet Delivery Ratio (*PDR*)

PDR is the ratio of the total number of data packets received at the destination nodes to the total number of CBR data packets generated at the source nodes. A higher value of *PDR* indicates that the routing protocol is more successful in delivering the data packets from the source nodes to the destination nodes. This metric characterizes both the correctness and the reliability of a routing protocol:

$$PDR = \frac{\sum_{dest\_nodes} CBR_{rx}}{\sum_{src\_nodes} CBR_{tx}} \times 100\% \tag{3.9}$$

3.4.2.4 Average End-to-end Packet Delay (*AvDelay*)

AvDelay measures the average time taken for a data packet to be successfully delivered to its destination. Delays in WSNs are caused by buffering during route discovery, waiting at the interface queue, MAC retransmission, and propagation and transmission delays. *AvDelay* is computed by summing up the end-to-end delay of each CBR data packet and dividing it by the total number of successfully delivered data packets. The lower the end-to-end delay, the better the application performance:

$$AvDelay = \frac{\sum_{1}^{CBR_{total\_rx}} (CBRrecvTime - CBRsentTime)}{CBR_{total\_rx}} \tag{3.10}$$

3.4.2.5 Average Hop-count

Average hop-count of data packets is defined as the average number of routers (hops) a data packet needs to traverse the network to reach its destination. In general, a lower hop-count contributes to lower packet delay and lower energy consumption. In a mobile network, this metric also measures how well a routing protocol adapts to network topology changes.

3.4.2.6 Normalised Routing Overhead (*NRO*)

Normalised routing overhead is also known as normalized routing load. It is defined as the number of routing packets (RREQ, RREP, and RERR) transmitted per data packet delivered to the destination. Each hop-wise transmission of a routing packet is counted as one transmission. Routing overhead measures the scalability of a routing protocol, the ability to function in congested or low-bandwidth environments, and the efficiency in terms of node energy consumption. Protocols with high routing overhead could also increase the probability of packet collisions and increase data packet delays. A routing protocol with a lower normalised routing overhead is considered to have better performance:

$$NRO = \frac{\sum_{all\_nodes}(RREQ + RREP + RERR)}{\sum_{dest\_nodes} CBR_{rx}} \quad (3.11)$$

3.5 Simulation Results – Static Network Scenarios

The simulations are performed for DSDV, AODV, DSR and EB-DSR routing protocols with *tx_rx_ratio* taking the values of 2, 4, 6, 8 and 10. Twenty independent runs are simulated for each protocol for each scenario and the average values of each performance parameter are presented. In all our scenarios, it is observed that average lifetime, *PDR*, DLR_{le}, and average hop-count performances have small standard deviations of less than 2%. For *AvDelay* and normalised routing overhead, the standard deviations are larger and will be plotted together with the averages.

3.5.1 Network Lifetime Performance

Fig. 3.9 shows the lifetimes of each of the 25 low-energy nodes. All 25 low-energy nodes are depleted well before the completion of 500s simulation time. We compare the network lifetimes of each protocol, which is the time to first node failure as defined in the previous section.

Table 3.2 Lifetime performance (static network)

Routing Protocol	***tx_rx_ratio* = 2**	***4***	***6***	***8***	***10***
	Network Lifetime – First Node Failure (s)				
DSR	277.6	236.2	203.3	187.3	160.3
DSDV	262.4	248.3	233.5	221.4	211.2
AODV	278.4	230.4	202.7	179.3	167.1
EB-DSR	292.7	284.6	288.3	278.6	289.1
	Standard Deviation of Node Lifetime (s)				
DSR	12.3	19.6	30.6	33.8	42.4
DSDV	8.0	8.7	12.7	13.0	15.2
AODV	14.8	21.5	29.4	37.2	42.0
EB-DSR	13.3	11.9	13.8	11.0	11.3

Table 3.2 shows the comparison of network lifetimes among the four routing protocols. It is apparent that EB-DSR has the longest network lifetime compared with the other protocols under study for all the scenarios. In addition, it is also least affected by the changing *tx_rx_ratio*. While the node lifetimes for the other protocols drop quickly as *tx_rx_ratio* increases, the lifetime performances of EB-DSR nodes remain consistent across the 5 scenarios in a small range between 278s and 292s. The relative lifetime performance of EB-DSR gets better with higher *tx_rx_ratio*.

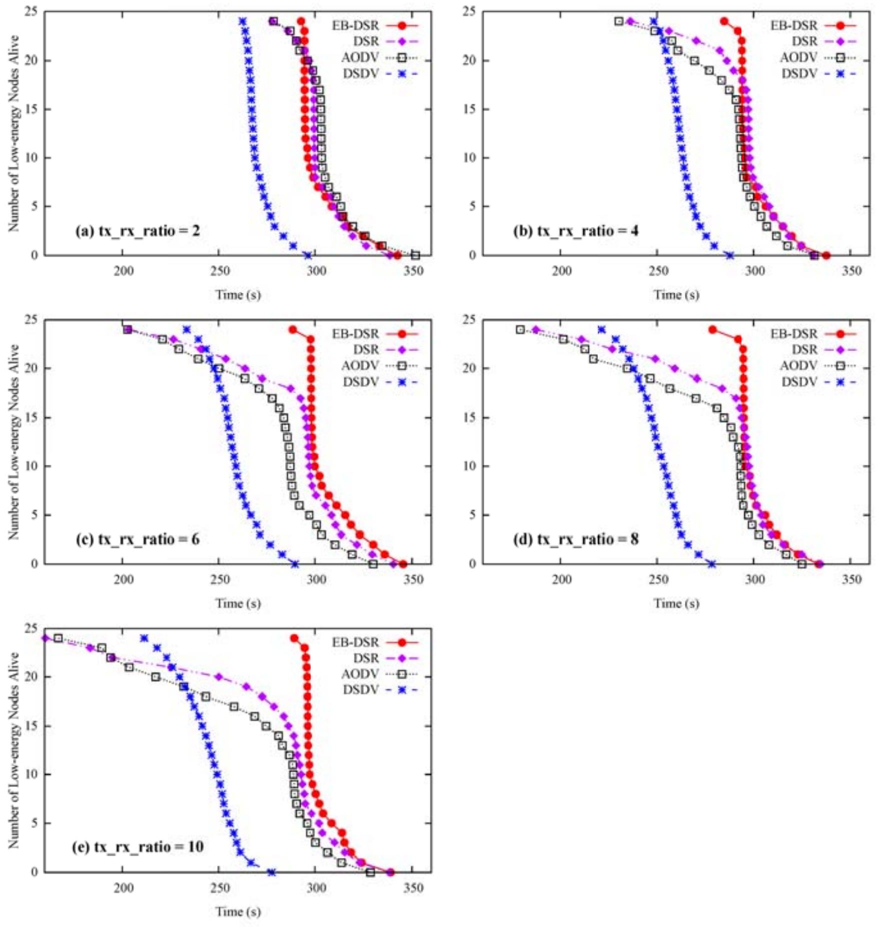


Fig. 3.9 Low-energy nodes lifetime performance for different transmit-receive power consumption ratio

The energy consumption among the low-energy nodes in EB-DSR are also more "balanced" as can be seen from its small standard deviations of node lifetimes. It is interesting to note that DSDV is more energy-balanced than AODV and DSR. The better energy-balancing performance is possibly due to the way DSDV performs its regular routing updates, preferring routes with higher sequence numbers when routing metrics are equal. The best route for a particular source-destination pair typically changes during the routing updates in a dense network, such as the one we studied. This has the unintended positive effect of distributing data load among more intermediate nodes. However, as the only table-driven routing protocol in the study, DSDV has the highest routing overhead and thus has the lowest average node lifetimes as clearly shown in **Fig. 3.9(a) – (e)**.

We have also performed simulations of a scenario where all 50 intermediate nodes are low-energy nodes. As shown in **Fig. 3.10** and **Table 3.3**, EB-DSR has a longer network lifetime than the two reactive protocols DSR and AODV and has the smallest standard deviation of the node lifetime. DSDV has 2% higher network lifetime than EB-DSR, however, all the nodes are depleted in less than 300s into the simulation time.

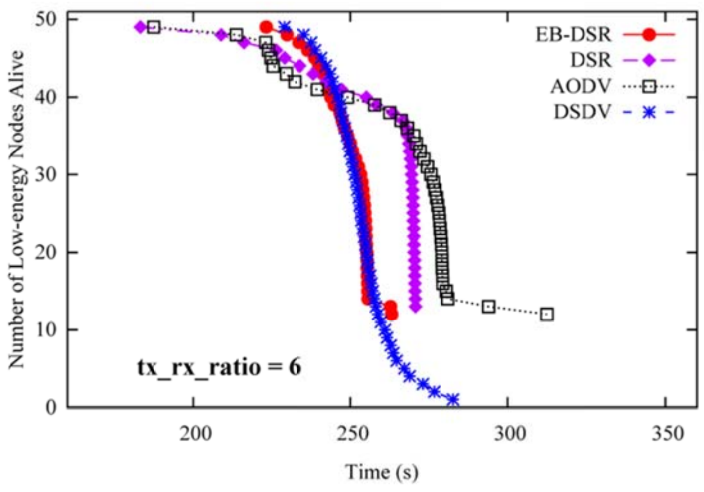


Fig. 3.10 Lifetime performance for a network with 50 low-energy intermediate nodes

Table 3.3 Lifetime performance (50 low-energy nodes)

Routing Protocol	***Network Lifetime – First Node Failure (s)***	***Std. Dev. of Node Lifetime (s)***
DSR	183.1	21.0
DSDV	229.1	10.1
AODV	187.4	25.6
EB-DSR	223.2	8.6

3.5.2 Data Load on Low-energy Nodes

As shown in **Fig. 3.11**, the effectiveness of EB-DSR in routing data packets away from low-energy nodes is clearly demonstrated. Although the number of low-energy nodes constitutes 50% of intermediate nodes, less than 2% of total data load is routed through these low-energy nodes. The other 3 protocols do not have energy-aware routing metrics to differentiate between high-energy and low-energy nodes, and therefore they have significantly higher percentages of data load on low-energy nodes ranging from 21% to 36%.

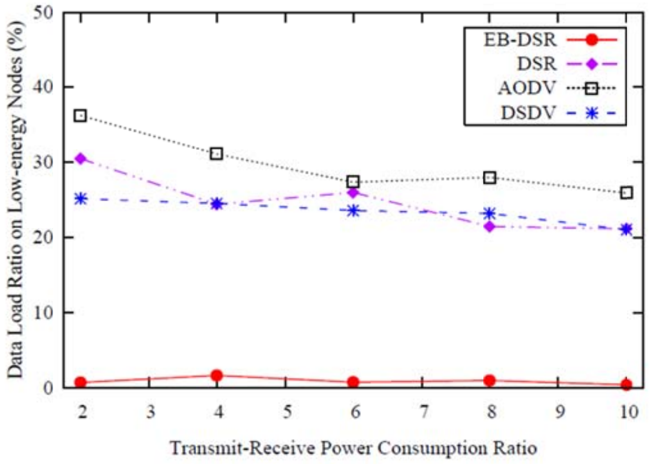


Fig. 3.11 Data load ratio on low-energy nodes (static network)

3.5.3 Packet Delivery Ratio

As shown in **Fig. 3.12 (top),** the three reactive protocols EB-DSR, DSR, and AODV deliver almost 100% of packets in all the scenarios simulated. DSR and EBDSR have a "packet salvaging" feature in which, when a node discovers that it cannot forward a data packet due to the broken link, it searches its own cache to find an alternate route from itself to its destination in order to forward this packet. AODV also has a "local repair" feature to repair breaks in active routes locally instead of notifying the source. These two features repair links with less overhead as well as reducing packet loss. Without these features, DSDV delivers a lower packet delivery ratio at about 92%.

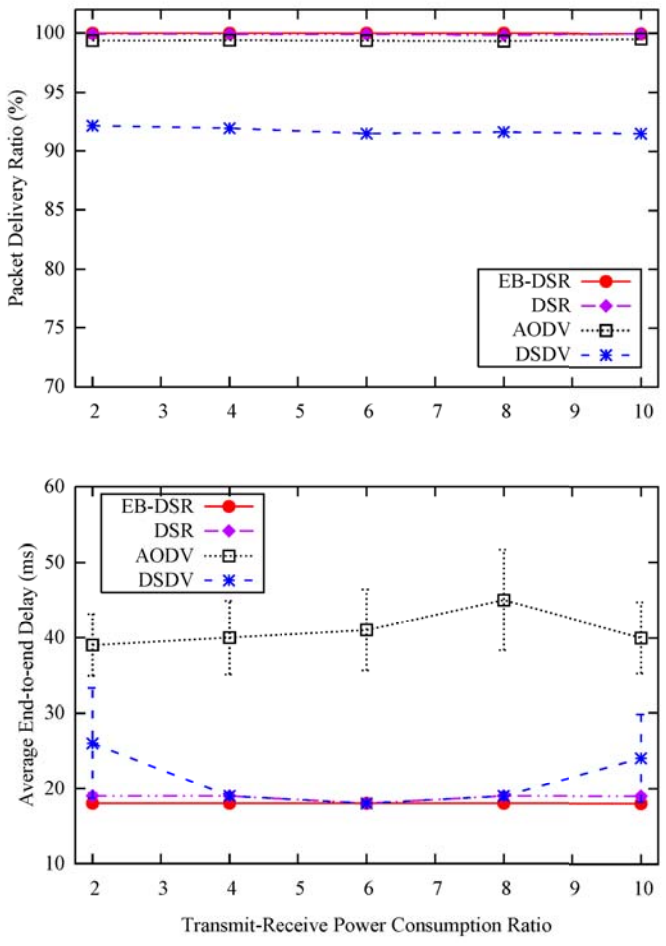


Fig. 3.12 Packet delivery ratio and end-to-end delay performance (static network)

3.5.4 Average End-to-end Packet Delay

As shown in **Fig. 3.12 (bottom)**, both DSR and EB-DSR have the lowest end-to-end delays under 20ms in all scenarios. DSR and EB-DSR nodes maintain multiple route entries in their route cache. When a selected route fails, the source node is able to quickly find an alternative route to the destination, reducing the end-to-end delay. This proves to be useful in the static heterogeneous network scenarios, in which route failures are generally due to energy depletion of low-energy nodes. DSDV also has a good delay performance as it is a proactive protocol, with quick notifications about link failures through routing updates. Upon receiving failure notifications, other nodes in the network will be able to compute new routes and update their routing tables. AODV has the worst delay performance, with an average at around 40ms. The variations are also large across different scenarios. This is because AODV maintains only a single route to a destination; a new route discovery process has to be initiated when the original route fails, and this process introduces additional packet delay.

3.5.5 Average Hop-count

As shown in **Fig. 3.13 (top)**, average hop-counts vary within a narrow range of between 3.08 and 3.33 among the four routing protocols. DSDV has the lowest average hop-count due to its proactive nature. The shortest paths to the destinations are maintained and updated periodically. For reactive protocols, longer routes will be taken by intermediate nodes to salvage data packets when they encounter link failures.

In DSR and EB-DSR, routes may be shortened if one of the intermediate nodes becomes unnecessary. If a node overhears a packet carrying a source route, in which the address of the node appears in the later portion of the packet's source route, it infers that the intermediate nodes before itself in the source route are no longer needed in the route. It can then send a "gratuitous" RREP to the original sender of the packet, shortening the original source route. This automatic route shortening feature results in a better hop-count performance for DSR and EB-DSR, compared to AODV.

3.5.6 Normalised Routing Overhead

The proposed EB-DSR scheme has the lowest normalised routing overhead as shown in **Fig. 3.13 (bottom)**. Additional Energy Alert messages (carried via RERR) that are introduced for energy-balancing are more than compensated for by the reduction of RERR messages due to link failure, based on the comparison between DSR and EB-DSR. Aggressive route caching in DSR and EB-DSR also results in the reduction of the routing overheads in these two protocols compared to AODV. DSDV, being a proactive protocol with regular routing updates, has the highest routing overhead. At around 1.5 routing packets per data packet delivered, the routing overhead in DSDV is 2.5 times higher than AODV, and 5 times higher than DSR and EB-DSR.

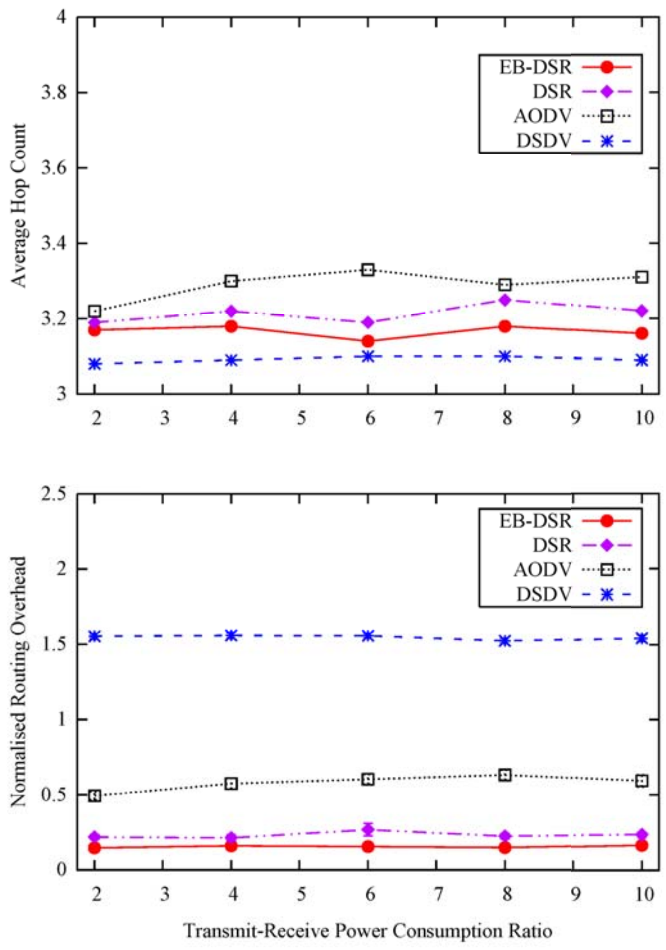


Fig. 3.13 Average hop count and normalised routing load performance (static network)

3.6 Simulation Results – Mobile Network Scenarios

The same 500m x 500m grid network with 50 intermediate nodes is used for mobile network simulations. While the source and destination nodes remain stationary at both ends of the grid, the intermediate nodes are given freedom to move randomly within the grid. Node mobility provides additional challenges to the routing protocols because the topology of the network changes constantly. Route entries in the routing table become stale quickly and more routing overhead is generated to update the link status and maintain the correctness of the routes.

Table 3.4 shows the parameters used in the mobile network simulations:

Table 3.4 WSN simulation parameters for mobility

Parameter	**Value**
Grid size	500m x 500m
Number of intermediate nodes	25 (high-energy), 25 (low-energy)
Number of source-destination pair	5
Data rate (CBR)	1 packet/s
Packet size	512 bytes
Interface queue length	50 packets
Node speed	U(1m/s, 10m/s)
Pause time	100s, 200s 300s, 400s, 500s
tx_rx_ratio	6
MAC protocol	IEEE 802.11
Simulation time	500s
Simulated routing protocols	DSDV, AODV, DSR, EB-DSR
Propagation model	Two-ray ground reflection
Carrier sensing range	550m
Transmission range	250m

The Random Waypoint model [116] is used to model node mobility in our simulations. Mobile nodes that follow this model move independently to randomly chosen destinations with randomly selected velocities. The nodes then remain stationary for a period of time known as *pause time* before continuing the random movement. This process repeats until the simulation ends. There are two key parameters in modelling node mobility in the Random Waypoint model, the node speed and the *pause time*. In our simulations, the speeds of intermediate nodes are uniformly distributed between 1 m/s and 10 m/s. The pause times range from 100s to 500s, with 100s *pause time* being the highest node mobility.

3.6.1 Network Lifetime Performance

Network lifetime performance for each of the 25 low-energy nodes is plotted in **Fig. 3.14**. All 25 low-energy nodes are depleted well before the completion of the 500s simulation time

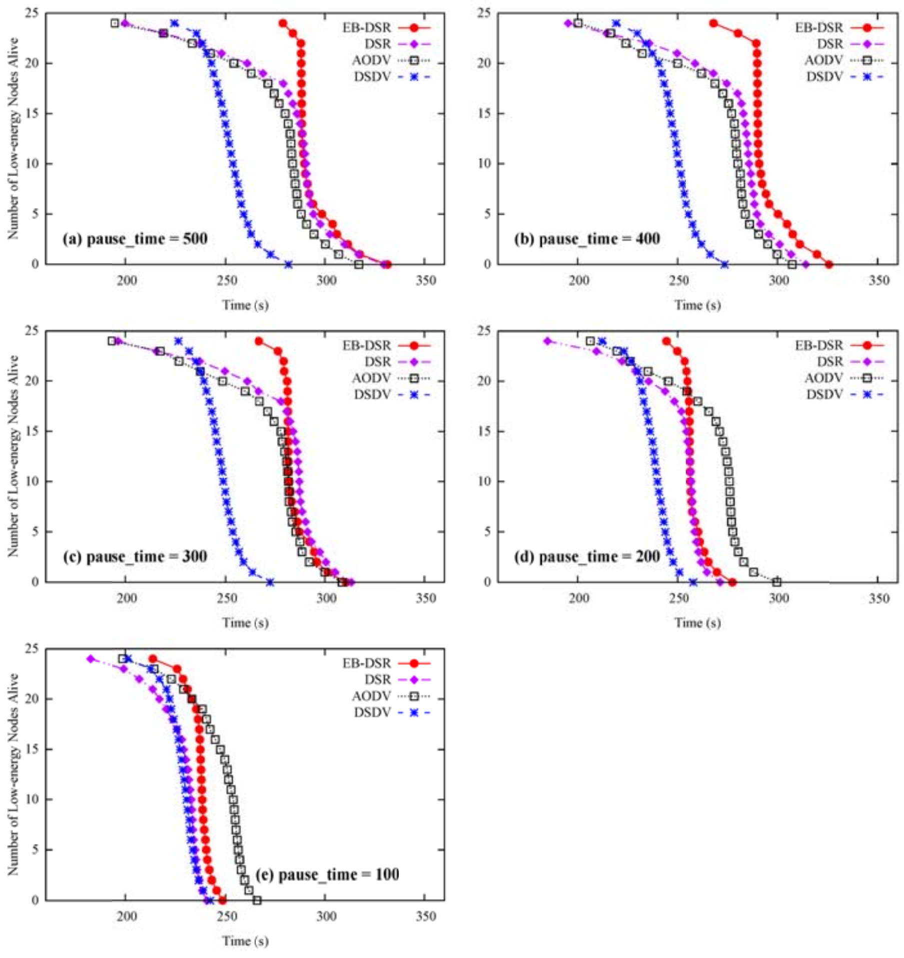


Fig. 3.14 Low-energy nodes lifetime performance for different pause times

The comparison of network lifetimes among the four routing protocols is shown in **Table 3.5**. Similarly to the static network scenarios, EB-DSR also has the longest lifetime performance in all mobility scenarios simulated. The relative lifetime performance of EB-DSR gets better with lower mobility (larger pause time). It is also apparent that energy consumptions among the low-energy nodes in EB-DSR are more balanced, which can be seen from the narrow ranges of its low-energy node lifetimes.

Table 3.5 Lifetime performance (mobile network)

Routing Protocol	***Pause Time = 500***	***400***	***300***	***200***	***100***
	Network Lifetime – First Node Failure (s)				
DSR	199.8	195.3	196.4	185.0	182.3
DSDV	224.4	219.2	226.5	212.1	201.2
AODV	194.6	200.3	192.9	206.3	198.6
EB-DSR	279.0	268.1	266.8	244.4	213.7
	Standard Deviation of Node Lifetime (s)				
DSR	28.4	26.7	26.5	18.8	13.3
DSDV	11.7	11.3	9.8	9.1	8.5
AODV	26.8	25.8	26.3	22.1	15.5
EB-DSR	11.4	11.6	8.5	6.2	6.7

3.6.2 Data Load on Low-energy Nodes

Similarly to the static network scenarios, the effectiveness of EB-DSR in routing data packets away from low-energy nodes is clearly demonstrated as shown in **Fig. 3.15**. Less than 6% of the total data load is routed through the low-energy nodes in EB-DSR. The other 3 protocols do not differentiate between high-energy and low-energy nodes in making routing decisions, and therefore they have significantly higher percentages of data load routed through the low-energy nodes, ranging from 21% to 28%.

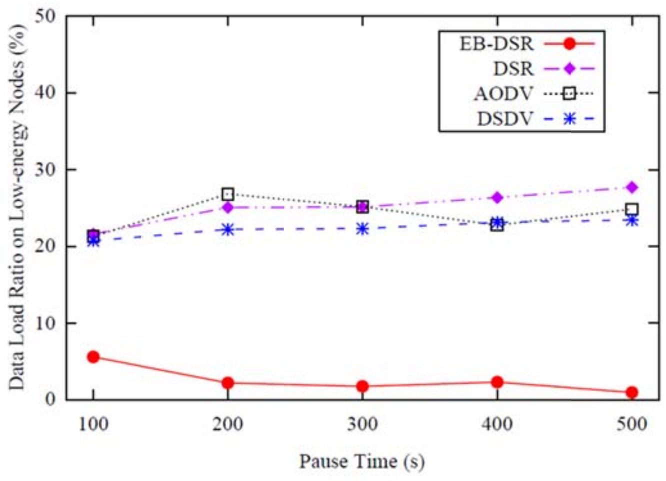


Fig. 3.15 Data load ratio on low-energy nodes (mobile network)

3.6.3 Packet Delivery Ratio

As shown in **Fig. 3.16 (top),** the three reactive protocols EB-DSR, DSR, and AODV have PDRs of almost 100% at low mobility (500s pause time). PDRs drop slightly to about 97% at the highest mobility (100s pause time). Again the “packet salvaging” and “local repair” features in these protocols play an important role in the high PDR performance. DSDV has poor PDR performance in the mobile scenarios, dropping from 90% at 500s pause time to 74% at 100s pause time. In high mobility networks, route entries become invalid quickly. Most of the packets dropped in DSDV are due to invalid route entries. As DSDV maintains only a single route per destination, the packets are dropped if they cannot be delivered due to broken links.

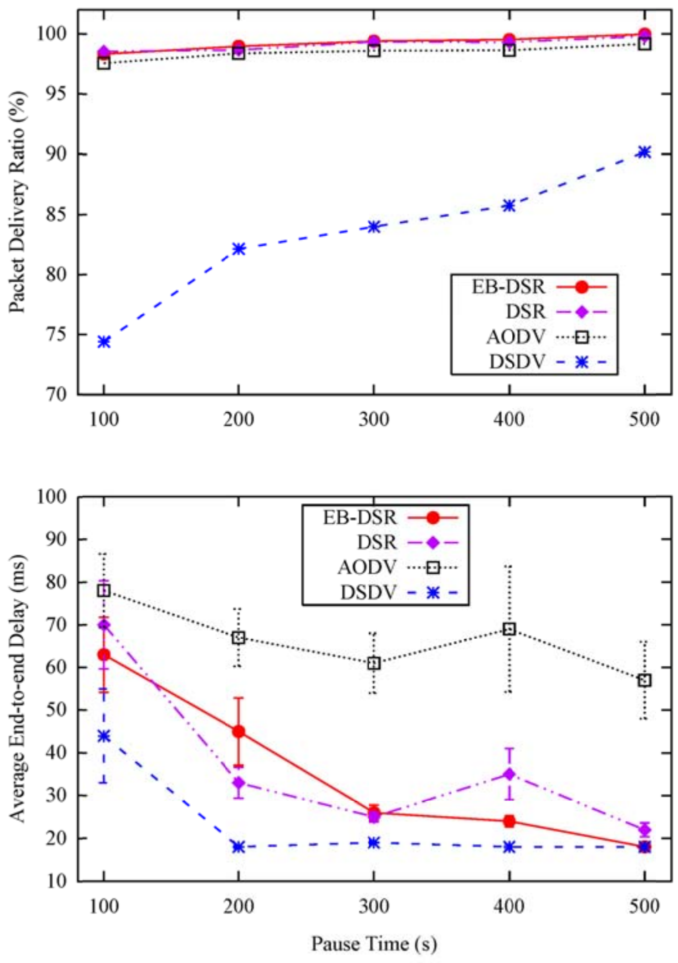


Fig. 3.16 Packet delivery ratio and end-to-end delay performances (mobile network)

3.6.4 Average End-to-end Packet Delay

In general, all four protocols have higher end-to-end packet delays when node mobility is higher. With higher mobility, network topology changes more frequently; data packets may need more tries on different route entries and thus take longer before they can be routed to the correct destinations.

In the mobility scenarios, DSDV has the lowest packet delay of all four routing protocols. However, the packet delay performance of DSDV does not reflect the full picture in high mobility networks. This is because packet delay is computed based on delivered data packets only and DSDV has much fewer delivered packets in high mobility networks due to high packet loss rates.

In all cases, AODV has higher delays than EB-DSR. The difference is more significant at low mobility than at high mobility. The average end-to-end packet delay for AODV is 3.16 times of that in EB-DSR at a pause time of 500s. The delay drops to 1.24 times at a pause time of 100s.

3.6.5 Average Hop-count

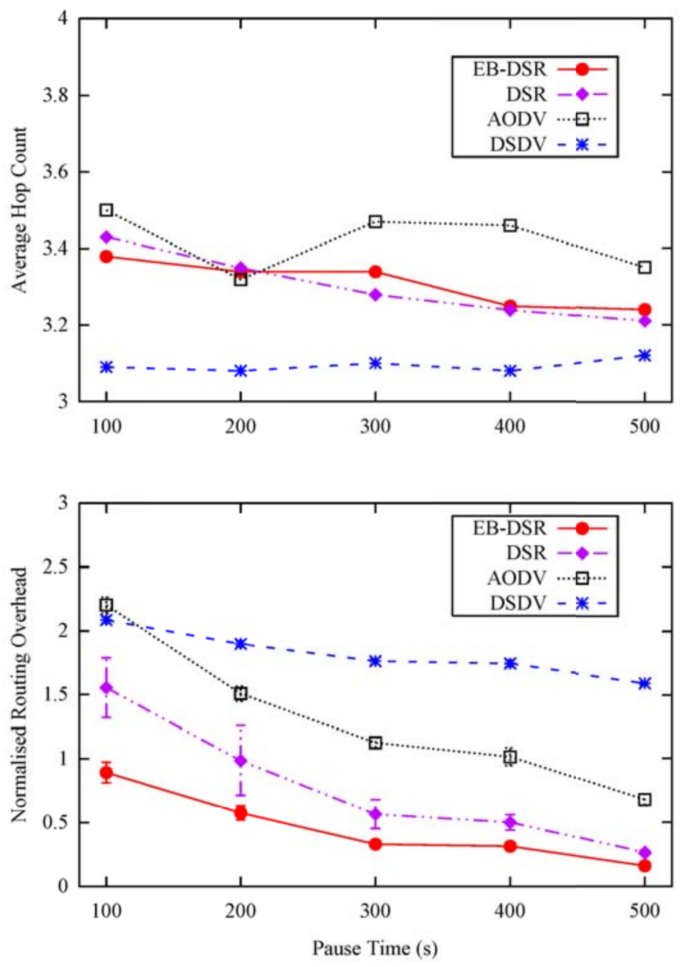


Fig. 3.17 Average hop count and normalised routing load performance (mobile network)

As shown in **Fig. 3.17(top)**, DSDV has the lowest average hop-count due to its proactive nature, in which the shortest paths to the destinations are maintained and updated periodically. The average hop-count varies within a narrow range of 4% under different mobility scenarios. This is because DSDV drops data packets when invalid route entries are encountered. It does not try to salvage these packets as in the other three protocols, which results in the increase of average hop-counts.

DSR and EB-DSR have better hop-count performance than AODV in general due to the route shortening feature present in both protocols. All three protocols record higher hop-counts with higher mobility as links are broken more frequently.

3.6.6 Normalised Routing Overhead

Normalised routing overhead increases with an increase in mobility for all four routing protocols as shown in **Fig. 3.17(bottom).** EB-DSR has the lowest normalised routing overhead in all cases. DSDV has the highest normalised routing overhead in most of the cases due to its proactive nature.

As mobility increases, more links are broken and route discovery process becomes more frequent in reactive protocols. However, as the route discovery process in AODV is dominated by RREQ which is broadcast in nature, routing overhead increases more rapidly than EB-DSR and DSR, in which route discovery is dominated by unicast RREP. At a high mobility of 100s pause time, AODV records the highest routing overhead among the four protocols.

3.7 Chapter Summary

In this chapter, we discussed the issue of network lifetime performance in wireless sensor networks caused by the unbalanced routing of data traffic. We proposed a multi-path Energy-balanced Dynamic Source Routing (EB-DSR) protocol that is fully distributed and computationally efficient. The proposed protocol also provides a novel energy update mechanism to delivery node energy information through the network efficiently.

Simulations of four routing protocols, DSDV, AODV, DSR and the proposed EB-DSR are run and their performances in various static and mobile network scenarios are compared. Performance metrics used for comparison include network lifetime, data load on low-energy nodes, packet delivery ratio, average end-to-end packet delay, average hop-count and normalised routing load.

Results from the simulations have shown that EB-DSR is able to prolong the network lifetime effectively through an energy-balanced, multipath approach, while maintaining high packet delivery ratio in both static and mobile heterogeneous WSNs. The results have also shown that the relative performance of EB-DSR in terms of data load distribution and normalised routing overhead are much better than the other protocols in the study. EB-DSR also has lower end-to-end packet delays than the other two reactive protocols in most of the cases.

Chapter 4

Energy-efficient Synchronisation Algorithms for Duty-cycle MAC

4.1 Introduction

The synchronisation algorithm adopted by the synchronous duty-cycle MAC protocol family derived from S-MAC is based on a fixed, periodic synchronisation packet broadcast algorithm (F-Sync) [95] by the sensor nodes in SYNC windows. This algorithm works fine when the network is sparse. When the network is dense, however, too many *sync* packets are generated and transmitted in the network, causing collisions and increasing energy consumption unnecessarily. Energy consumed for the synchronisation process in SYNC windows is not insignificant. For many of the synchronous MAC protocols implemented, the duration of the SYNC window time is about 50% of the DATA window time. In addition, many sensor nodes are able to go to sleep in the DATA windows when there is an active transmission in the neighbourhood not involving them [95], whereas all the sensor nodes need to stay active in the SYNC windows in the case of F-Sync. It is therefore worthwhile to examine the synchronisation process of duty-cycle MAC in more detail.

Another synchronisation algorithm compatible with synchronous duty-cycle MAC, found in literature attempting to improve the energy efficiency of the synchronisation process, is Intelligent Network Synchronisation (INS) [117]. INS attempts to reduce energy consumption by putting sensor nodes to sleep during SYNC windows when *sync* packets are not expected to arrive. INS has been shown to be effective when the network is sparse. However, when the network is dense, the energy performance of INS converges with F-Sync.

In this chapter, a new synchronisation algorithm, referred to as 1-Sync, is proposed. 1-Sync conserves energy by turning off the radios of sensors nodes in the SYNC windows after they receive one valid *sync* packet in the neighbourhood. The nodes will wake up periodically when transmission or reception of *sync* packets is necessary. The analytical energy consumption models and synchronisation performance of the three synchronisation algorithms are also presented and they are validated through extensive simulations. Both analysis and simulation results show that 1-Sync has better energy efficiency compared to F-Sync in all cases. Compared to INS, 1-Sync also has better energy efficiency except in very sparse network scenarios.

4.2 Existing Synchronisation Algorithms for Synchronous Duty-Cycle MAC

The listen period in synchronous duty-cycle MAC protocols is much longer than the clock drift. As such, a much looser synchronisation among neighbouring nodes is required compared with TDMA schemes with very short timeslots [95]. In addition, as the frame structure of synchronous duty-cycle MAC protocols provide only small time windows for exchanging timing messages, unidirectional single message broadcast is the most appropriate and energy efficient among the three approaches for synchronizing sleep/wakeup schedules of the sensor nodes.

4.2.1 Frame Structure

Synchronous duty-cycle MAC protocols such as S-MAC, DW-MAC and their derivatives divide their operating cycles into listen and sleep periods. A complete cycle of a listen and sleep period is called a *frame*. As shown in **Fig. 4.1**, the listen period, during which the node's radio is active, is further divided into SYNC and DATA windows.

SYNC windows are meant for the broadcast of *sync* packets to synchronise the clocks of neighbouring nodes so that they can be awake simultaneously. Data packets from upper layers, if any, will be sent after the start of DATA windows. A sensor node turns off its radio during sleep periods to conserve energy unless it is in the middle of sending or receiving data. In this case the radio will only be turned off upon completion of data transfer activities. Typically, the duration of the listen period is fixed and the duration of the sleep period is selected to achieve certain performance objectives that include packet delay and throughput. The duty-cycle of the MAC protocol is defined as the fraction of a listen period in a frame.

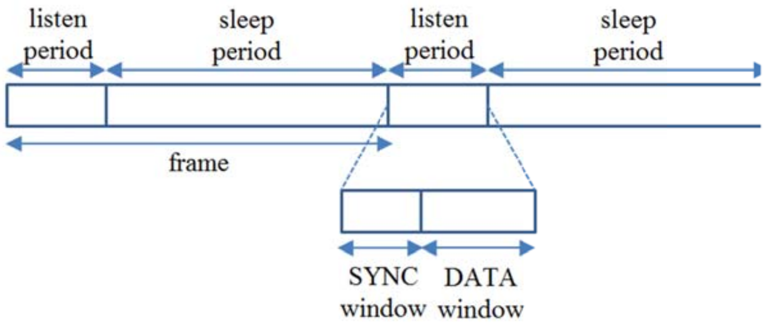


Fig. 4.1 Synchronous duty-cycle MAC frame structure

4.2.2 The Operation of F-Sync

F-Sync was proposed in [95] together with the S-MAC protocol and has since been the default synchronisation algorithm used in the synchronous MAC protocols that were developed later. As the neighbouring nodes need to coordinate their sleep/wakeup schedules, and the clock for each sensor node drifts independently from one another, the drifts can cause data loss if the clocks are left unsynchronised. Using the F-Sync algorithm, each sensor node broadcasts one *sync* packet in every N_{SP} frames to update the neighbours on its sleep/wakeup schedule for the prevention of long-term phase offset among the neighbours. The number of frames N_{SP} is selected as the period based on various network parameters such as clock drift, data rate, duty cycle, etc.

A *sync* packet is a very short packet that includes the sender address and its next sleep time. The next sleep time is relative to the moment that the sender starts transmitting the *sync* packet. A receiver adjusts its timer accordingly when it receives the *sync* packet.

At the beginning of each SYNC window, a sensor node wakes up to either transmit or receive a *sync* packet. If N_{SP} frames have elapsed since the last time the node sent a *sync* packet, it will schedule a new *sync* packet and follow a contention procedure for the transmission of the packet. It first starts a carrier sense timer and selects a random timeslot to sense the medium. If there is no transmission by the end of that timeslot, it wins the contention and starts sending its *sync* packet. During the contention window, if the medium is busy, it will revert to packet receiving mode and postpone the *sync* transmission to the next SYNC window. Upon receiving a *sync* packet, it re-synchronises its sleep/wakeup schedule with the time information given in the *sync* packet received. It stays idle for the entire SYNC window if there is no packet to transmit or receive. At no time will a sensor node go to sleep during SYNC windows. The detailed procedure of the proposed F-Sync algorithm is shown in **Fig. 4.2**.

F-Sync Algorithm

```
initialization:
    nextTxSync = $N_{SP}$
sync window begins:
    wakeup()
    if (nextTxSync != 0) {
        nextTxSync --      // not time to send sync yet
    }
    else {
        send_sync()        // procedure to send sync packet
        if (send_sync_successful) {
            nextTxSync = $N_{SP}$     // schedule next sync transmission
        }
    }
    if (sync_ received) {
        synchronise_node()  // procedure to synchronise clock
    }
sync window ends:
```

Fig. 4.2 F-sync algorithm with fully awake sensor nodes

Each node stays awake in the SYNC windows to ensure it receives the *sync* packets from all its neighbours. As the number of nodes in the 1-hop neighbourhood (node density) increases, there are more *sync* packets scheduled and transmitted within the neighbourhood, and hence the number of *sync* packets received by a sensor node within the N_{SP} period increases, reducing *sync* inter-arrival times. For WSNs with different densities, there could be a wide variation of *sync* inter-arrival times among the sensor nodes. However, sensor nodes in a WSN typically have similar clock drifts and require similar synchronisation time intervals. If N_{SP} is selected to ensure proper synchronisation in low density networks, then the sensor nodes in high density networks will receive more *sync* packets than necessary and thus more energy is consumed unnecessarily. In fact, for most of the sensor network setups, node densities vary across the entire network and F-Sync will not be effective for all neighbourhoods. Multi-neighbourhood network performance will be studied in Chapter 5.

4.2.3 The Operation of INS

In F-Sync networks, each node wakes up in every SYNC window. When there is no *sync* packet transmission in the network, which is quite often the case in a sparse network, these nodes will just be idling and consuming energy. Intelligent Network Synchronisation (INS) [117] attempts to improve energy efficiency in the synchronisation process by exploiting the periodic nature of *sync* packet transmission in F-Sync.

In INS networks, each node maintains a counter for each of its neighbours. Each counter is increased by one after every cycle. When a node receives a *sync* packet from its neighbour, the corresponding counter will be reset to zero. By examining the list of its counters, the node is able to determine whether there will be a *sync* packet arriving in the current SYNC window. If any of the counter values is greater than or equal to N_{SP}, the node wakes up in the current window as it is expecting a *sync* packet to arrive. It will otherwise go to sleep to conserve energy. INS was simulated and evaluated over a linear network and a sparse grid network with good energy performances. However, in a dense network where collisions frequently occur, the periodicity of *sync* from each neighbour cannot be guaranteed and therefore INS faces the same energy inefficiencies as F-Sync in high density neighbourhoods.

4.3 Proposed Energy Efficient Synchronisation Algorithm

Similar to both F-Sync and INS, the proposed 1-Sync integrates well with synchronous duty-cycle MAC with SYNC, DATA and sleep periods. The algorithms operate in the SYNC periods of the MAC protocols which are independent of the DATA and the sleep periods. The optimization of energy performance of the synchronisation algorithm thus provides a new and added dimension of overall energy performance of the MAC protocols it integrates with without compromising their delay and throughput performance.

The objective of the proposed 1-Sync algorithm is to reduce the energy consumption of the sensor nodes in SYNC windows by allowing them to go to sleep during these windows as much as possible. By design, N_{SP} is chosen such that the sensor nodes need to receive just one *sync* packet within the time period T_{SP}, known as the synchronisation period, for synchronisation regardless of node density. Any other *sync* packets received in this period could be discarded. However, in some networks, multiple schedules could exist in the different neighbourhoods of border nodes. For these border nodes, *sync* packets are also used to maintain multiple schedules. Border nodes with multiple schedules spend more time listening and sending data than other nodes and are therefore highly energy inefficient. To eliminate multiple schedules, *Global Schedule Algorithm* (GSA) is proposed in [103]. Experimental results in [103] and our simulations have both shown that the nodes converge to a single schedule very quickly within a few listening periods.

1-Sync is activated after the schedule has converged using GSA. Similar to both F-Sync and INS, a sensor node using the 1-Sync algorithm sends out *sync* packets at a regular interval T_{SP} (synchronisation period), which can be obtained as:

$$T_{SP} = \frac{(n_{SW} + n_{DW})t_{TS}}{D} N_{SP}, \qquad (4.1)$$

where n_{SW} and n_{DW} denote the lengths of SYNC and DATA window in terms of timeslots, t_{TS} is the duration of a timeslot, and D is the duty cycle.

After a *sync* packet is sent, the sensor node stays awake during the subsequent SYNC windows and waits for a valid *sync* packet from its neighbours. Once a valid *sync* packet is received, the node goes into a synchronised state and will go to sleep in subsequent SYNC windows. The sensor node will only turn its radio on when it is ready to transmit its *sync* packet and the cycle repeats. **Fig. 4.3** illustrates the proposed 1-Sync algorithm.

1-Sync Algorithm

```
initialization:
    nextTxSync = $N_{SP}$
    state = unsynchronised
sync window starts:
    if (nextTxSync != 0 and state == synchronised) {
        nextTxSync --          // not time to send sync, continue sleeping
    }
    else {
        wakeup()
        if (nextTxSync != 0) {
            nextTxSync --
        }
        else {
            send_sync()                // procedure to send sync packet
            if (send_sync_successful) {
                nextTxSync = $N_{SP}$    // schedule next sync transmission
                state = unsynchronised
            }
        }
        if (sync_ received) {
            synchronise_node()       // procedure to synchronise clock
            state = synchronised
        }
    }
sync window ends:
```

Fig. 4.3 1-sync algorithm puts sensor node to sleep after receiving a valid *sync* packet

The key improvement of 1-Sync over F-Sync and INS is that it is able to sleep in the SYNC windows to conserve energy after a sensor node receives its first valid *sync* packet within the synchronisation period. This is especially important in high density neighbourhoods where both F-Sync and INS nodes are fully active, and are hence likely to receive large numbers of corrupted *sync* packets due to collisions.

4.4 Performance Metrics and Analysis

4.4.1 Performance Metrics

In this chapter, we study the energy and synchronisation performance of the synchronisation algorithms in single hop neighbourhoods. Data performance in multi-hop neighbourhoods will be studied in the next chapter.

i. **Average Node Energy Consumption (ANEC):** Node energy consumption is a key performance parameter in duty-cycle WSN. ANEC is computed by dividing the total energy consumed by all the nodes in the network by the total simulation time and the number of nodes.

ii. ***Sync* Packet Inter-arrival Time:** Inter-arrival time between consecutive valid *sync* packets received is another important performance parameter for synchronisation. The longer the inter-arrival time between two consecutive *sync* packets, the worse the phase offset is and the higher the chances of the nodes getting out of synchronisation. The tolerance of the phase offset of a wireless sensor node is dependent on various factors including the protocol it runs, the data rate of the network, and the clock drift specifications of the sensor node. Based on the S-MAC testbed measurements in [92], the synchronisation update period can be in the order of tens of seconds.

iii. **Average Waiting Period for *Sync* Packet Transmission (AWPST):** In high density neighbourhoods, it is common to have multiple *sync* packets scheduled in the same SYNC window and some of them will not have the chance to be transmitted immediately. As such, they will be postponed until the next window. AWPST is the number of frames a node needs to wait from the time a *sync* packet is scheduled to the time it is transmitted. A higher AWPST means that the sensor nodes have less sleep time in SYNC windows and hence have higher energy consumption for the synchronisation process.

4.4.2 Energy Consumption Analysis with Low Neighbourhood Density (N < N_{SP})

In this section, the energy consumption of the sensor nodes in single hop, unsaturated neighbourhoods for F-Sync, INS and 1-Sync are analysed. Analysis on saturated neighbourhoods is provided in the next section.

The operation cycle of a duty-cycle MAC protocol can be divided into three distinct intervals, namely the SYNC window, the DATA window and the sleep period. The total energy consumption for each node, E_{total}, is simply the sum of energy consumptions in the three sub-intervals as follows:

$$E_{total} = E_{SW} + E_{DW} + E_{SLP}, \tag{4.2}$$

where E_{SW}, E_{DW}, and E_{SLP} denote the total energy consumed in SYNC windows, DATA windows and sleep periods respectively.

In the SYNC windows, each node transmits one *sync* packet per synchronisation period T_{SP}. As node density increases, the total number of *sync* packets to be transmitted within a synchronisation period increases. The number of frames in one synchronisation period, N_{SP}, can be considered as the saturation point for the node density N, above which there are more *sync* packets to be sent than the number of SYNC windows available in one synchronisation period. This saturation effect will increase the *sync* packet transmission interval and the probability of a *sync* packet collision. In this section, we focus on energy consumption in the case of low density neighbourhoods ($N < N_{SP}$).

Fig. 4.4 illustrates the *sync* packet transmission scenario in a low density N-node neighbourhood. Each SYNC window allows only a single *sync* packet to be transmitted. As the number of SYNC windows in one synchronisation period N_{SP} is greater than the node density N, and each node only transmits one *sync* packet per synchronisation period; each node will be able to schedule and transmit its *sync* packet in its respective SYNC window in the steady state with no collision.

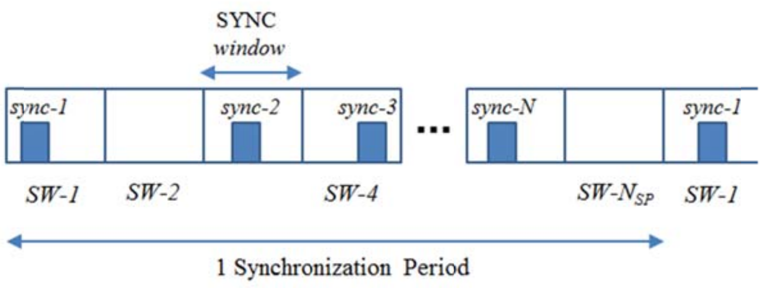


Fig. 4.4 Illustration of *sync* packet transmissions in SYNC windows for a low density$N$-node neighbourhood ($N < N_{SP}$).

4.4.2.1 Energy Consumption in SYNC Windows

For the F-Sync algorithm, each node transmits one *sync* packet and receives (N–1) *sync* packets for every N_{SP} SYNC windows in a single synchronisation period T_{SP}. The duration of a *sync* packet is n_{sync} timeslots, during which the powers P_{tx} and P_{rx} are consumed for transmitting and receiving the packets respectively. For the remaining ($n_{SW} - n_{sync}$) timeslots in the N transmitting and receiving SYNC windows, as well as the remaining ($N_{SP} - N$) SYNC windows when the channel is idle, the node's radio remains active with a power consumption of P_{idle}. The average node energy consumption in the SYNC windows, $\overline{E_{SW}(F\text{-}Sync)}$, in total operation time T can thus be obtained as:

$$\begin{aligned}\overline{E_{SW}(F\text{-}Sync)} = [\ & n_{sync}P_{tx} + (N-1)\,n_{sync}P_{rx} \\ & + \left(N_{SP}n_{SW} - Nn_{sync}\right)P_{idle}\ ]\,t_{TS}\,\tfrac{T}{T_{SP}}.\end{aligned} \tag{4.3}$$

Similarly, for the INS algorithm, each node transmits one *sync* packet and receives (N–1) *sync* packets from its neighbours in T_{SP} as in the F-Sync algorithm. The only difference is that it goes to sleep during the idle SYNC windows. The average node energy consumption in the SYNC windows, $\overline{E_{SW}(INS)}$, in total operation time T can thus be obtained as:

$$\begin{aligned}\overline{E_{SW}(INS)} = [\ & n_{sync}P_{tx} + (N-1)\,n_{sync}P_{rx} \\ & + N\left(n_{SW} - n_{sync}\right)P_{idle} + (N_{SP} - N)n_{SW}P_{slp}\ ]\,t_{TS}\,\tfrac{T}{T_{SP}},\end{aligned} \tag{4.4}$$

where P_{slp} denotes the power consumption in the sleep state.

For the 1-Sync algorithm, each node transmits one *sync* packet and receives one *sync* packet within the time interval T_{SP}. The transmission and the reception of *sync* packets take place in two different SYNC windows. On average, each node stays idle for $\frac{(N_{SP}-N)}{N}$ SYNC windows before receiving a *sync* packet. It will be in the sleep state in the other $[N_{SP}-2-\frac{(N_{SP}-N)}{N}]$ SYNC windows. The average energy consumption in the SYNC windows $\overline{E_{SW}(\mathit{1\text{-}Sync})}$ in time T can be obtained as:

$$\begin{aligned}\overline{E_{SW}(\mathit{1\text{-}Sync})} = \{ & n_{sync}P_{tx} + n_{sync}P_{rx} + 2(n_{SW} - n_{sync})P_{idle} \\ & + \tfrac{(N_{SP}-N)}{N} n_{SW} P_{idle} + \left[N_{SP} - 2 - \tfrac{(N_{SP}-N)}{N}\right] n_{SW} P_{slp} \; \} \\ & t_{TS} \tfrac{T}{T_{SP}}.\end{aligned} \tag{4.5}$$

4.4.2.2 Energy Consumption in DATA Windows

Node energy consumption in DATA windows is dependent on its data load. In this chapter, the focus is on the comparison of energy consumptions attributed to the different synchronisation algorithms, therefore we consider only scenarios where there is no data traffic. With no data traffic, sensor nodes remain idle throughout DATA windows. Energy consumptions in DATA windows for all three synchronisation algorithms are the same and can be expressed as:

$$E_{DW} = N_{SP} n_{DW} P_{idle} t_{TS} \tfrac{T}{T_{SP}}, \tag{4.6}$$

where n_{DW} denotes the length of DATA window in terms of timeslots.

4.4.2.3 Energy Consumption in Sleep Periods

The length of a sleep period, n_{SLP}, is determined by the selection of the duty-cycle D as:

$$n_{SLP} = \tfrac{(1-D)}{D}(n_{SW} + n_{DW}). \tag{4.7}$$

The smaller the duty-cycle, the longer is the sleep period compared to the listen period. Energy consumption in sleep periods is the same for all three synchronisation algorithms and can be expressed as:

$$\begin{aligned} E_{SLP} &= N_{SP} n_{SLP} P_{slp}\, t_{TS} \tfrac{T}{T_{SP}} \\ &= (1-D) P_{slp}\, T\,. \end{aligned} \tag{4.8}$$

4.4.3 Energy Consumption Analysis with High Neighbourhood Density (N > *N_{SP}*)

Consider an N-node single-hop neighbourhood in the absence of hidden or exposed terminals. When the density is low ($N < N_{SP}$), there are more SYNC windows available than the number of nodes in the neighbourhood. There could be some collisions in the initial periods when the nodes schedule their *syn*c transmission in the same window. However, in the steady state, each node eventually settles into its own unique window for periodic *sync* packet transmissions with no collision.

In a high density neighbourhood i.e. $N > N_{SP}$, the number of *sync* packets scheduled to be transmitted within the synchronisation period is more than the number of SYNC windows available. Consequently, one of the following three scenarios may occur when a sensor node is trying to broadcast a *sync* packet:

i. Only one *sync* packet is scheduled in the current window and it is transmitted successfully.

ii. Two or more nodes transmit their *sync* packets in the same timeslot in the current window, resulting in a *sync* packet collision.

iii. The *sync* packets are scheduled in different timeslots, and those scheduled in later timeslots will postpone their *sync* packet transmissions to the next SYNC window upon detection of the first *sync* packet transmission.

The last two scenarios together affect the frequency of *sync* packet transmission and the energy consumption in high density neighbourhoods.

Sync packet transmission scenario in a single saturation neighbourhood from the perspective of a sensor node is illustrated in **Fig. 4.5**. The window with successful *sync* packet transmission is denoted as **T**. In between two *sync* packet transmissions, the sensor node could receive either one valid *sync* packet (denoted as **R**), or corrupted *sync* packets (denoted as **Co**) due to multiple transmissions in each SYNC window. In the case of 1-Sync, the senor node goes to sleep (denoted as **S**) after it receives the first valid *sync* packet and wakes up only when its *sync* packet is scheduled for transmission. Although the sensor node schedules its *sync* packet N_{SP} frames after the previous successful *sync* transmission, the sensor node is only able to transmit its *sync* packet in a longer period (N'_{SP} frames) due to the congestion.

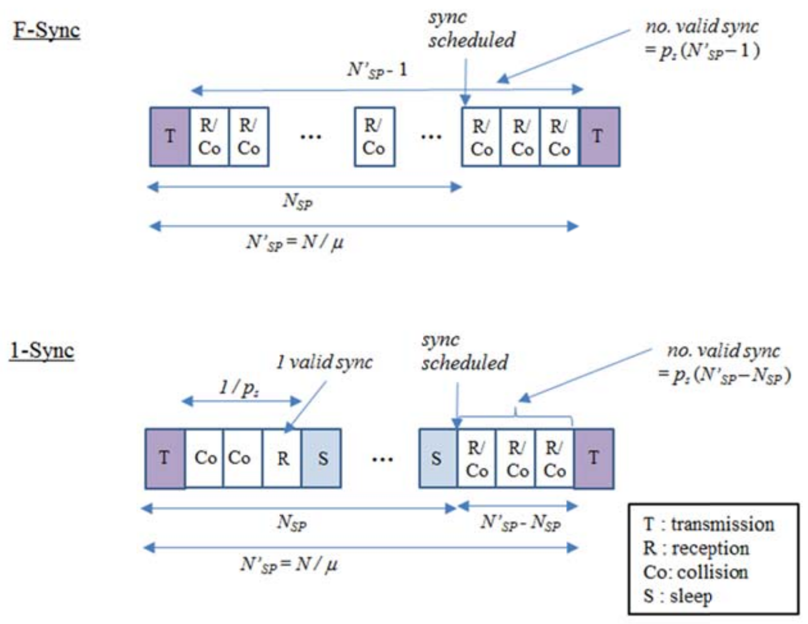


Fig. 4.5 *Sync* transmission analysis for single, saturation neighbourhood

4.4.3.1 *Sync* Packet Transmission and Collision

The Carrier Sense Multiple Access (CSMA) scheme is adopted for the broadcast of *sync* packets. When a *sync* packet is scheduled, the sensor node starts a uniformly distributed carrier-sense (CS) timer within a contention window of n_{CW} timeslots (TS). Upon expiry of this timer, it broadcasts its *sync* packet if no other node is transmitting. Otherwise, the CS timer is cancelled and the transmission attempt is postponed to the next frame. In the case when two or more nodes are transmitting concurrently, *sync* packet collision occurs and the transmitted *sync* packets are corrupted and become invalid. As all the three algorithms have the same *sync* packet transmission behaviour, we do not expect their *sync* packet collision probabilities to be different.

Let U_j and V_j denote the number of *sync* packet transmissions scheduled on the j^{th} contention timeslot and before the j^{th} contention timeslot respectively. The probabilities of successful *sync* packet transmission p_s and collision p_c in the neighbourhood are given as:

$$p_s = \sum_{j=1}^{n_{CW}} p_{s,j} = \sum_{j=1}^{n_{CW}} P(V_j = 0)\, P(U_j = 1 \mid V_j = 0), \tag{4.9}$$

$$p_c = \sum_{j=1}^{n_{CW}} p_{c,j} = \sum_{j=1}^{n_{CW}} P(V_j = 0)\, P(U_j > 1 \mid V_j = 0), \tag{4.10}$$

where $p_{s,j}$ and $p_{c,j}$ are the successful *sync* packet transmission and collision probabilities in timeslot j respectively.

The probabilities of the two events U_j and V_j are dependent on the number of contending nodes, N_c, that have their *sync* packets scheduled in the current window and can be computed as:

$$P(V_j = 0) = \left(1 - \frac{j-1}{n_{CW}}\right)^{N_c} = \left(\frac{n_{CW} - j + 1}{n_{CW}}\right)^{N_c}, \tag{4.11}$$

$$\begin{aligned} P(U_j > 1 \mid V_j = 0) &= \sum_{k=2}^{N_c} P(U_j = k \mid V_j = 0) \\ &= \sum_{k=2}^{N_c} \binom{N_c}{k} \left(\frac{1}{n_{CW} - j + 1}\right)^k \left(\frac{n_{CW} - j}{n_{CW} - j + 1}\right)^{N_c - k}. \end{aligned} \tag{4.12}$$

The average number of contending nodes in the steady state is in turn dependent on the mean number of *sync* packet transmissions in a SYNC window μ_U, which can be seen as the expected number of *sync* packets scheduled on timeslot j with no *sync* packet scheduled before that. The mean number μ_U can be computed as:

$$\mu_U = \sum_{k=0}^{N_c} k \sum_{j=1}^{n_{CW}} P(V_j = 0)\, P(U_j = k \mid V_j = 0). \tag{4.13}$$

In a period of N_{SP} SYNC windows, there are $\mu_U N_{SP}$ *sync* packets transmitted on average; $(N - \mu_U N_{SP})$ *sync* packets are therefore postponed to the next SYNC window for transmission. In the next SYNC window, there will be an average of μ_U new *sync* packets joining the contention since these nodes have successfully transmitted their *sync* packets in this window in the previous synchronisation period. Therefore N_c can be computed as:

$$\begin{aligned} N_c &= N - \mu_U N_{sp} + \mu_U \\ &= N - \mu_U (N_{sp} - 1). \end{aligned} \tag{4.14}$$

4.4.3.2 Energy Consumption in SYNC Windows

While the energy consumption behaviour in the DATA and sleep windows remain unchanged in high density neighbourhoods, the occurrences of *sync* packet collision and postponement affect the frequency of *sync* packet transmission and therefore node energy consumption in SYNC windows. The average number of SYNC windows elapsed for all *N* nodes to transmit one *sync* packet each, N'_{SP} can be obtained as:

$$N'_{SP} = N/\mu_U\,. \tag{4.15}$$

The corresponding time interval, which is referred to as the effective synchronisation period, T'_{SP}, can be obtained as:

$$T'_{SP} = \frac{(n_{SW} + n_{DW})t_{TS}}{D} N'_{SP}\,. \tag{4.16}$$

As there are *sync* packets in practically all SYNC windows in high density neighbourhoods, there is no opportunity for INS nodes to go to sleep in these windows. Therefore, INS nodes display the same energy consumption behaviour as F-Sync nodes. In both cases, the average node energy consumption for high density neighbourhoods $\overline{E'_{SW}(F\text{-}Sync)}$ and $\overline{E'_{SW}(INS)}$ are the same and can be obtained as:

$$\begin{aligned} \overline{E'_{SW}(F\text{-}Sync)} &= \overline{E'_{SW}(INS)} \\ &= \big[\; n_{sync}P_{tx} + \left(N'_{SP} - 1\right) n_{sync}P_{rx} \\ &\quad + N'_{SP}\left(n_{SW} - n_{sync}\right)P_{idle} \;\big]\, t_{TS}\,\tfrac{T}{T'_{SP}} \end{aligned}\,. \tag{4.17}$$

For the 1-Sync algorithm, each node wakes up every N_{SP} period to schedule a *sync* packet transmission but on average only transmits successfully every N'_{SP} period. The average waiting period before successful transmission is ($N'_{SP} - N_{SP}$) windows. After its *sync* packet transmission, it will wait for a valid *sync* packet before it goes to sleep.

Since the probability of successful *sync* packet transmission in the neighbourhood is p_s, the average waiting time to receive a valid *sync* packet is $1/p_s$ frames. Since there are *sync* packet transmissions effectively in every frame in the saturated neighbourhood, each node will receive ($N'_{SP} - N_{SP} + 1/p_s$) collision-free and corrupted *sync* packets. The average energy consumed, $\overline{E'_{SW}(1\text{-}Sync)}$, can thus be obtained as:

$$\begin{aligned}\overline{E'_{SW}(1\text{-}Sync)} = [\ & n_{sync}P_{tx} + (N'_{SP} - N_{SP} + 1/p_s)n_{sync}P_{rx} \\ & + (N'_{SP} - N_{SP} + 1 + 1/p_s)(n_{SW} - n_{sync})P_{idle} \\ & + (N_{SP} - 1 - 1/p_s)n_{SW}P_{slp}\ ]\, t_{TS} \tfrac{T}{T'_{SP}}.\end{aligned} \tag{4.18}$$

4.4.4 *Sync* Packet Inter-arrival Time

In low density neighbourhoods, each node in an F-Sync or INS network receives ($N - 1$) *sync* packets in one synchronisation period of T_{SP} seconds. There is no *sync* packet collision in the steady-state and the *sync* packets received are all valid, therefore the average inter-arrival times, $\overline{W(F\text{-}Sync)}$ and $\overline{W(INS)}$, between two consecutive *sync* packets can be obtained as:

$$\overline{W(F\text{-}Sync)} = \overline{W(INS)} = \frac{T_{SP}}{(N-1)}. \tag{4.19}$$

In the case of 1-Sync, each node receives only one *sync* packet per synchronisation period and the inter-arrival time $\overline{W(1\text{-}Sync)}$ can be obtained as:

$$\overline{W(1\text{-}Sync)} = T_{SP}. \tag{4.20}$$

In high density neighbourhoods, the effective synchronisation period is T'_{SP} as defined in section 4.4.3.2, and each node in an F-Sync or INS network receives ($N'_{SP} - 1$) *sync* packets in this period (**Fig. 4.5**). However, due to collisions, there are only p_s ($N'_{SP} - 1$) valid *sync* packets on average. Therefore the average *sync* packet inter-arrival times in high density neighbourhoods, $\overline{W'(F\text{-}Sync)}$ and $\overline{W'(INS)}$, can thus be obtained as:

$$\overline{W'(F-Sync)} = \overline{W'(INS)} = \frac{T'_{SP}}{p_s(N'_{SP}-1)}. \tag{4.21}$$

In the case of 1-Sync, as illustrated in **Fig. 4.5**, on average, each node receives ($N'_{SP} - N_{SP} + 1/p_s$) collision-free and corrupted *sync* packets in the period T'_{SP}, out of which, $(N'_{SP} - N_{SP})p_s + 1$ are collision-free. Therefore, the average *sync* packet inter-arrival time for 1-Sync in high density neighbourhoods, $\overline{W'(1\text{-}Sync)}$, can be obtained as:

$$\overline{W'(1-Sync)} = \frac{T_{SP}'}{p_s(N_{SP}'-N_{SP})+1}. \tag{4.22}$$

4.5 Performance Evaluation

In this section, we compare and evaluate the energy and synchronisation performance of the F-Sync, INS and 1-Sync algorithms. The F-Sync and 1-Sync algorithms are simulated in ns-2, and their results are compared with the analytical models. INS algorithm is designed for nodes to sleep only when no *sync* packet is expected, they will always wake up to receive all expected *sync* packets. Therefore, for non-energy related performance parameters such as *sync* packet inter-arrival time and collision probability, there is no difference in INS and F-Sync algorithms by design. The performance results of such parameters from F-Sync simulations can therefore also serve as good indicators of INS performance.

4.5.1 Simulation Setup

An ad hoc network within a grid of 100m x 100m is set up to simulate a single-hop neighbourhood. We have selected S-MAC, which is integrated in ns-2 version 2.35, as the representative protocol for synchronous duty-cycle MAC protocols in our simulation. GSA is used to enable the nodes in the simulated network to converge to a single schedule. In the simulations, active neighbour discovery (once in every 22 frames) is disabled for both algorithms for comparison with and validation of the analytical model. In the case of the 1-Sync algorithm, we still let the network run in default F-Sync mode for 3 T_{SP} to achieve a steady state before the 1-Sync algorithm is activated. Simulations are run over a period of 9050s where the statistics are collected for the 9000s of steady-state period after the initial 50s period.

As the length of the sleep period varies with duty-cycle *D*, and the power consumption for the sleep state P_{slp} is very low for most of the hardware today, we have chosen the value zero for P_{slp} in our comparisons to eliminate the dependency on the choice of duty-cycles. However, both the analytical model and the simulations are able to handle the most general cases. The effect of duty-cycles on synchronisation performance will be investigated in Chapter 5.

For each scenario, 30 independent simulation runs are performed. In the single hop single neighbourhood scenarios, we focus on the study of synchronisation performance. Data delivery is generally not a problem in a 1-hop network and therefore the simulations are without data load. Data performance will be evaluated for multi-hop networks in Chapter 5. Parameters used in the simulations are shown in **Table 4.1**:

Table 4.1 Single-hop Network Simulation Parameters

Parameter	**Value**
grid size	100m x 100m
channel data rate	20 kbps
duty-cycle, *D*	10%
simulation time	9000s
N_{SP}	10 frames
n_{CW}	32
P_{tx}, P_{rx}, P_{idle}, P_{slp}	36 mW, 14 mW, 14 mW, 0 mW
n_{sync}, n_{SW}, n_{DW}	10 TS, 55 TS, 105 TS,
t_{TS}	1 ms
propagation model	two-ray ground reflection
carrier sensing range	550m
transmission range	250m

4.5.2 Simulation Results

The simulations are performed using the F-Sync and 1-Sync algorithms within the SYNC windows of S-MAC protocol with no data load. INS performance is based on the analytical models developed in the previous sections.

4.5.2.1 *Sync* Packet Collision Probability

Fig. 4.6 shows the results of *sync* packet collision fractions for the algorithms. From the analysis, all three algorithms have the same *sync* packet collision probabilities since they have the same algorithm for *sync* packet transmission. However, only F-Sync collision statistics are collected in the simulations because 1-Sync nodes could be in the sleep state when some of the collisions occur, and therefore they would not be able to sense all the collisions that occurred.

In the low density region, we do not expect any *sync* packet collisions in the steady state, which is also observed in the simulations. In the high density region, simulation results show that the probabilities of *sync* packet collision increase quite rapidly with increasing node densities, and agree well with the analytical model that we have developed. The *Sync* packet collision probability reaches almost 16% at N=20. This high collision rate is undesirable and impacts the synchronisation performance.

It is interesting to note that when clock drifts of 40 ppm (parts-per-million) and 80 ppm are introduced in the simulations, *sync* packet collision fractions drop drastically to almost zero. In a single neighbourhood with no hidden nodes, collisions occur when multiple *sync* packets are transmitted simultaneously in the same timeslot. Due to random clock drifts in different nodes, *sync* packet transmission from a node with a faster clock will be detected by nodes with slower clocks through carrier sensing even when the *sync* packets are scheduled to be transmitted in the same timeslot. Nodes with slower clocks will then postpone their *sync* packet transmissions, if any, to the next SYNC window; thereby preventing a collision from occurring. It is noted that the reduction of collisions only occurs in the SYNC windows of a single neighbourhood network for the broadcast *sync* traffic. For unicast data packets, the presence of RTS/CTS and retransmission mechanisms has a much greater effect on the data delivery. Data performance in multi-hop networks will be discussed in Chapter 5.

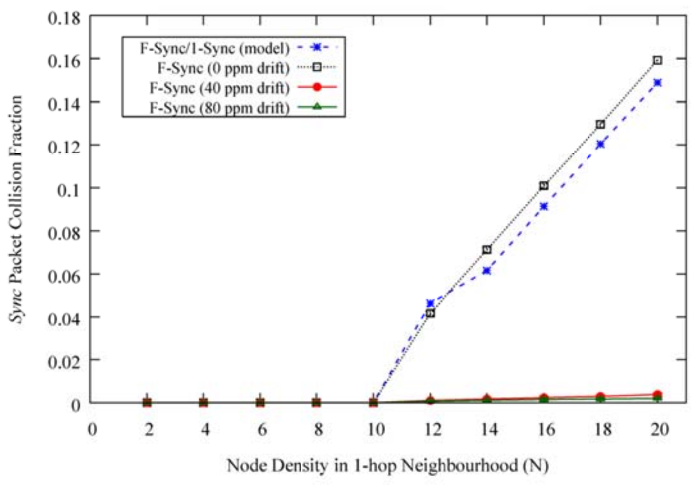


Fig. 4.6 Modelled and simulated *sync* packet collision probabilities for different clock drifts

4.5.2.2 Average Node Energy Consumption

Fig. 4.7 shows the analytical and simulation results of the average energy consumption per node over different node densities within a 1-hop neighbourhood. From the graphs, the simulation results agree very well with the analytical results.

At low density region (N < 10), it can be seen that node energy consumption for F-Sync remains constant even though each node receives more SYNC packets as node density increases. This is because the energy model used has the same idling and receiving power. For 1-Sync, the average node energy consumption decreases substantially as node density increases up to the saturation density N_{SP}=10. This is because when the node density is higher, each node spends less time waiting for a valid *sync* packet before it goes to sleep and hence has a longer sleep time in the subsequent SYNC windows. Energy savings based on simulations range from 14% at N=2 to 27% at N=10.

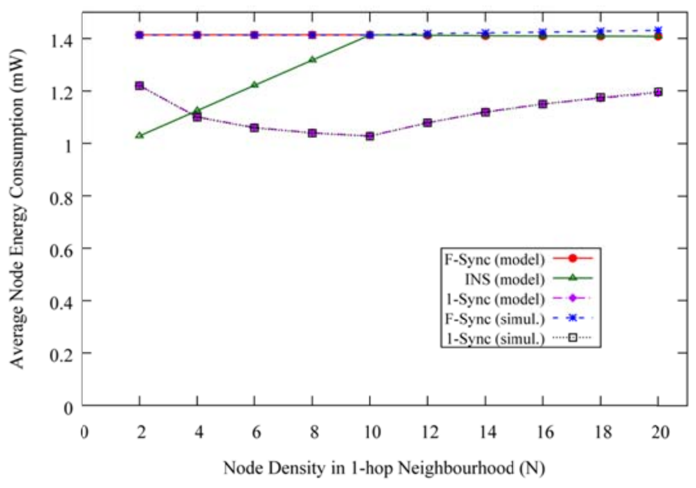


Fig. 4.7 Average node energy consumptions in F-Sync, INS and 1-Sync networks (up to 20 nodes)

As node density increases beyond the saturation point of N=10, heavy *sync* packet traffic causes each 1-Sync node to stay awake for a longer time before it has a chance to transmit its *sync* packet, which contributes to the increase in energy consumption gradually. From the simulation results, the savings in average energy consumption drop to 16% at N=20 compared to F-Sync.

The energy performance of INS based on the analytical model is also plotted for comparison. As can be seen from **Fig. 4.7**, 1-Sync outperformed INS for all scenarios where $N\geq4$. As node density increases, energy savings increase until the saturation density N_{SP}=10. Beyond this saturation density, the INS energy consumption model is the same as the F-Sync model. It is worth noting that in an extremely low density network of N=2 (for academic purposes only), INS has a savings of 16% over 1-Sync determined based on analytical computation. In a two-node network, both 1-Sync and INS nodes receive only one *sync* packet in one synchronisation period. However, INS nodes are able to sleep in all but one SYNC window when the *sync* packet arrives, due to the accurate prediction of periodical *sync* packet arrivals. On the other hand, 1-Sync nodes will only sleep in 50% of the SYNC windows on average. For N=4 and other higher density networks, the energy performance of 1-Sync is 2% to 27% better than INS.

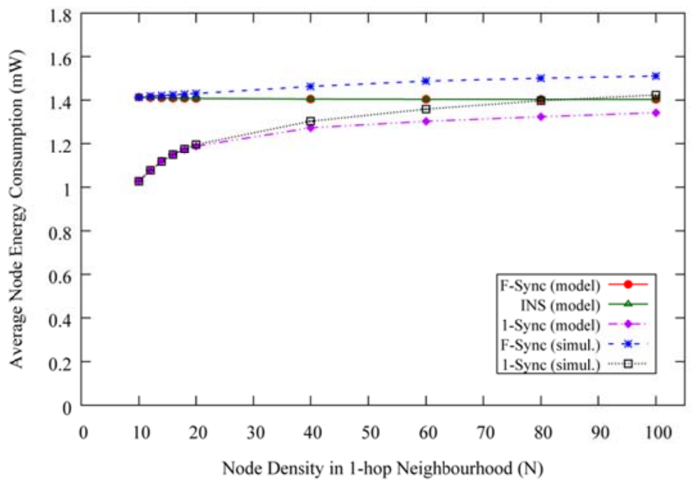


Fig. 4.8 Average node energy consumptions in F-Sync, INS and 1-Sync networks (up to 100 nodes)

To investigate the behaviour of the synchronisation algorithms in the very high density neighbourhoods, simulations of up to 100-node single neighbourhood network were performed. As shown in **Fig. 4.8**, energy consumption for F-Sync and 1-Sync algorithms continues to increase as network density increases and start to deviate from the analytical models which predict the saturation of the energy consumption. This is due to the instability of both algorithms in such high density networks. Further results of high node energy consumption due to algorithm instability will be discussed in Chapter 5.

Fig. 4.9 shows that node energy consumption is not affected by the presence of drift in low density region. In the high density region, the differences are also in a narrow range within 3% in the absence of data traffic.

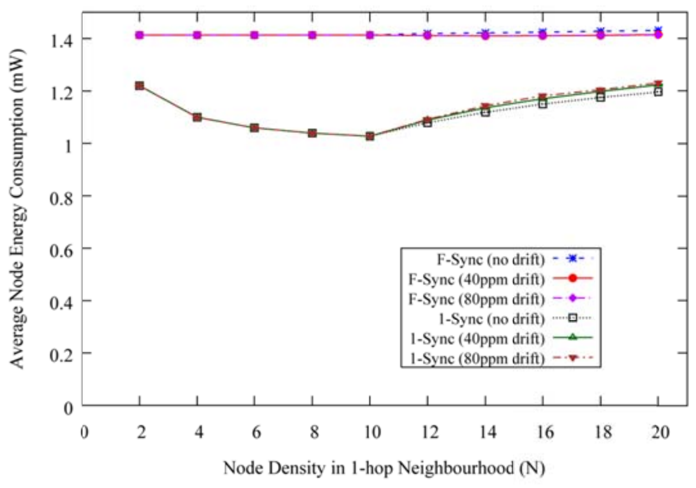


Fig. 4.9 Average node energy consumptions in F-Sync and 1-Sync networks with different clock drifts

4.5.2.3 *Sync* Packet Inter-arrival Time

Fig. 4.10 shows the average and maximum time intervals between two consecutive *sync* packets received by sensor nodes in simulations. Analytical results for average *sync* packet inter-arrival time are also plotted for comparison. From the graphs, the analytical results track the simulation outcome closely throughout the range of node densities simulated.

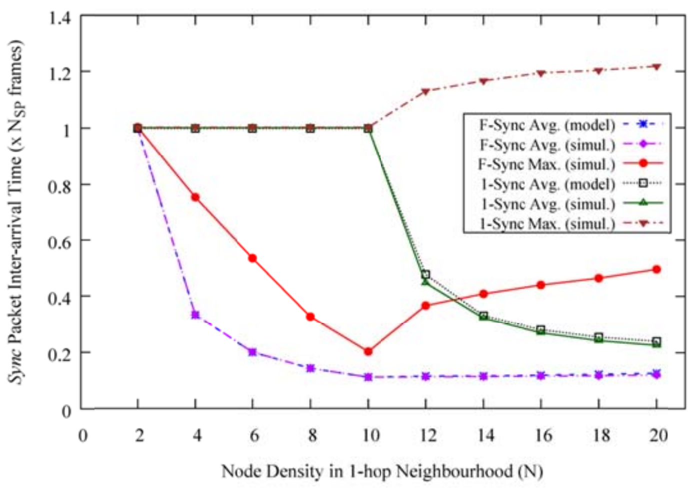


Fig. 4.10 Average and maximum *sync* packet inter-arrival times

In the low density region where $N < N_{SP}$, as node density increases, more *sync* packets are received in a single synchronisation period of N_{SP} frames for an F-Sync network, which results in a substantial reduction in both the maximum and average received inter-arrival times. In the case of 1-Sync, only one *sync* packet is received in a single synchronisation period, which is independent of node density. Both the average and the maximum received intervals are very close to each other and stay at N_{SP} frames. Although F-Sync, which has shorter update intervals, provides better synchronisation performance, 1-Sync provides a consistent level of performance near the desired interval N_{SP} frames.

In the high density region where $N > N_{SP}$, both F-Sync and 1-Sync display substantial variations in their maximum and average received intervals. This is mainly due to the increasing *sync* packet collision rates at higher node densities. On average, both algorithms have shorter *sync* packet received intervals in the high density region than in the low density region. In the worst case comparison considering maximum packet inter-arrival time, both algorithms have the received *sync* intervals that increase gradually with node density.

It is interesting to note that with drifts, maximum *sync* received intervals drop drastically in the high density region in both algorithms as shown in **Fig. 4.11**, which corresponds to the drop in collision rates as shown in **Fig. 4.6**. For the 1-Sync algorithm, the intervals drop to the desirable level of N_{SP} frames.

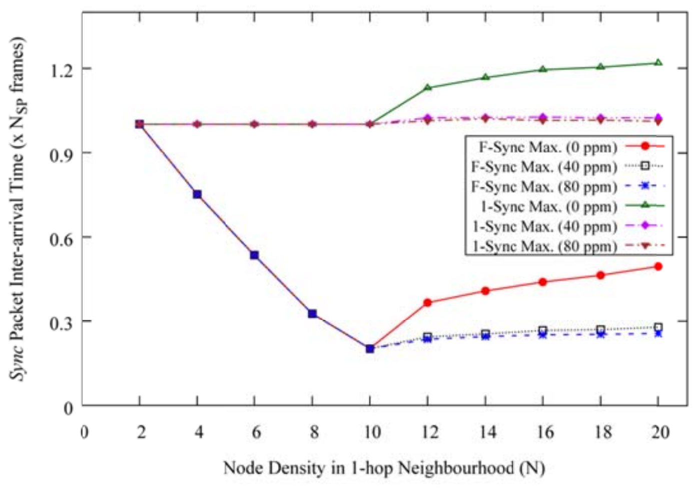


Fig. 4.11 Maximum *sync* packet inter-arrival times with drifts

4.5.2.4 Average Waiting Period for *Sync* Packet Transmission (*AWPST*)

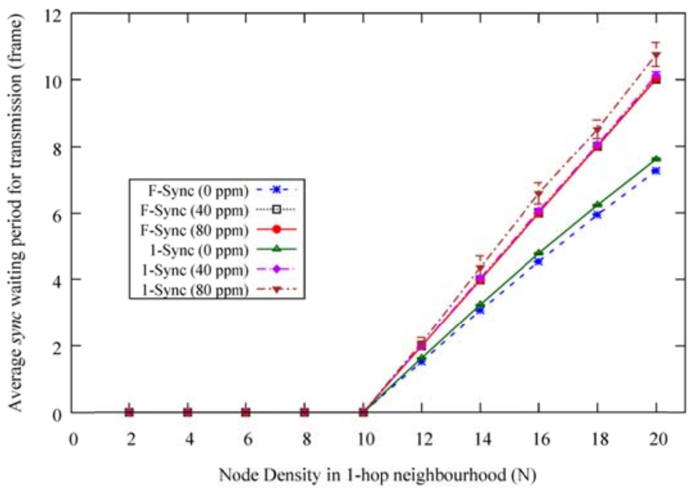


Fig. 4.12 Average waiting period for *sync* packet transmission in F-Sync and 1-Sync networks with different clock drifts

As shown in **Fig. 4.12**, in the low density region where $N < N_{SP}$, both F-Sync and 1-Sync nodes do not have to wait to transmit their *sync* packets. In the steady state, each node settles into their unique *sync* packet transmission window and does not have to contend for the transmission as the number of *sync* packets is fewer than the number of transmission windows.

In the high density region where $N > N_{SP}$, *AWPST* for both algorithms increases rapidly as node density increases. This is because many of the *sync* packets scheduled have to be postponed to later SYNC windows due to the congestion of *sync* traffic in the network. For the 1-Sync algorithm, because the sensors nodes have to stay awake while waiting for their *sync* packet transmissions, an increase in *AWPST* also means an increase in energy consumption (**Fig. 4.9**).

There is little difference in *AWPST* performance between the two algorithms as expected since both F-Sync and 1-Sync algorithm use the same *sync* transmission algorithm and face the same *sync* congestion problem in high density neighbourhoods. With drifts added, sensor nodes with slower clocks are able to detect *sync* packet transmission of a faster node in the same timeslot, which reduces the probability of collision. However, the postponement of *sync* transmission contributes to a higher *AWPST*.

4.6 Chapter Summary

This chapter presents a new synchronisation algorithm, 1-Sync, to improve the energy performance of synchronous duty-cycle MAC protocols. Node energy consumptions for single neighbourhood networks using F-Sync, INS, and 1-Sync are modelled and analysed against different node densities. From the analysis and simulations, the proposed 1-Sync algorithm yields better energy performance than the F-Sync algorithm in all node densities, and better than the INS algorithm for node densities $N \geq 4$.

In terms of *sync* inter-arrival time performance, simulation results show that 1-Sync provides a consistent interval in a single synchronisation period in the unsaturated region. Although this is higher than F-Sync and INS, the consistency enables network designers to fine tune the synchronisation period based on the hardware clock specifications and network node densities.

However, although the 1-Sync algorithm has better energy performance than F-Sync in high node density regions where $N > N_{SP}$, the increase in energy consumption with increasing network densities due to *sync* packet collision and postponement is not desirable. In addition, the large variation in *sync* packet inter-arrival time could result in nodes drifting out of synchronisation, and affecting the data transfer performance. This leads us to design an improved, adaptive synchronisation algorithm in the next chapter to address these issues.

The analytical energy and synchronisation performance models and simulation results in single neighbourhood networks provide us with valuable insights into the behaviour of the synchronisation algorithms under different density scenarios. In the next chapter, the study of network synchronisation will be extended to multi-hop multi-neighbourhood WSNs of different densities, clock drifts, and duty cycles.

Chapter 5

Adaptive Synchronisation Algorithm for Multi-hop WSN

5.1 Introduction

The proposed 1-Sync algorithm in Chapter 4 effectively reduces node energy consumption in both low and high density neighbourhoods. However, it does not solve the problems of *sync* packet congestion and collision in high density neighbourhoods faced by other synchronisation algorithms. *Sync* packet congestion and *sync* packet collision are the main problems of deteriorating energy and synchronisation performance in high density neighbourhoods. These problems will be more pronounced in large WSNs with multiple synchronisation neighbourhoods, in which *sync* packet collisions cannot be eliminated due to the presence of “hidden nodes”.

In this chapter, we present a new, counter-based synchronisation algorithm (C-Sync) that works in the framework of synchronous MAC protocols. The C-Sync algorithm enables a sensor node to adapt its synchronisation mechanisms in different density neighbourhoods. It reduces energy consumption and improves the effectiveness of the synchronisation process by adaptively switching off the radio when not required, as well as reducing unnecessary *sync* packet transmission when the network density is high. It also enables the sensor nodes to wake up more frequently to receive *sync* packets when the network density is low.

In multi-hop WSNs, in addition to energy consumption, the effectiveness of the synchronisation algorithms also affects data performance, including packet delivery ratio and end-to-end packet delay. Extensive simulations are conducted for multi-hop multi-neighbourhood grid networks with various network densities for performance evaluation. Effects of drift and duty-cycling on both the energy and data performance are also studied.

5.2 Duty-cycle MAC in Multi-hop Networks

The design goal of the proposed C-Sync algorithm is to offer energy consumption efficiency while providing good data performance, including packet delivery ratio and end-to-end packet delay, across a wide range of multi-hop WSNs. In this chapter, we will study the performances of the C-Sync algorithm against the F-Sync and 1-Sync algorithms in WSNs with different network densities, clock drifts and duty cycles for multi-hop networks. We will also compare the robustness and stability of these three synchronisation algorithms by examining individual node energy performance under a wide range of network scenarios.

5.2.1 Need for Adaptive Synchronisation Algorithm

An analysis of single-hop neighbourhoods in Chapter 4 indicates that in the saturation region, *sync* packet broadcast traffic increases rapidly as neighbourhood density increases. The increase in *sync* packet traffic increases the probabilities of *sync* packet collision, and *sync* packet postponement.

Both *sync* packet collisions and *sync* packet postponements are undesirable as they increase the time intervals between consecutive *sync* packets received, which lowers the synchronisation performance of the network. *Sync* packet collisions corrupt the *sync* packets transmitted in the neighbourhood, reducing the number of valid *sync* packets and therefore increasing the synchronisation time of the sensor nodes in the neighbourhood. In the case of *sync* packet postponement, due to a *sync* packet transmission in one neighbourhood, sensor nodes scheduled for *sync* packet transmissions have to postpone their transmissions, which could affect synchronisation in other neighbourhoods.

For 1-Sync, both *sync* packet collisions and *sync* packet postponements cause another undesirable effect: an increase in energy consumption. This is because 1-Sync nodes have to stay active while waiting for their *sync* packet transmission and reception, when they can go to sleep otherwise.

To reduce energy consumption, both 1-Sync and INS modify the *sync* packet reception process, enabling sensor nodes to sleep during the SYNC windows when they do not require synchronisation or do not expect *sync* packets to arrive. However they make no changes to the *sync* packet transmission process. As a result, they face the same *sync* packet congestion problem as F-Sync in high density neighbourhoods.

In addition, in a multi-hop sensor network, there are multiple synchronisation neighbourhoods with additional complexities listed as follows:

i. Sensor nodes in different regions of the network are in neighbourhoods of different densities. Inter-arrival times between *sync* packets received by each node can have large variations.

ii. As discussed in Chapter 4, due to the *sync* packet postponement effect, *sync* packet transmission is not periodic in high density neighbourhoods. In a multi-hop multi-neighbourhood network, aperiodic *sync* packet transmission in high density neighbourhoods can affect the periodicity of *sync* packet transmission in low density neighbourhoods due to interactions among different neighbourhoods. Therefore *sync* packet periodicity is not guaranteed even in low density neighbourhoods.

iii. *Sync* packet collision is unavoidable even at unsaturated neighbourhoods due to the “hidden node” or “hidden terminal” problem. The hidden node problem occurs when a node is within the neighbourhood of a receiver, but is not visible to some other nodes in the neighbourhood of the same receiver. As the hidden node is unable to detect the transmissions of the other nodes, a collision will occur at the receiving node when the hidden node has an overlapping *sync* packet transmission with one of the other nodes.

It is therefore desirable to have an adaptive synchronisation algorithm that enables sensor nodes to dynamically adjust their *sync* packet transmissions in the different density neighbourhoods to reduce *sync* packet collision and *sync* packet postponement, and at the same time, maintain an optimum level of synchronisation and energy performance.

5.2.2 *Sync* Scheduling and Broadcast Methods

The overloading of *sync* traffic in high density neighbourhoods is similar to the well-known broadcast storm problem [118]. The key difference is that the former is due to the generation and transmission of new single-hop broadcast packets and the latter is due to the retransmission of the same multi-hop broadcast packets.

The broadcast storm problem is primarily caused by simple *flooding* of broadcast packets to reach all nodes in the network. In a simple *flooding* algorithm, each node tries to forward all unseen broadcast packets it receives to all its neighbours, except the source node. This results in the packets being delivered to all nodes in the network eventually. This algorithm is simple to implement, but it causes serious redundancy, contention and collision problems, especially in dense networks.

As radio signal propagation is omnidirectional, a wireless node within the transmission range of multiple nodes will receive many redundant packets when these nodes rebroadcast the same packet. Heavy contention could also exist because these rebroadcasting nodes are likely to be close to one another. In addition, as the timings of rebroadcasts are highly correlated and RTS/CTS exchange is not applicable for broadcast, collisions are more probable.

Different broadcasting methods are developed to reduce broadcast storms in ad hoc wireless networks [119]. They can be broadly classified into probabilistic-based, area-based, neighbour knowledge and multipoint relay methods.

The key objective of area-based methods is to transfer the packets to the downstream areas and to minimise the repetition of these packets within the same area. Therefore the decision to rebroadcast is based on a valuation of the additional coverage area of the given node. Area-based methods require the nodes to have estimations of distances to, or locations, of the neighbours.

Neighbour knowledge methods evaluate the necessity of rebroadcasting the packets based on whether they can be transmitted to at least one new node. These methods require the nodes to collect information about the neighbours via periodic Hello packets.

Multipoint relay (MPR) methods identify a subset of 1-hop neighbours to be responsible for retransmission of the broadcast packets to all the 2-hop neighbours of the original nodes. These methods are not applicable to single hop broadcasts.

There are two approaches in probability-based methods: the use of a probability value *p* and the use of counter-value *c* for re-transmission decisions. The counter-based approach [120], when adapted to our *sync* packet scheduling, has the desirable characteristic of automatically regulating *sync* packet traffic transmission with different neighbourhood densities.

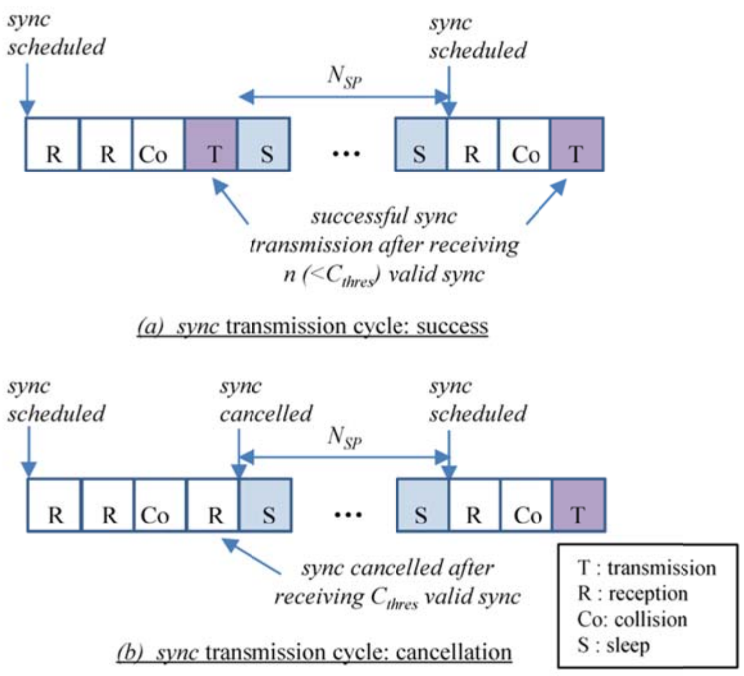
5.3 C-Sync Algorithm Design

The proposed C-Sync algorithm operates in the SYNC windows of a synchronous duty-cycle MAC protocol. There are two sub-algorithms in C-Sync, a counter-based *sync* transmission algorithm that reduces *sync* packet load and energy consumption when the network neighbourhood is dense, and an adaptive exponential-smoothing *sync* reception algorithm that improves sync performance when the network is sparse.

5.3.1 Counter-based Algorithm for *Sync* Transmission

Similar to F-Sync and 1-Sync, a C-Sync node schedules the next *sync* packet N_{SP} frames after a successful transmission of the current *sync* packet. However, when the *sync* transmission is unsuccessful, the F-Sync and 1-Sync algorithms will attempt to transmit the scheduled *sync* packets in every subsequent SYNC window until they are successfully transmitted. In a high density neighbourhood where the number of *sync* packets scheduled is more than the number of SYNC windows available in a synchronisation period, many *sync* packets scheduled will be withheld and congestion remains a problem. C-Sync, on the other hand, provides a mechanism to cancel the scheduled *sync* packets when transmission is unsuccessful, reducing the traffic load when the neighbourhood gets congested.

When a *sync* packet is first scheduled, a C-Sync node initiates a counter c_{sn} to count the number of valid *sync* packets received while waiting for its turn to transmit. The sensor node attempts to transmit its scheduled *sync* packet following a contention procedure. If it is unsuccessful, the counter c_{sn} is incremented by 1 whenever a valid sync packet is received. If the value of c_{sn} is less than the counter threshold C_{thres}, the *sync* packet is postponed and rescheduled to the next SYNC window. Otherwise, the scheduled *sync* packet is cancelled, and the sensor node then goes to sleep and a new *sync* packet will be scheduled N_{SP} frames (one synchronisation period) later. If the *sync* packet transmission is successful, a new *sync* packet will also be scheduled one synchronisation period later. The counter-based *sync* transmission scheme implemented in C-Sync is illustrated in **Fig. 5.1**.



sync scheduled
sync scheduled
N_{SP}
R R Co T S ••• S R Co T
successful sync transmission after receiving n (<C_{thres}) valid sync
(a) sync transmission cycle: success
sync scheduled
sync cancelled
sync scheduled
N_{SP}
R R Co R S ••• S R Co T
sync cancelled after receiving C_{thres} valid sync
(b) sync transmission cycle: cancellation
T : transmission
R : reception
Co: collision
S : sleep

Fig. 5.1 Illustration of C-Sync counter-based *sync* transmission with C_{thres}=3

When neighbourhood density is low, *sync* packet traffic generated is also low and therefore there is a high probability of successful *sync* packet transmission without the need to trigger the *sync* packet cancellation algorithm. This is advantageous as each and every *sync* packet is important for synchronisation, since there are so few of them present in low density neighbourhoods.

When neighbourhood density is high, there are many more *sync* packets in the neighbourhood. There is a high probability for a node to receive C_{thres} *sync* packets from its neighbours before its own successful *sync* packet transmission. This will trigger the algorithm to cancel the scheduled *sync* and allows the node to go to sleep in the subsequent SYNC windows. This process reduces the *sync* packet load in the neighbourhood, shortens the active waiting periods for *sync* packet transmission, and lowers the energy consumption of the sensor nodes.

5.3.2 Exponential-smoothing Algorithm for *Sync* Reception

A key mechanism of energy conservation algorithms for synchronisation is the ability to go to sleep in the SYNC windows as long as clock drift is within the tolerance limit of the synchronisation. As the *sync* packet traffic load varies with neighbourhood density, sensor nodes that wake up to receive a sync packet will have to wait for different time intervals before they receive a valid *sync* packet. The waiting interval tends to be longer when the density is low and this will affect the synchronisation performance.

The proposed C-Sync algorithm enables a sensor node to adjust its wakeup time dynamically to compensate for the waiting time it takes to receive a valid *sync* packet so that it has a higher chance of receiving a valid *sync* packet within a desired number of frames N_{RP} under different density and traffic conditions. A C-Sync node maintains a counter, w_{wk}, the number of SYNC windows it should be sleeping in before waking up to receive *sync* packets. Upon waking up, it stays active in all the subsequent SYNC windows until a valid *sync* packet is received, after which it will go to sleep. The waiting period from the time the node wakes up to the time a valid *sync* packet arrives is denoted as w_a (frames). To maintain the received synchronisation interval close to N_{RP}, w_{wk} should be compensated with w_a as follows:

$$w_{wk} = N_{RP} - w_a \,. \tag{5.1}$$

The waiting interval w_a varies over time and is dependent on both the node density and the collision level in the neighbourhood. Therefore it is necessary to forecast the next waiting time for the computation of the next wake-up interval in order to keep the received synchronisation within a tight range. As the waiting time is not expected to show any trend, a simple exponential smoothing technique is appropriate and used in the adaptive C-Sync *sync* packet reception algorithm. Use of the exponential-smoothing technique is also advantageous as it is both memory- and computational-efficient. The computation for the wake-up interval at the $(k+1)^{th}$ period, $w_{wk}(k+1)$ can be formulated as

$$w_{wk}(k+1) = \lfloor \alpha\{ N_{RP} - w_a(k)\} + (1-\alpha)w_{wk}(k) \rfloor , \tag{5.2}$$

where α is the smoothing factor which can be chosen between zero and one, and the initial value of the wake-up interval, $w_{wk}(0)$, is chosen to be the midpoint value of the desired received interval as

$$w_{wk}(0) = \left\lfloor \tfrac{N_{RP}}{2} \right\rfloor . \tag{5.3}$$

The exponential-smoothing *sync* packet reception scheme implemented in C-Sync is illustrated in **Fig. 5.2**.

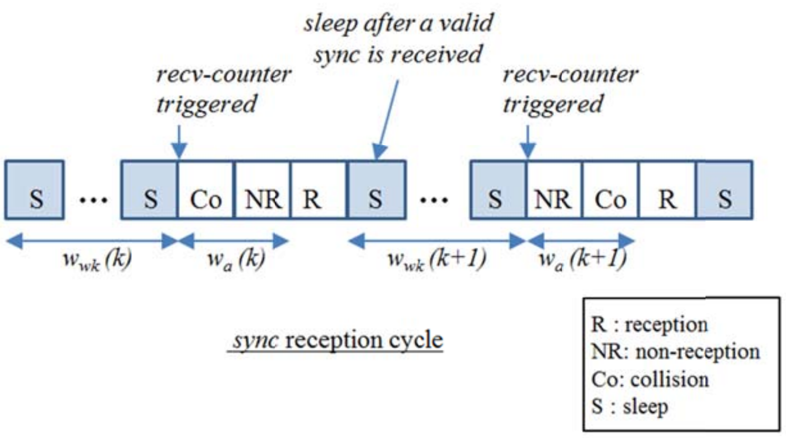


Fig. 5.2 Illustration of C-Sync adaptive *sync* reception cycle

Combining the counter-based *sync* packet transmission and exponential-smoothing *sync* packet reception algorithms, the C-Sync algorithm will be able to modify the sensor nodes' behaviour adaptively in a wide range of network neighbourhood densities. The pseudo-code of the proposed C-Sync algorithm is shown in **Fig. 5.3**.

C-Sync Algorithm

```
initialization:
    syncTxCounter = $N_{SP}$
    nextRxWkUp = int ($N_{RP}$ / 2)
    rxWkUpCounter = nextRxWkUp
    syncWaitCounter = 0
    syncRecvCounter = 0

sync window begins:
    if (syncTxCounter == 0 or rxWkUpCounter == 0) {          // time to wake up
        wakeup()
        if (syncTxCounter == 0) {
            send_sync()                               // procedure to send sync packet
            if (send_success) {
                syncTxCounter = $N_{SP}$
                syncRecvCounter = 0
            }
        }
        if (rxWkUpCounter == 0) {
            if (sync_ received) {
                synchronise_node()                    // procedure to synchronize clock
                nextRxWkUp = int ($\alpha$ * ($N_{RP}$ – syncWaitCounter)
                                        + (1 - $\alpha$) * nextRxWkUp)
                rxWkUpCounter = nextRxWkUp
                syncWaitCounter = 0
                if (syncTxCounter == 0) {
                    syncRecvCounter++                 // count number of sync packets received
                    if (syncRecvCounter == $C_{thres}$) {
                        syncTxCounter = $N_{SP}$      // cancel scheduled sync packet
                        syncRecvCounter = 0
                    }
            }
            else syncWaitCounter ++                   // increment sync waiting period counter
        }
    }
    else {
        if (syncTxCounter != 0) syncTxCounter --      // transmit counter countdown
        if (rxWkUpCounter != 0) rxWkUpCounter --      //receive wakeup counter countdown
sync window ends:
```

Fig. 5.3 Adaptive C-sync algorithm with counter-based *sync* transmission and exponential-smoothing *sync* reception sub-algorithms

5.4 Performance Evaluation

In this section, we evaluate and compare the synchronisation, energy and data performance of C-Sync against F-Sync and 1-Sync using ns-2 version 2.35.

5.4.1 Simulation Setup

Multi-hop grid networks within an area of 500m x 500m are set up for the simulation of different synchronisation scenarios. S-MAC, which is integrated in ns-2, is selected as the representative protocol for the duty-cycle MAC protocols in our simulations.

To represent networks of different densities, we started with a low density network of 9 nodes, forming 3x3 equal size square grids, and steadily increased to a high density of 49-node network forming 7x7 square grids, all in the same 500mx500m simulation area.

Both the synchronisation and data performance of the protocols are evaluated. Data packets will be sent from the lower left corner of the grid to the upper right corner of the grid. To prevent the influence of routing protocols on the network performance, fixed (static) routing with external routing tables is used in the simulations. The use of fixed routing also enables us to control and fix the end-to-end paths at 4 hops from the source to the destination across different density networks so that fair comparisons can be made among different density networks. The end-to-end data paths for the different density grid networks are shown in **Fig. 5.4**.

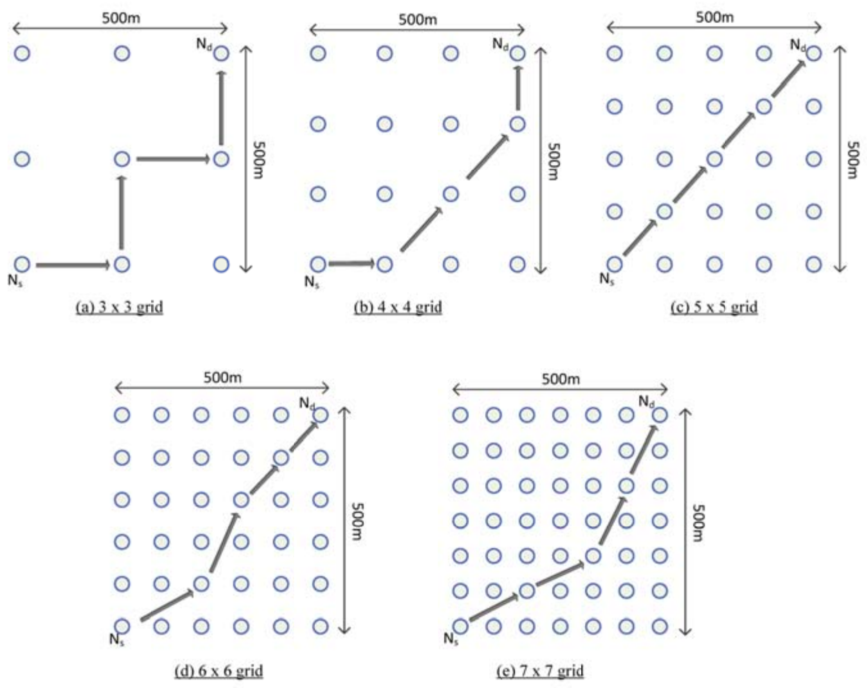


Fig. 5.4 Illustration of 4-hop fixed routing path for different density grid networks

Networks with different drift scenarios are also simulated. For each drift scenario, each sensor node has an independent local clock with a drift rate that is uniformly distributed between $\pm\Delta f$, where Δf ranges from 0 ppm to 80 ppm. Simulations are also performed at four different duty cycles of 20%, 10%, 5% and 2%.

For each scenario, 30 independent simulation runs are performed over a period of 9000s. Constant bit rate (CBR) data traffic at 1 packet per minute is sent from the source node at one corner of the grid to the destination node at the diagonally opposite corner of the grid. The source node starts the data traffic at 100s after the start of the simulations to allow the MAC layer protocol to stabilize, and stops the data traffic 60s before the simulation ends. The size of the data packet and S-MAC protocol data unit (PDU) used are 100-byte and 120-byte respectively so that each data packet can fit into a single PDU without fragmentation.

The key parameters used in the simulations are summarised in **Table 5.1**.

Table 5.1 Grid Network Simulation Parameters

Parameter	Value	Parameter	Value
grid size	500m x 500m	data rate (CBR)	1 pkt / min
bandwidth	20 kbps	packet size	100B
simulation time	9000s	routing	fixed
tx power	36 mW	no. of hops	4
rx power	14 mW	retry limit	5
idle power	14 mW	N_{SP}	10 frames
sleep power	0 mW	N_{RP}	10 frames
propagation	2-ray ground reflection	α	0.5
CS range	550m	C_{thres}	3
tx range	250m		

5.4.2 Performance Metrics

The following two synchronisation performance metrics in the SYNC windows are first evaluated.

5.4.2.1 Average Waiting Period for *Sync* Packet Transmission (*AWPST*)

In high density networks, it is common to have multiple *sync* packets scheduled in the same SYNC window; hence, some of them will not have the chance to be transmitted immediately. Instead, they will be postponed till the next window. *AWPST* is the number of frames a node needs to wait from the time a *sync* packet is scheduled to the time it is transmitted. Higher *AWPST* means that the sensor nodes have less sleep time in SYNC windows and hence have higher energy consumption for the synchronisation process.

5.4.2.2 Fraction of Desired *Sync* Inter-arrival Time (*FDSIT*)

FDSIT is the fraction of *sync* packet received intervals that are smaller than N_{RP}, the desired sync received interval. Higher *FDSIT* means that the probability of sensor nodes getting out of synchronisation is higher and data performance will be affected.

These two metrics are the direct outcomes of the different *sync* packet scheduling, transmission and reception mechanisms in the synchronisation algorithms.

In addition, the energy and data performances of the sensor networks with different synchronisation algorithms can also be evaluated using the following three performance metrics:

5.4.2.3 Packet Delivery Ratio (*PDR*)

PDR is the ratio of the total number of CBR data packets generated at the source node N_s to the total number of data packets received at the destination node N_d. As fixed routing with the same number of hops (4 hops) are used in all the scenarios, a higher value of *PDR* indicates that the data delivery performance of the underlying MAC protocol is better. In our simulations, the same S-MAC data protocol is used in DATA windows, alongside the three synchronisation algorithms in SYNC windows for comparison. Thus, a higher *PDR* values indicates that the synchronisation algorithm used is more reliable, which results in a better performance.

5.4.2.4 Average End-to-end Packet Delay (*AvDelay*)

AvDelay measures the average time taken for a data packet to be successfully delivered from the source node N_s to the destination node N_d. *AvDelay* is computed by summing up the end-to-end delay of each CBR data packet and dividing it by the total number of successfully delivered data packets. The lower the end-to-end delay, the better the application performance.

In a multi-hop network using contention-based MAC protocols, a packet experiences the following delays at each hop [95]:

i. Carrier Sense Delay is introduced when the sensor node performs carrier sense, and is dependent on the size of the contention window.

ii. Back-off delay occurs if the sensor node detects a transmission or collision in the medium during carrier sense.

iii. Transmission delay is determined by channel bandwidth, packet length, as well as coding scheme.

iv. Propagation delay is determined by the distance between the transmitting and receiving nodes. It is negligible in WSN compared to the other components.

v. Processing delay is the time needed to process the packet before forwarding it to the next hop. This delay mainly depends on the computing power of the node.

vi. Queuing delay depends on the traffic load.

The above delays are all present in the contention-based MAC protocols, including the S-MAC data protocol used in the simulations; statistically, they should contribute to the same amount of delay. In duty-cycle MAC protocols, the key differences in the delay performance attributed to the different synchronisation algorithms are sleep delay and retransmission delay. Sleep delay is the time spent in waiting for the receiver to wake up. Retransmission delay occurs when a packet is not correctly received by the receiving node and retransmission is required. In general, if sensor nodes are out of synchronisation, both sleep and retransmission delays will increase.

A lower duty-cycle WSN will have a higher end-to-end packet delay compared to a higher duty-cycle network due to longer sleep periods. To compare *AvDelay* performance across different duty-cycle networks meaningfully, it is normalised by dividing the actual delay in seconds by the amount of time it takes to transmit a *frame*.

5.4.2.5 Average Node Energy Consumption (*ANEC*):

Node energy consumption is a key performance parameter in duty-cycle WSN. *ANEC* is computed by dividing the total energy consumed by all the nodes in the network by the total simulation time and the number of nodes. Although energy performances are compared among the different synchronisation algorithms, node energy consumption values are collected for the entire simulation duration for two reasons. First, ns-2 is a discrete event simulation tool; the computation of energy consumption in ns-2 is based on events (transmit, receive, idle, and sleep), and node energy consumption cannot be accurately obtained based on the S-MAC frame structure (SYNC, DATA, and SLEEP). Second, the performance of synchronisation algorithms affects data performance and hence the energy consumption in DATA windows.

5.4.2.6 Individual Node Energy Consumption

While *ANEC* measures network-wide average energy consumption, it is also important for energy consumption to be evenly distributed among individual nodes in the network. As discussed in Chapter 3, network lifetime will be impacted if some of the sensor nodes consume more energy than the others. These nodes will be depleted faster and will cause network segmentation.

Individual nodes in the simulated grid networks are labelled with node identifiers (Node IDs) based on their positions on the grid. The source node at the lower left corner of the grid is assigned a Node ID of ‘0’ and it is incremented rightward and upward. The destination node at the upper right corner will have the largest Node ID. An example of the Node ID assignment for a 5 x 5 grid network is shown in **Fig. 5.5**.

The energy consumption for each node in the respective position is averaged over the simulation runs. The variation of individual node energy consumptions in the network will be measured using the standard deviation.

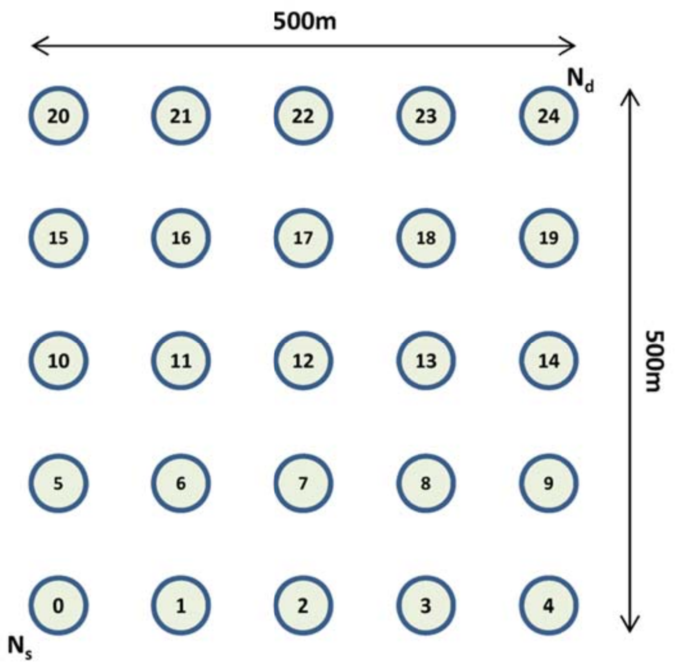


Fig. 5.5 Node IDs configuration for a 5 x 5 grid network

5.5 C-Sync Parameters

The behaviour of the C-Sync algorithm is controlled by two parameters: C_{thres} for the counter-based *sync* packet transmission algorithm, and α for the exponential-smoothing *sync* packet reception algorithm.

5.5.1 Variation of Counter Threshold (C_{thres})

In the multi-hop broadcast scenarios, the choice of counter threshold used in a counter-based algorithm affects the broadcast performance in two ways. When a small counter threshold value is used, there are significantly fewer rebroadcasts. However, the reachability will be sacrificed in a sparse network. Increasing the threshold will increase reachability but also increase the number of rebroadcasts.

For the scenarios of the *sync* broadcast in multi-hop WSNs, the key considerations are synchronisation performance, which translates to data performance, as well as energy consumption performance. Simulations for a subset of network scenarios are performed using different C_{thres} and their performances are examined.

To study the effect of C_{thres}, we vary the values of C_{thres} from 1 to 4 in our network scenarios. Both 10% and 2% duty-cycle networks, with network densities ranging from 3x3 to 7x7 grids are simulated. A midpoint value of 40 ppm clock drift is used throughout. The results of *PDR*, *AvDelay* and energy consumption are shown in **Fig. 5.6**.

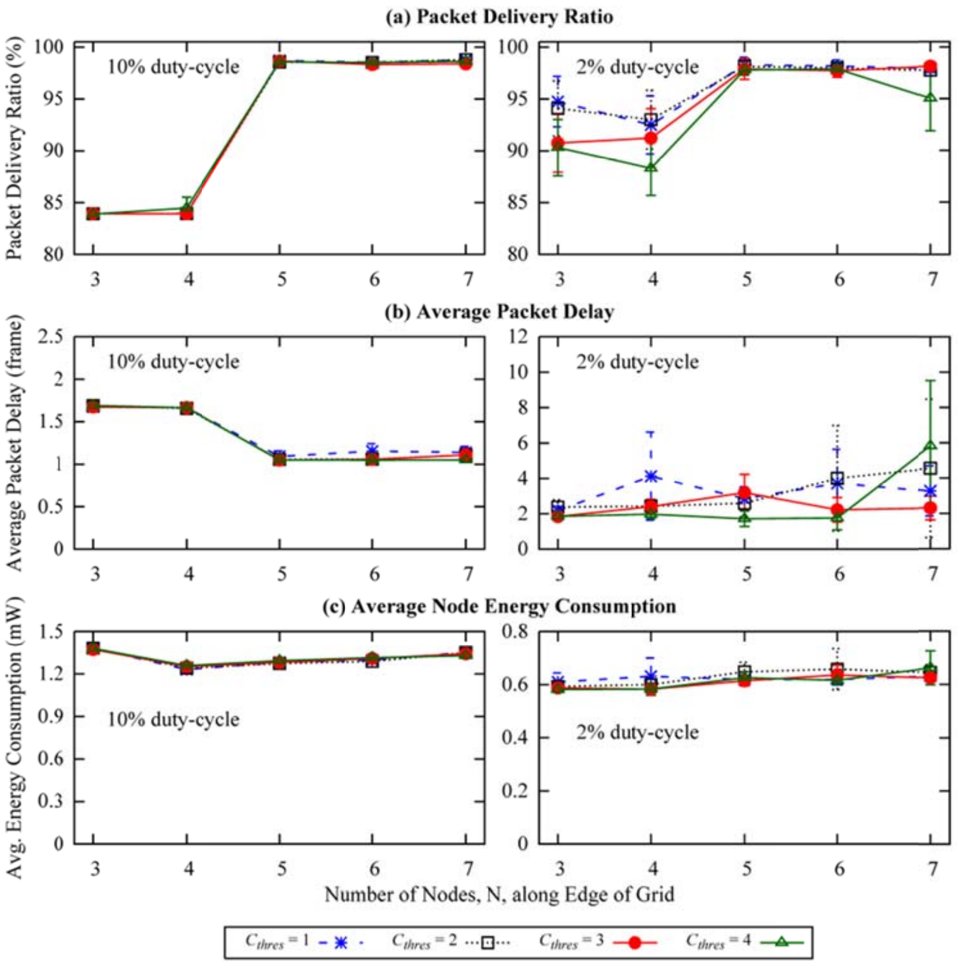


Fig. 5.6 Energy and data performance of C-Sync algorithm with different C_{thres} in grid networks at 40 ppm drift rate

At 10% duty cycle, the differences in all three performance metrics are not significant as shown. The maximum differences are in energy consumption in the 4x4 grid network, which is in the range of 2.2%. At 2% duty cycle, *PDR* performance for the case of C_{thres} = 4 is the lowest among all. It is also the worst performer for all three metrics in high density 7x7 grid networks. For the case of C_{thres} = 1, it has the worst *AvDelay* performance for most of the scenarios and the highest energy consumption in low density networks. Between C_{thres} = 2 and C_{thres} = 3, C_{thres} = 2 has better PDR performance in low density networks, while C_{thres} = 3 has better energy consumption and *AvDelay* performances in most of the scenarios. We will therefore use C_{thres} = 3 in the subsequent simulations for comparison among the different synchronisation algorithms in this work.

5.5.2 Variation of Smoothing Factor (α)

The smoothing factor α in the exponential-smoothing algorithm represents the weighting applied to the most recent data. Values of α that are close to one have less of a smoothing effect and are more responsive to recent changes in the data, while the opposite is true for values of α closer to zero.

To study the effect of α, we use three different values of α at 0.25, 0.50 and 0.75 for the same grid network scenarios used in the previous section. The results of *PDR*, *AvDelay* and energy consumption are shown in **Fig. 5.7**.

The results of the simulations show that each of the three α values has its strength in different scenarios for different metrics. At 10% dc, *PDR* and *AvDelay* for all three α are similar. In terms of energy consumption, the case of α = 0.75 has the best performance in low density 3x3 grid networks while the case of α = 0.25 has the best performance in high density 7x7 grid networks. At 2% dc, the case of α = 0.50 has the best performances in all the three metrics measured in high density 7x7 networks. In 6x6 grid networks, it also has the best performance in energy consumption while maintaining similar performance on *PDR* and *AvDelay* as the other two values of α.

The above results have shown that the selection of C_{thres} = 3 and α = 0.5 has close to optimum performance and will be used for subsequent simulations.

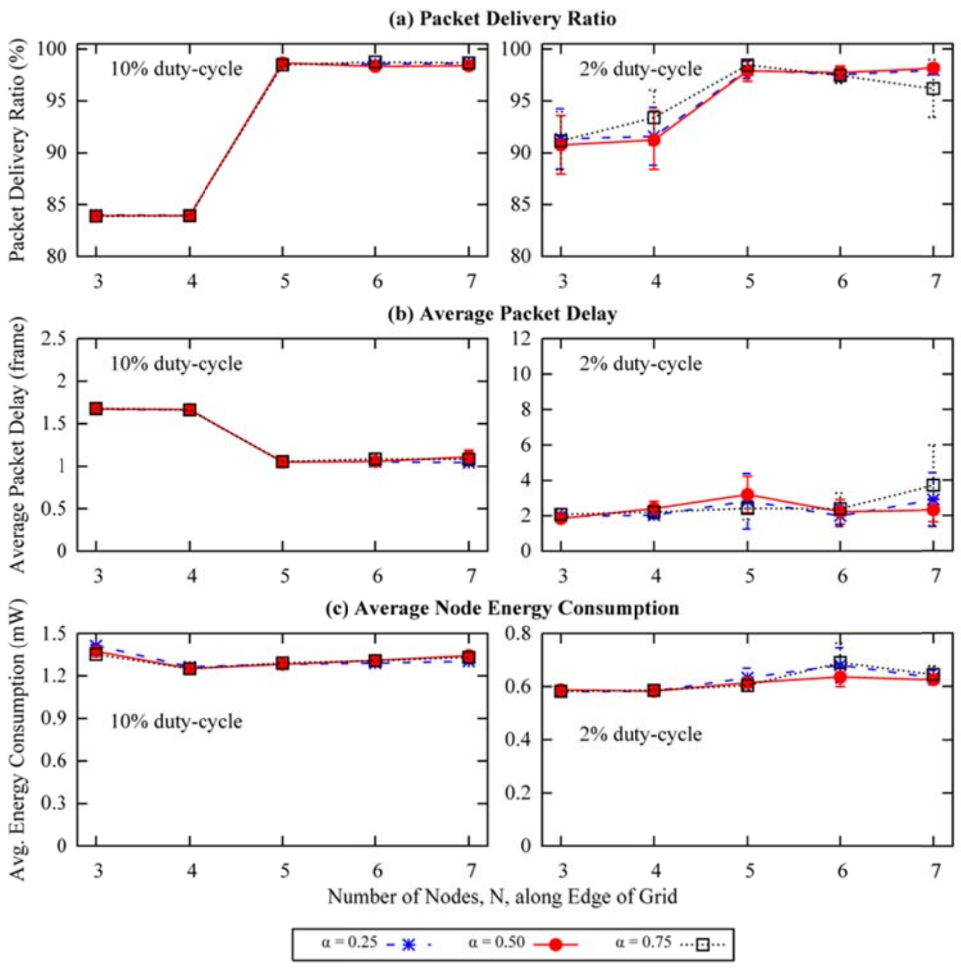


Fig. 5.7 Energy and data performances of C-Sync algorithm with different α in grid networks at 40 ppm drift rate

5.6 Simulation Results and Analysis

The three synchronisation algorithms F-Sync, 1-Sync and C-Sync are simulated under different network densities, clock drift rates, and duty cycles. Means and 95% confidence intervals of the performance metrics for each scenario are plotted for evaluation.

5.6.1 Performance in SYNC windows

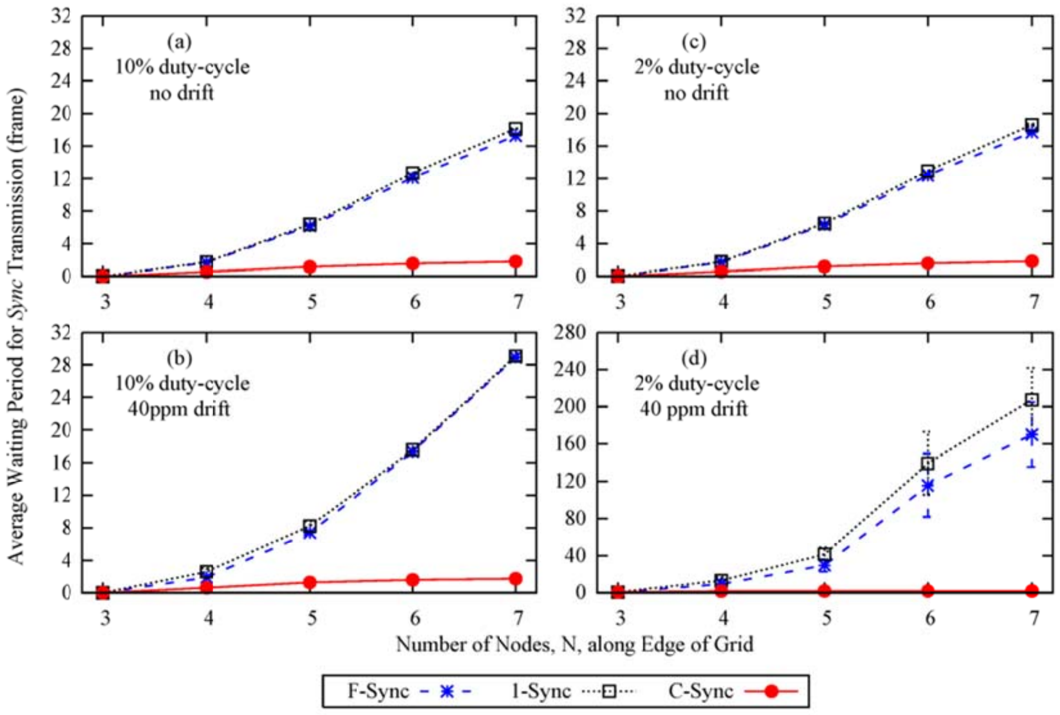


Fig. 5.8 Average waiting period for *sync* packet transmission against different network densities

Fig. 5.8 shows the average waiting period from the time a *sync* packet is scheduled to the time it is transmitted in different density networks. For F-Sync and 1-Sync, *AWPST* increases due to the increased *sync* packet traffic and congestion as density increases. At 10% duty cycle with no drift, *AWPST* performance for F-Sync and 1-Sync are similar (**Fig. 5.8** (a)), increasing from 0.0 frames in the 3x3 network to more than 17.0 frames in the 7x7 network. *AWPST* for C-Sync, on the other hand, is consistently less than 2.0 frames.

Comparing two different drift rates at 0 and 40 ppm, the reference time in the *sync* packet from a transmitting node deviates more from the receiving nodes' local clocks in general at a higher drift rate. If the time difference is greater than a threshold, the receiving nodes may assume that the transmitting node is on a different sleep/wakeup schedule and this can trigger the receiving nodes to generate more *sync* packets. In high density networks, this will further increase the congestion and thus increase the *AWPST*.

In the scenario of the 7x7 network at 2% dc (**Fig. 5.8** (c, d)), *AWPST* for F-Sync and 1-Sync increase tremendously from 17.7 and 18.6 frames at no drift to 170 and 207 frames at 40 ppm drift respectively. This means that F-Sync and 1-Sync nodes have little opportunity to sleep in SYNC windows and consume more energy. On the other hand, *AWPST* for C-Sync are consistently below 2.0 frames.

As can be seen from **Fig. 5.9**, at the very low density of the 3x3 grid with 10% dc, 1-Sync and C-Sync achieve only 61.7% and 63.5% *FDSIT* respectively whereas F-Sync achieves almost 100% *FDSIT* because the nodes are active in all SYNC windows. When the density increases, the performance of 1-Sync and C-Sync improve, achieving more than 99% *FDSIT* in the 7x7 grid network.

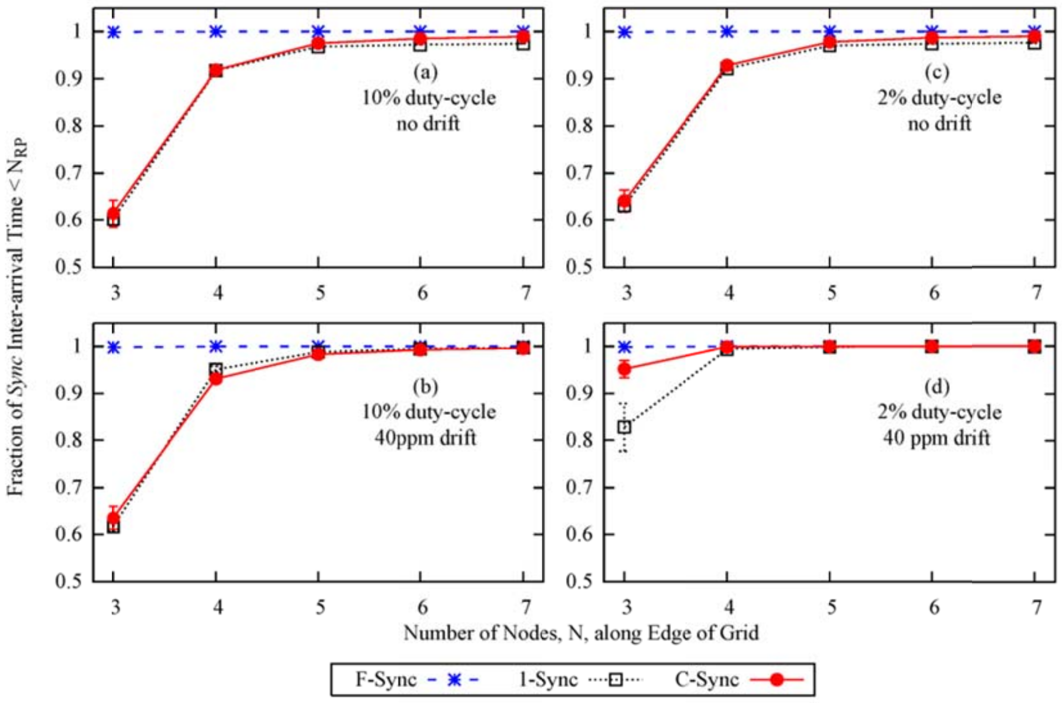


Fig. 5.9 Fraction of *sync* inter-arrival time less than the predetermined period of N_{RP}

It is also worth noting that the presence of clock drift could cause the time difference between sensor nodes to be large. When a sensor node receives a *sync* packet from a sending node that has a time difference larger than a pre-specified value, the receiving node will assume that the sending node is on a different synchronisation schedule (multiple schedules). This will trigger the receiving node to schedule a new *sync* packet to be transmitted in the next SYNC window, which increases the number of *sync* packets transmitted in the network. The increase in the number of *sync* packets improves the FDSIT performance in the low density low duty-cycle networks. In the 3x3 grid network at 2% dc, *FDSIT* improves from 63% to 83% for 1-Sync, and from 64% to 95% for C-Sync.

5.6.2 Performance against Network Densities

The energy and data performance of the three synchronisation algorithms in different density networks are shown in **Fig. 5.10**. The clock drift rate for these scenarios is fixed at 40ppm.

As shown in **Fig. 5.10**(a), all 3 algorithms have similar *PDR* performance at 10% dc, and higher density networks have a better performance than the lower density networks. At 2% dc, C-Sync has the best *PDR* performance among the 3 algorithms ranging from 90.8% to 98.1%.

Fig. 5.10(b) shows the *AvDelay* performance. Similarly, there is no significant difference in *AvDelay* performance at 10% dc. At 2% dc, *AvDelay* performance deteriorates for F-Sync and 1-Sync when density increases, reaching 5.3 and 7.0 frames respectively. C-Sync, on the other hand, has a consistent performance of *AvDelay* less than 3.2 frames.

As shown in **Fig. 5.10**(c), C-Sync is the most energy efficient algorithm except in a very sparse network of 3x3 grid at 10% dc. As network density increases, the relative efficiencies of C-Sync become better. For the 7x7 grid network at 2% dc, average energy consumptions for F-Sync and 1-Sync nodes are 135% and 141% higher than C-Sync nodes.

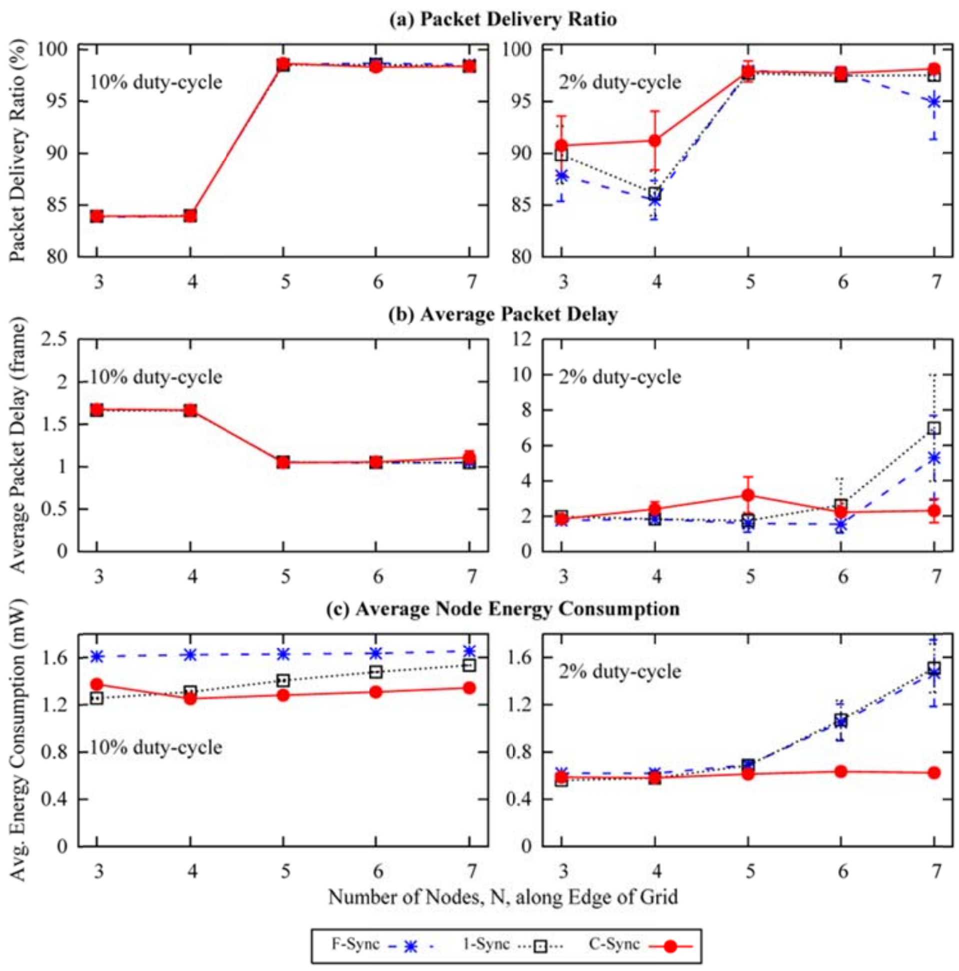


Fig. 5.10 Energy and data performance for F-Sync, 1-Sync and C-Sync in different density grid networks at 40 ppm drift rate

5.6.3 Performance against Duty Cycles

The energy and data performance of the three synchronisation algorithms in different duty-cycle networks are shown in **Fig. 5.11**. The clock drift rate for these scenarios is fixed at 40ppm.

Fig. 5.11(a) shows the *PDR* performance. In both the 3x3 and 7x7 grid networks, C-Sync has similar or better PDR performance than F-Sync and 1-Sync for all duty cycles.

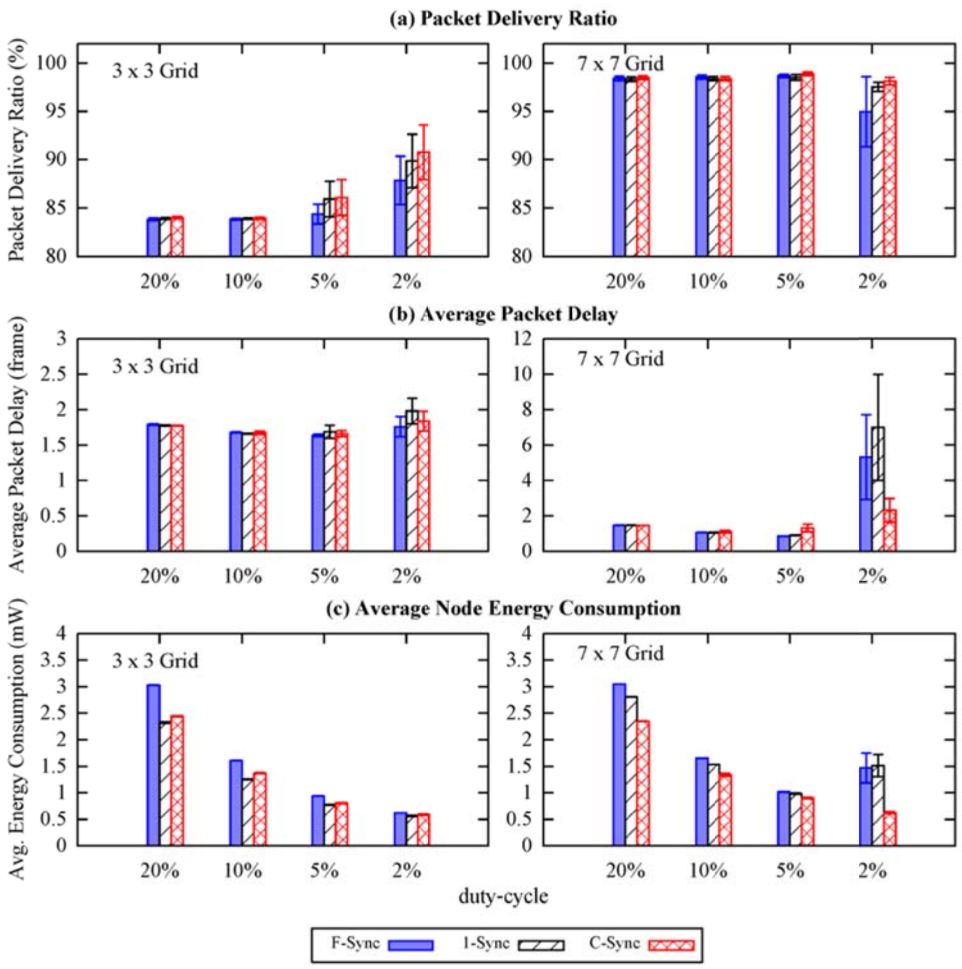


Fig. 5.11 Energy and data performance for F-Sync, 1-Sync and C-Sync in 3x3 and 7x7 grid networks at 40 ppm drift at different duty cycles

As shown in **Fig. 5.11**(b), there is no significant difference among the 3 algorithms in the *AvDelay* performance in the 3x3 grid network. However, the in 7x7 grid network at 2% dc, F-Sync and 1-Sync have 130% and 202% longer delay than C-Sync. At 2% dc, both F-Sync and 1-Sync also have significantly longer delays, as well as and significantly larger delay variations, compared to their performances at higher duty cycles.

As shown in **Fig. 5.11**(c), F-Sync has the highest energy consumption in all except one scenario. Only in the 7x7 grid network with 2% dc does it have marginally lower consumption than 1-Sync. In the low density 3x3 grid network, 1-Sync has the best energy performance, consuming 4% to 9% lower energy than C-Sync. For the 7x7 grid network, C-Sync has the best energy performance, consuming 9% to 142% lower energy than 1-Sync.

Under normal circumstances, energy consumption in a low dc network is lower than in a high dc network due to the existence of longer sleep periods. However, the results show that energy consumptions of F-Sync and 1-Sync in the 7x7 grid network are higher at 2% dc than at 5% dc, which indicate that these two synchronisation algorithms may not be functioning well and are not suitable to operate in low dc, high density networks.

5.6.4 Performance against Clock Drifts

The energy and data performance of the 3 synchronisation algorithms in 3x3 and 7x7 grid networks with different clock drift rates are shown in **Fig. 5.12**. The duty cycle for these scenarios is fixed at 2%.

As shown in **Fig. 5.12**(a), in the dense 7x7 grid network, *PDR* decreases as clock drift increases. This is expected because network synchronisation becomes more challenging when clock drift increases, leading to more errors in data transmission and reception. However, the opposite trend is seen in the sparse 3x3 grid network. When there is no drift, *sync* packets are few and far between in a sparse network. The presence of clock drift could trigger the generation of more *sync* packets in the network as explained in section 5.6.1. The increase in the number of *sync* packets improves the *FDSIT* performance shown in **Fig. 5.9**, which also improves the *PDR* performance in the low density low duty-cycle 3x3 network. However, *AvDelay* is not improved because it is measured based on the delivered *sync* packets only. Similarly, energy consumption increases because more *sync* packets are transmitted with increasing drift.

Comparing the algorithms, C-Sync has the highest *PDR* ranging from 83.5% to 93.1% in the 3x3 grid network. In the 7x7 grid network, both C-Sync and 1-Sync have similar *PDR* ranging from 93.4% to 98.3%, which is marginally higher than F-Sync from 92.8% to 98.1%.

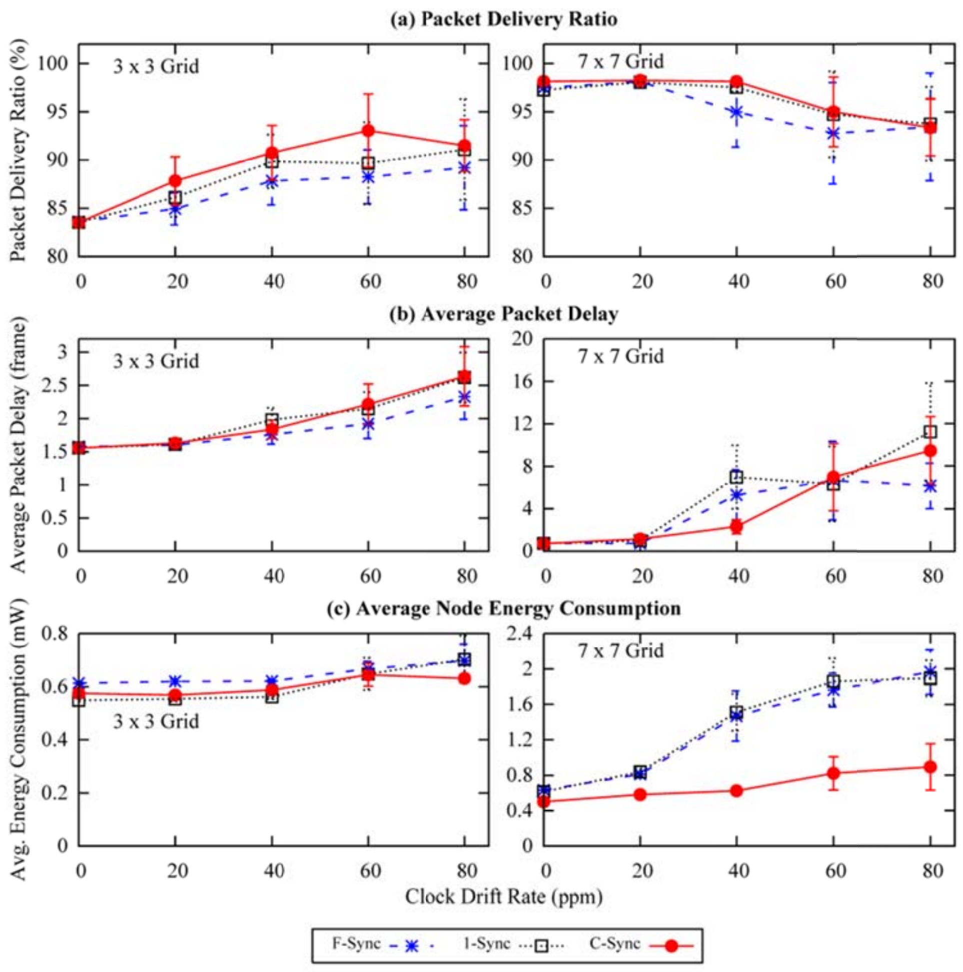


Fig. 5.12 Energy and data performance for F-Sync, 1-Sync and C-Sync in 3x3 and 7x7 grid networks at 2% duty-cycle with different drift rates

Fig. 5.12(b) shows the *AvDelay* performance. All 3 algorithms have a similar delay performance up to a 20 ppm drift rate. The results of *AvDelay* performance are inconclusive as the drift increases further.

In terms of energy consumption, there is no significant difference between 1-Sync and C-Sync in the 3x3 grid network as shown in **Fig. 5.12**(c). The differences are more significant in the 7x7 grid scenarios; C-Sync is more energy efficient than F-Sync and 1-Sync in all drift scenarios. Energy savings range from 18.2% at zero drift to 69% at 80 ppm drift rate.

5.6.5 Individual Node Energy Consumptions

From the results in the previous sections, it is observed that average energy consumptions for F-Sync and 1-Sync increase significantly in high density (7x7 grid), low duty-cycle (2% dc), and high clock drift (80ppm) networks. To account for the unusually large increase in energy consumption, we look into the details of individual node energy consumptions within the grid networks in these scenarios.

5.6.5.1 Effect of Network Density

As shown in **Fig. 5.13**, when network density is low (4x4 grid), node energy consumptions for all three algorithms are almost indistinguishable. The variations in energy consumption among different nodes are also small. As network density increases, both F-Sync and 1-Sync networks display large variations in node energy consumption.

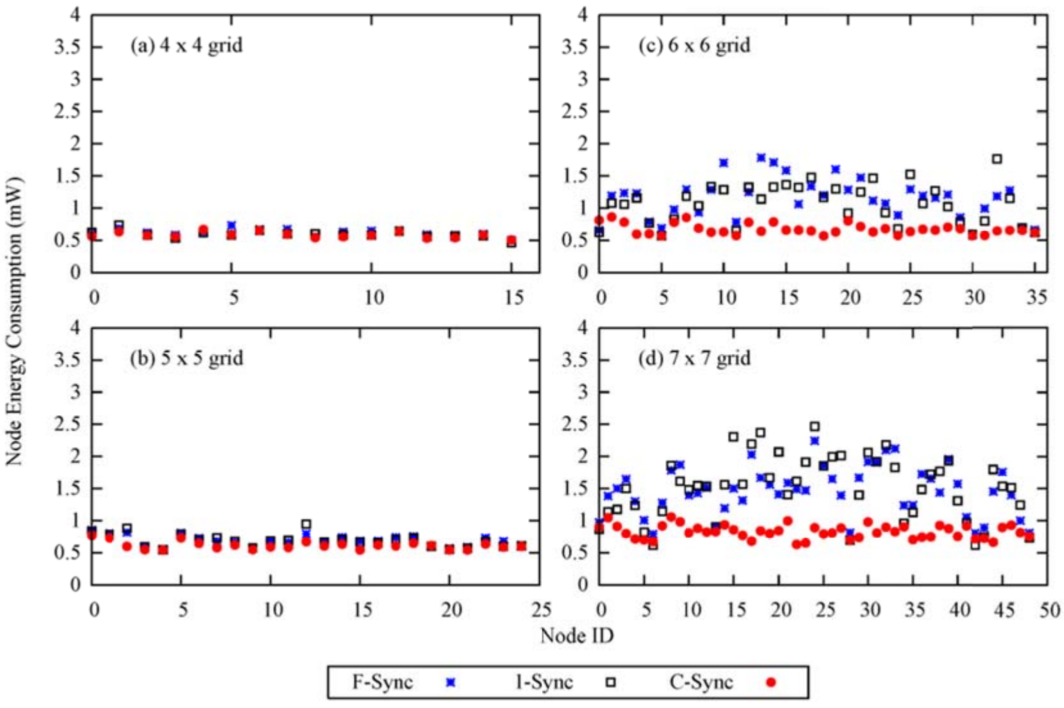


Fig. 5.13 Individual node energy consumptions in different density grid networks at 2% duty-cycle with 40 ppm drift rate

The means and standard deviations of node energy consumptions are shown in **Table 5.2**. The poor energy performance of F-Sync and 1-Sync in high density networks agree with the analysis of the *sync* packet congestion problem of the two algorithms in saturated networks. The large variations in node energy consumption will result in some nodes becoming depleted of energy in much shorter times than the others, which will impact the connectivity and shorten the lifetime of the WSN. On the other hand, the counter-based *sync* packet transmission and *sync* packet cancellation sub-algorithm implemented in C-Sync has effectively removed this problem, as shown by the consistency in node energy consumption across different network densities.

Table 5.2 Means and standard deviations of node energy consumption for different network densities

Synchronisation Algorithm	***grid = 4 x 4***	***5 x 5***	***6 x 6***	***7 x 7***
	Node Energy Consumption - Mean (mW)			
F-Sync	5.602	6.215	10.310	13.167
1-Sync	5.346	6.188	9.641	13.592
C-Sync	5.243	5.527	6.061	5.627
	Node Energy Consumption - Std. Dev. (mW)			
F-Sync	0.507	0.768	2.827	3.367
1-Sync	0.532	0.906	2.757	4.398
C-Sync	0.415	0.554	0.750	0.581

5.6.5.2 Effect of Duty Cycle

Low duty-cycle operations prove to be a great challenge for F-Sync and 1-Sync. As shown in **Fig. 5.14** and **Table 5.3**, F-Sync and 1-Sync perform well in networks with duty cycles at 5% and higher. At 2% dc, both F-Sync and 1-Sync networks display large variations in node energy consumption. At higher duty cycles with shorter *sync* packet intervals, a certain number of *sync* packet collisions and *sync* postponements can be tolerated. However, at lower duty cycles, *sync* packet collisions and *sync* postponements contribute to higher probabilities of asynchronous nodes, which cause instability in the networks and high variations in the node energy consumption.

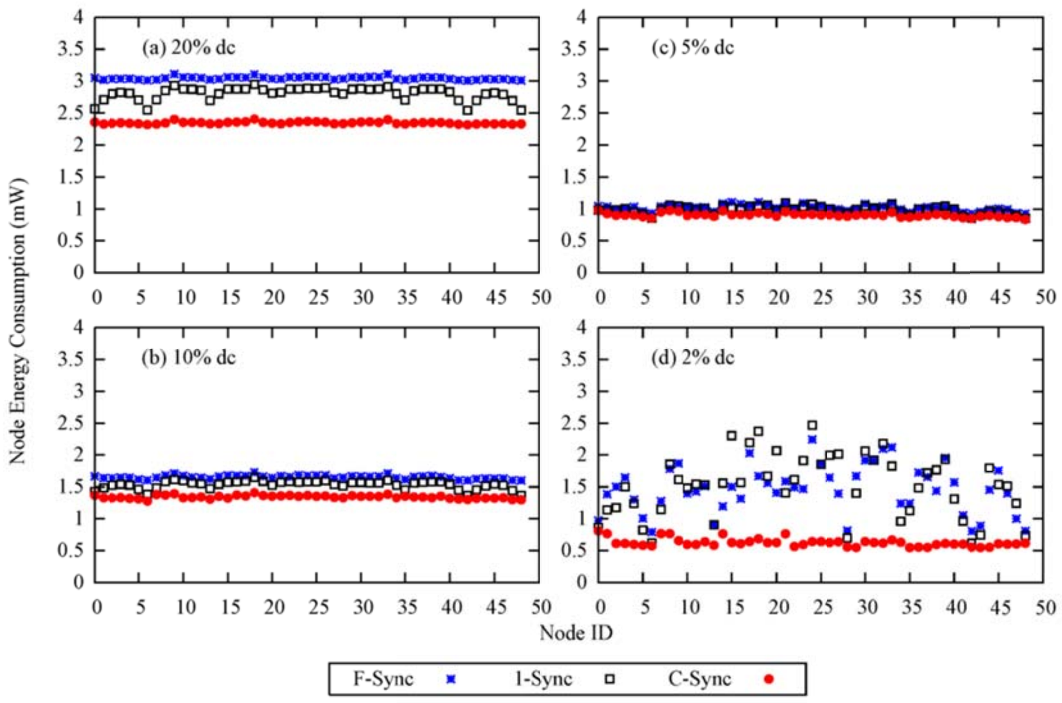


Fig. 5.14 Individual node energy consumptions in 7x7 grid networks with 40 ppm drift rate at different duty-cycles

Table 5.3 Means and standard deviations of node energy consumption for different duty cycles

Synchronisation Algorithm	*dc = 20%*	*10%*	*5%*	*2%*
	Node Energy Consumption - Mean (mW)			
F-Sync	27.405	14.882	9.147	13.167
1-Sync	25.255	13.804	8.857	13.592
C-Sync	21.117	12.082	8.121	5.627
	Node Energy Consumption - Std. Dev. (mW)			
F-Sync	0.214	0.258	0.433	3.367
1-Sync	0.916	0.583	0.541	4.398
C-Sync	0.175	0.235	0.285	0.581

5.6.5.3 Effect of Clock Drift

As shown in **Fig. 5.15**, when there is no drift, all three algorithms function well and energy consumptions are at similar levels among the different nodes in the network. As drift rate increases, not only do the energy consumptions of each node in F-Sync and 1-Sync increase significantly, the variability in node energy consumption also increase significantly. The means and standard deviations of node energy consumptions at different rates are tabulated in **Table 5.4**. C-Sync, on the other hand, maintains a consistent level of energy consumption among the different nodes in the network across all drift rates simulated.

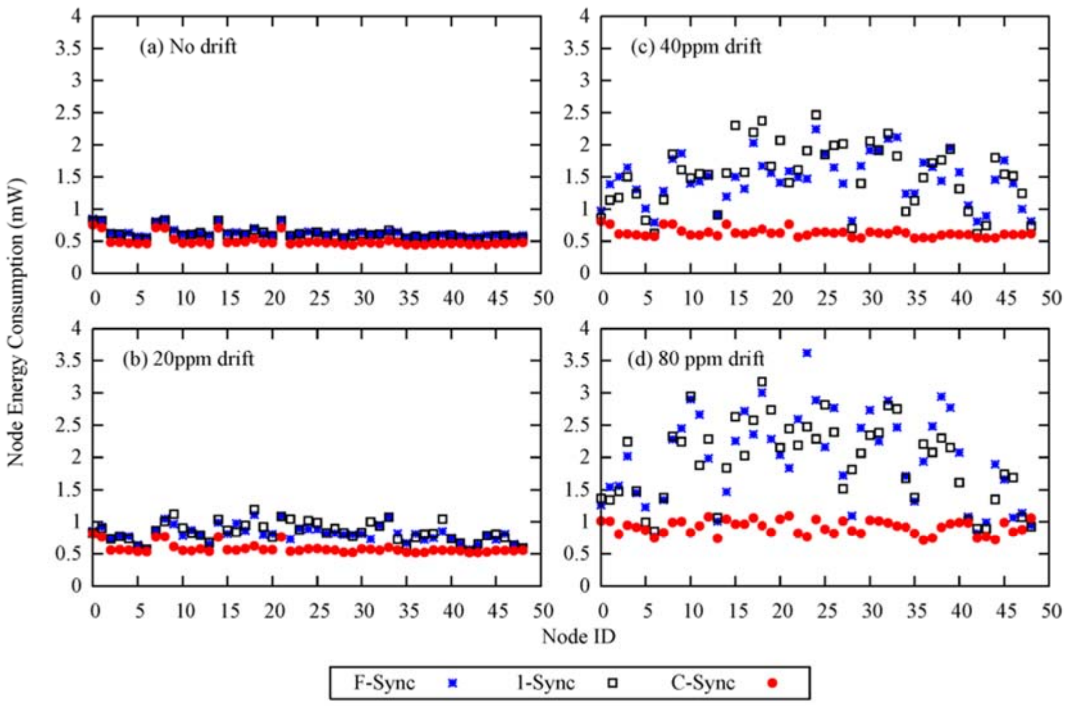


Fig. 5.15 Individual node energy consumptions in 7x7 grid network at 2% duty-cycle with different drift rates

Table 5.4 Means and standard deviations of node energy consumption for different drift rates

Synchronisation Algorithm	***drift = 0 ppm***	***20 ppm***	***40 ppm***	***80 ppm***
	Node Energy Consumption - Mean (mW)			
F-Sync	5.697	7.301	13.167	17.792
1-Sync	5.528	7.544	13.592	17.316
C-Sync	4.512	5.248	5.627	8.213
	Node Energy Consumption - Std. Dev. (mW)			
F-Sync	0.716	1.157	3.367	6.341
1-Sync	0.747	1.378	4.398	5.591
C-Sync	0.754	0.683	0.581	0.970

5.7 Chapter Summary

In this chapter, an adaptive synchronisation algorithm for duty-cycle MAC protocols, C-Sync is proposed. C-Sync reduces energy consumption by adaptively regulating the synchronisation traffic and synchronisation wakeup period based on the changing network neighbourhood conditions through counter-based and exponential smoothing algorithms. The combination of counter-based *sync* packet transmission and exponential-smoothing *sync* packet reception algorithms effectively reduces congestion and collision when *sync* traffic is high and maintains synchronisation performance when *sync* traffic is low.

Extensive simulations of multi-hop multi-neighbourhood grid network scenarios are performed using ns-2. From the results of the simulations, C-Sync consistently outperforms F-Sync and 1-Sync in terms of packet delivery ratio, average packet delay and energy consumption in most of the scenarios. The relative energy performance of C-Sync is also significantly better in the more challenging scenarios of high density, high drift and low duty-cycle networks.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Energy efficiency and network lifetime of WSNs is one of the key challenges faced by the researchers and protocol developers due to the environments in which WSNs are deployed, as well as the physical constraints of sensor nodes imposed by the WSN applications. In general, power efficiency in WSNs can be accomplished through multi-hop routing, low duty-cycle operation, and data aggregation. This thesis has presented the proposed energy-efficient algorithms for multi-hop ad hoc WSNs in the Network and MAC layers that improve the energy and lifetime performances of these networks.

The issue of network lifetime performance in WSNs, caused by the unbalanced routing of data traffic in the Network layer, is discussed in Chapter 3. IoT applications, in which heterogeneous WSN nodes are commonly deployed, further increase the complexity of optimising the network lifetime. A fully distributed and computationally-efficient Energy-balanced Dynamic Source Routing (EB-DSR) protocol is proposed to address the problems of the short lifetime performance of heterogeneous network. The proposed protocol provides a new energy update mechanism to delivery node energy information across the network efficiently. Results from the simulations have shown that EB-DSR is able to prolong the network lifetime effectively through an energy-balanced, multipath approach, while maintaining high packet delivery ratio in both static and mobile heterogeneous WSNs. The results have also shown that the performance of EB-DSR in terms of data load distribution and normalised routing overhead, are much better than the other protocols in the study.

Duty-cycle protocols are commonly used to reduce energy consumption in the MAC layer. However, the synchronisation algorithm adopted by the current synchronous MAC protocols is not energy-efficient. Node energy consumption behaviours of current synchronisation algorithms, such as F-Sync and INS as well as the proposed 1-Sync, are first modelled and analysed for single neighbourhood networks in Chapter 4. Analysis and simulations have shown that the proposed 1-Sync algorithm yields better energy performance than F-Sync in all node densities, similarly besting INS for node densities $N \geq 4$. Although the 1-Sync algorithm has better energy performance than F-Sync, the increase in energy consumption due to *sync* packet collision and postponement in high-density, saturated neighbourhoods is undesirable. To address the issues of *sync* packet collision and postponement, an adaptive synchronisation algorithm C-Sync is proposed in Chapter 5.

C-Sync reduces energy consumption by adaptively regulating the synchronisation traffic and synchronisation wakeup periods based on the changing network neighbourhood conditions through counter-based and exponential-smoothing algorithms. The combining of the counter-based *sync* packet transmission and the exponential-smoothing *sync* packet reception algorithms has effectively reduced congestion and collision when *sync* packet traffic is high and maintains the synchronisation performance when *sync* packet traffic is low. Extensive simulations of multi-hop multi-neighbourhood grid networks with different densities, clock drift rates and duty cycles have been performed. The results of the simulations have shown that C-Sync consistently outperforms F-Sync and 1-Sync in terms of packet delivery ratio, average packet delay and energy consumption in most of the scenarios tested.

6.2 Future Work

There is a wide scope of research that can be done as part of the next step for this work. Areas for further research could include:

i. interaction of routing protocols with the proposed synchronisation algorithm;

ii. performance of synchronisation algorithms with the effect of node mobility;

iii. automatic tuning of counter threshold (C_{thres}) and smoothing factor (α) in the C-Sync algorithm; and

iv. implementation of the proposed synchronisation algorithm with other duty-cycle MAC protocols.

We have simulated the proposed C-Sync algorithm on fixed routing grid networks for performance comparisons across different network densities. Going forward, the performance of the C-Sync algorithm with different dynamic routing protocols could be investigated. This is because dynamic routing provides additional challenges for MAC layer protocols in terms of variations in traffic load and packet delay requirements.

With dynamic routing, the performance of C-Sync on mobile WSNs can also be studied further. The mobility of sensor nodes produces network neighbourhoods with changing densities. It would be interesting to study the adaptive behaviour of the C-Sync algorithm in mobile sensor networks.

The results in Chapter 5 have shown that the performance of the C-Sync algorithm could be further improved if the two parameters C_{thres} and α could be tuned automatically to accommodate different network scenarios. An algorithm that is able to adjust the two parameters dynamically would have the potential to further optimise the energy and data performance of the networks.

In our current study, C-Sync has been paired with S-MAC, which is available in ns-2. S-MAC is one of the earliest synchronous duty-cycle MAC protocols, and forms the basis in the studies of energy-efficient MAC layer protocols. Subsequently, protocols such as DW-MAC and AS-MAC have made several enhancements to S-MAC data scheduling and transmission algorithms to improve energy consumption, data delivery and latency performance. The implementation of C-Sync with DW-MAC or AS-MAC is thus a natural next step to further improve the performance of these two protocols.

References

[1] T. Rault, A. Bouabdallah, and Y. Challal, “Energy efficiency in wireless sensor networks: A top-down survey,” *Computer Networks*, vol. 67, pp. 104–122, 2014.

[2] J. Yick, B. Mukherjee, and D. Ghosal, “Wireless sensor network survey,” *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

[3] K. Sohraby, D. Minoli, and T. Znati, *Wireless sensor networks: technology, protocols, and applications*. John Wiley & Sons, 2007.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer Networks*, vol. 38, pp. 393–422, 2002.

[5] A. K. Singh, S. Rajoriya, S. Nikhil, and T. K. Jain, “Design constraint in single-hop and multi-hop wireless sensor network using different network model architecture,” in *Communication Automation International Conference on Computing*, 2015, pp. 436–441.

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[7] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, “Energy conservation in wireless sensor networks: A survey,” *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, May 2009.

[8] W. Fang, Z. Liu, and F. Liu, “A cross-layer protocol for reliable and efficient communication in wireless sensor networks,” *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 10, pp. 7185–7198, 2012.

[9] T. Melodia, M. C. Vuran, and D. Pompili, “The state of the art in cross-layer design for wireless sensor networks,” in *Wireless systems and network architectures in next generation internet*, Springer, 2006, pp. 78–92.

[10] “The Network Simulator - ns-2.” [Online]. Available: http://www.isi.edu/nsnam/ns/.

[11] M. I. A. Khan and M. Atif, “An Overview of Mobile Ad-hoc Network Simulators and Associated Simulation Techniques,” *International Journal of Computer Science and Telecommunications*, vol. 3, no. 6, Jun. 2012.

[12] K.-P. Ng and C. Tsimenidis, “Energy-balanced dynamic source routing protocol for wireless sensor network,” in *IEEE Conference on Wireless Sensors*

(ICWISE), 2013, pp. 36–41.

[13] K.-P. Ng, C. C. Tsimenidis, and W. L. Woo, “Energy-efficient synchronization algorithm for duty-cycle MAC protocols,” in *IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, 2015, pp. 255–260.

[14] K.-P. Ng, C. Tsimenidis, and W. L. Woo, “C-Sync: Counter-based Synchronization for Duty-cycled Wireless Sensor Networks,” *Ad Hoc Networks*, vol. 61, pp. 51–64, 2017.

[15] “Gartner says 6.4 billion connected ‘Things’ will be in use in 2016, up 30 percent from 2015,” *Gartner Press Release, November 10, 2015*. [Online]. Available: http://www.gartner.com/newsroom/id/3165317.

[16] C. Bormann, A. P. Castellani, and Z. Shelby, “Coap: An application protocol for billions of tiny internet nodes,” *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[17] A. Banks and R. Gupta, “MQTT Version 3.1. 1,” *OASIS standard*, vol. 29, 2014.

[18] D. H. Mun, M. L. Dinh, and Y. W. Kwon, “An Assessment of Internet of Things Protocols for Resource-Constrained Applications,” in *IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, 2016, vol. 1, pp. 555–560.

[19] S. Bandyopadhyay and A. Bhattacharyya, “Lightweight Internet protocols for web enablement of sensors using constrained gateway devices,” in *International Conference on Computing, Networking and Communications (ICNC)*, 2013, pp. 334–340.

[20] K. E. Nolan, W. Guibene, and M. Y. Kelly, “An evaluation of low power wide area network technologies for the Internet of Things,” in *International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2016, pp. 439–444.

[21] U. Raza, P. Kulkarni, and M. Sooriyabandara, “Low Power Wide Area Networks: An Overview,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 855–873, 2017.

[22] H. Wang and A. O. Fapojuwo, “A Survey of Enabling Technologies of Low Power and Long Range Machine-to-Machine Communications,” *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2017.

[23] M. Johnson *et al.*, “A comparative review of wireless sensor network mote technologies,” in *IEEE Sensors*, 2009, pp. 1439–1442.

[24] S. Sudevalayam and P. Kulkarni, “Energy Harvesting Sensor Nodes: Survey and Implications,” *IEEE Communications Surveys Tutorials*, vol. 13, no. 3, pp. 443–461, 2011.

[25] K. Z. Panatik *et al.*, “Energy harvesting in wireless sensor networks: A survey,” in *IEEE 3rd International Symposium on Telecommunication Technologies (ISTT)*, 2016, pp. 53–58.

[26] Z. Zeng, X. Li, A. Bermak, C. Y. Tsui, and W. H. Ki, “A WLAN 2.4-GHz RF energy harvesting system with reconfigurable rectifier for wireless sensor network,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 2362–2365.

[27] X. Lu and S.-H. Yang, “Thermal energy harvesting for WSNs,” in *IEEE International Conference on Systems, Man and Cybernetics*, 2010, pp. 3045–3052.

[28] Deepti and S. Sharma, “Energy harvesting using piezoelectric for Wireless Sensor Networks,” in *IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, 2016, pp. 1–3.

[29] Y. Wu, W. Liu, and Y. Zhu, “Design of a wind energy harvesting wireless sensor node,” in *IEEE Third International Conference on Information Science and Technology (ICIST)*, 2013, pp. 1494–1497.

[30] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, “Power management in energy harvesting sensor networks,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 4, p. 32, 2007.

[31] G. J. Pottie and W. J. Kaiser, “Wireless integrated network sensors,” *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[32] R. C. Carrano, D. Passos, L. Magalhaes, and C. V. Albuquerque, “Survey and taxonomy of duty cycling mechanisms in wireless sensor networks,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 181–194, 2014.

[33] D. Goyal and M. R. Tripathy, “Routing protocols in wireless sensor networks: a survey,” in *Advanced Computing & Communication Technologies (ACCT), Second International Conference*, 2012, pp. 474–480.

[34] I. Stojmenović and S. Olariu, “Data-centric protocols for wireless sensor networks,” in *Handbook of sensor networks: algorithms and architectures*, Wiley, 2005, pp. 417–456.

[35] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, “A review of routing protocols for mobile ad hoc networks,” *Ad hoc networks*, vol. 2, no. 1, pp. 1–22, 2004.

[36] A. Koliousis and J. Sventek, “Proactive vs reactive routing for wireless sensor networks,” *Department of Computing Science, University of Glasgow, Glasgow*, 2007.

[37] C. E. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers,” in *ACM SIGCOMM computer communication review*, 1994, vol. 24, pp. 234–244.

[38] T. Clausen and P. Jacquet, “RFC 3626: Optimized link state routing protocol (OLSR),” *IETF, October*, vol. 4, 2003.

[39] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, “Optimized link state routing protocol for ad hoc networks,” in *IEEE INMIC Multi Topic Conference Proceedings*, 2001, pp. 62–68.

[40] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, “Scalable routing strategies for ad hoc wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1369–1379, 1999.

[41] M. Gerls, “Fisheye State Routing (FSR) for ad hoc networks,” *Internet Draft, draft-ietf-manet-fsr-03. txt*, 2002.

[42] D. Johnson, Y. Hu, and D. Maltz, “RFC 4728: The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4.” Feb-2007.

[43] C. Perkins, E. Belding-Royer, and S. Das, “Ad hoc on-demand distance vector (AODV) routing (RFC 3561),” IETF MANET Working Group, 2003.

[44] Z. J. Haas, M. R. Pearlman, and P. Samar, “The zone routing protocol (ZRP) for ad hoc networks,” *Internet Draft, draft-ietf-manet-zone-zrp-04.txt*, 2002.

[45] Z. J. Haas and M. R. Pearlman, “The performance of query control schemes for the zone routing protocol,” *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 4, pp. 427–438, 2001.

[46] M. Joa-Ng and I.-T. Lu, “A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks,” *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1415–1425, 1999.

[47] N. Nikaein, H. Labiod, and C. Bonnet, “DDR: distributed dynamic routing algorithm for mobile ad hoc networks,” in *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, 2000, pp. 19–27.

[48] N. Nikaein, C. Bonnet, and N. Nikaein, “Harp-hybrid ad hoc routing protocol,” in *Proceedings of international symposium on telecommunications (IST)*, 2001,

pp. 56–67.

[49] S. Doshi, S. Bhandare, and T. X. Brown, “An on-demand minimum energy routing protocol for a wireless ad hoc network,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 3, pp. 50–66, Jun. 2002.

[50] S. Banerjee and A. Misra, “Minimum energy paths for reliable communication in multi-hop wireless networks,” in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, New York, USA, 2002, pp. 146–156.

[51] J. Zhu, C. Qiao, and X. Wang, “On Accurate Energy Consumption Models for Wireless Ad Hoc Networks,” *Trans. Wireless. Comm.*, vol. 5, no. 11, pp. 3077–3086, Nov. 2006.

[52] J. Ren, Y. Tu, M. Zhang, and Y. Jiang, “An ant-based energy-aware routing protocol for ad hoc networks,” in *International Conference on Computer Science and Service System (CSSS)*, 2011, pp. 3844–3849.

[53] J.-H. Chang and L. Tassiulas, “Routing for Maximum System Lifetime in Wireless Ad-hoc Networks,” in *37th Annual Allerton Conference on Communication, Control, And Computing*, 1999.

[54] B. K. Cetin, N. R. Prasad, and R. Prasad, “A novel linear programming formulation of maximum lifetime routing problem in wireless sensor networks,” in *International Wireless Communications and Mobile Computing Conference*, 2011, pp. 1865–1870.

[55] G. Martinez, S. Li, and C. Zhou, “Multi-commodity online maximum lifetime utility routing for energy-harvesting wireless sensor networks,” in *IEEE Global Communications Conference*, 2014, pp. 106–111.

[56] Q. Li, J. Aslam, and D. Rus, “Online Power-aware Routing in Wireless Ad-hoc Networks,” in *MOBICOM*, 2001, pp. 97–107.

[57] L. Sheng, J. Shao, and J. Ding, “A Novel Energy-Efficient Approach to DSR Based Routing Protocol for Ad Hoc Network,” in *International Conference on Electrical and Control Engineering*, 2010, pp. 2618–2620.

[58] J. E. Garcia, A. Kallel, K. Kyamakya, K. Jobmann, J. C. Cano, and P. Manzoni, “A novel DSR-based energy-efficient routing algorithm for mobile ad-hoc networks,” in *IEEE 58th Vehicular Technology Conference*, 2003, vol. 5, pp. 2849–2854 Vol.5.

[59] Y. Luo, J. Wang, and S. Chen, “An energy-efficient DSR routing protocol based on mobility prediction,” in *International Symposium on a World of Wireless,*

Mobile and Multimedia Networks (WoWMoM), 2006, p. 3 pp.–446.

[60] C. K. Toh, “Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks,” *IEEE Communications Magazine*, Jun. 2001.

[61] J.-H. Chang and L. Tassiulas, “Maximum lifetime routing in wireless sensor networks,” *IEEE/ACM Trans. Netw.*, vol. 12, no. 4, pp. 609–619, Aug. 2004.

[62] A. Sankar and Z. Liu, “Maximum Lifetime Routing in Wireless Ad-hoc Networks,” in *INFOCOM*, 2004, pp. 1089–1097.

[63] H. Hassanein and A. Zhou, “Routing with load balancing in wireless Ad hoc networks,” in *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, New York, USA, 2001, pp. 89–96.

[64] G. S. Tomar, L. Shrivastava, and S. S. Bhadauria, “Load Balanced Congestion Adaptive Routing for Randomly Distributed Mobile Adhoc Networks,” *Wireless Pers Commun*, vol. 77, no. 4, pp. 2723–2733, Aug. 2014.

[65] L. Wang, L. Zhang, Y. Shu, and M. Dong, “Multipath source routing in wireless ad hoc networks,” in *Canadian Conference on Electrical and Computer Engineering*, 2000, vol. 1, pp. 479–483.

[66] D. Kim, R. Ha, and H. Cha, “Traffic load and lifetime deviation based power-aware routing protocol for wireless ad hoc networks,” in *Proceedings of the 4th international conference on Wired/Wireless Internet Communications*, Berlin, Heidelberg, 2006, pp. 325–336.

[67] Y. Z. Zhao, C. Miao, M. Ma, J. B. Zhang, and C. Leung, “A survey and projection on medium access control protocols for wireless sensor networks,” *ACM Computing Surveys*, vol. 45, no. 1, p. 7, 2012.

[68] N. K. Ray and A. K. Turuk, “A review on energy efficient MAC protocols for wireless LANs,” in *IEEE International Conference on Industrial and Information Systems (ICIIS),* 2009, pp. 137–142.

[69] M. R. Ahmad, E. Dutkiewicz, and X. Huang, “A Survey of low duty cycle MAC protocols in wireless sensor networks,” in *Emerging Communications for Wireless Sensor Networks*, InTech, Croatia, 2010, pp. 69–90.

[70] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi, “The Evolution of MAC Protocols in Wireless Sensor Networks: A Survey,” *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 101–120, 2013.

[71] M. Doudou, D. Djenouri, N. Badache, and A. Bouabdallah, “Synchronous contention-based MAC protocols for delay-sensitive wireless sensor networks: A review and taxonomy,” *Journal of Network and Computer Applications*, vol. 38, pp. 172–184, 2014.

[72] M. Doudou, D. Djenouri, and N. Badache, “Survey on latency issues of asynchronous MAC protocols in delay-sensitive wireless sensor networks,” *IEEE Trans. on Communications Surveys Tutorials*, vol. 15, no. 2, pp. 528–550, 2013.

[73] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks,” in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 95–107.

[74] M. Buettner, G. V. Yee, E. Anderson, and R. Han, “X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks,” in *Proceedings of the 4th international conference on Embedded networked sensor systems*, 2006, pp. 307–320.

[75] A. El-Hoiydi and J.-D. Decotignie, “WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks,” in *Algorithmic Aspects of Wireless Sensor Networks*, Springer, 2004, pp. 18–31.

[76] W. Pak, K.-T. Cho, J. Lee, and S. Bahk, “W-MAC: Supporting Ultra Low Duty Cycle in Wireless Sensor Networks,” in *IEEE Global Telecommunications Conference*, 2008, pp. 1–5.

[77] D. L. Mills, “Internet time synchronization: the network time protocol,” *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.

[78] D. L. Mills, “A Brief History of NTP Time: Memoirs of an Internet Timekeeper,” *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 9–21, Apr. 2003.

[79] J. Elson and D. Estrin, “Time synchronization for wireless sensor networks,” in *Proceedings of the 15th international parallel and distributed processing symposium*, 2001, vol. 1, pp. 1965–1970.

[80] S. Youn, “A Comparison of Clock Synchronization in Wireless Sensor Networks,” *International Journal of Distributed Sensor Networks*, 2013.

[81] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, “Clock Synchronization of Wireless Sensor Networks,” *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 124–138, Jan. 2011.

[82] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, “Clock synchronization for wireless sensor networks: a survey,” *Ad Hoc Networks*, vol. 3, pp. 281–323,

2005.

[83] S. El Khediri, N. Nasr, A. Kachouri, and A. Wei, “Synchronization in wireless sensors networks using balanced clusters,” in *6th Joint IFIP Wireless and Mobile Networking Conference (WMNC)*, 2013, pp. 1–4.

[84] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, “The Flooding Time Synchronization Protocol,” in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2004, pp. 39–49.

[85] J. Shannon, H. Melvin, and A. G. Ruzzelli, “Dynamic Flooding Time Synchronisation Protocol for WSNs,” in *IEEE Global Communications Conference (GLOBECOM)*, 2012, pp. 365–371.

[86] S. Ganeriwal, R. Kumar, and M. B. Srivastava, “Timing-sync Protocol for Sensor Networks,” in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2003, pp. 138–149.

[87] M. L. Sichitiu and C. Veerarittiphan, “Simple, accurate time synchronization for wireless sensor networks,” in *IEEE Conference on Wireless Communications and Networking*, 2003, vol. 2, pp. 1266–1273.

[88] D. R. Jeske, “On maximum-likelihood estimation of clock offset,” *IEEE Transactions on Communications*, vol. 53, no. 1, pp. 53–54, 2005.

[89] Q. M. Chaudhari, E. Serpedin, and K. Qaraqe, “On maximum likelihood estimation of clock offset and skew in networks with exponential delays,” *IEEE Transactions on Signal Processing,* vol. 56, no. 4, pp. 1685–1697, 2008.

[90] J. Van Greunen and J. Rabaey, “Lightweight time synchronization for sensor networks,” in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, 2003, pp. 11–19.

[91] I.-K. Rhee, J. Lee, J. Kim, E. Serpedin, and Y.-C. Wu, “Clock synchronization in wireless sensor networks: An overview,” *Sensors*, vol. 9, no. 1, pp. 56–85, 2009.

[92] J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” *ACM SIGOPS Operating Systems Review*, vol. 36(SI), pp. 147–163, 2002.

[93] W. Su and I. F. Akyildiz, “Time-diffusion synchronization protocol for wireless sensor networks,” *IEEE/ACM Transactions on Networking,* vol. 13, no. 2, pp. 384–397, 2005.

[94] F. Gong and M. L. Sichitiu, “CESP: A power efficient, accurate coefficient exchange synchronization protocol,” in *IEEE International Conference on*

Wireless for Space and Extreme Environments (WiSEE), 2013, pp. 1–6.

[95] W. Ye, J. Heidemann, and D. Estrin, “An energy-efficient MAC protocol for wireless sensor networks,” in *Proceedings of Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002, vol. 3, pp. 1567–1576.

[96] M. Kuorilehto, M. Kohvakka, J. Suhonen, P. Hämäläinen, M. Hännikäinen, and T. D. Hämäläinen, “MAC Protocols,” in *Ultra-Low Energy Wireless Sensor Networks in Practice*, John Wiley & Sons, Ltd, 2007, pp. 73–89.

[97] W. Ye, J. Heidemann, and D. Estrin, “Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks,” *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, Jun. 2004.

[98] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson, “DW-MAC: a low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks,” in *Proceedings of the 9th ACM international*, 2008.

[99] Y. Z. Zhao, M. Ma, C. Y. Miao, and T. N. Nguyen, “An energy-efficient and low-latency MAC protocol with Adaptive Scheduling for multi-hop wireless sensor networks,” *Computer Communications*, vol. 33, no. 12, pp. 1452–1461, 2010.

[100] Y. Z. Zhao, C. Y. Miao, and M. Ma, “An Energy-Efficient Self-Adaptive Duty Cycle MAC Protocol for Traffic-Dynamic Wireless Sensor Networks,” *Wireless Personal Communications*, vol. 68, no. 4, pp. 1287–1315, Feb. 2013.

[101] K. Nguyen, Y. Ji, and S. Yamada, “Low overhead MAC protocol for low data rate wireless sensor networks,” *International Journal of Distributed Sensor Networks*, vol. 2013, 2013.

[102] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *IEEE Trans. onWireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[103] Y. Li, W. Ye, and J. Heidemann, “Energy and latency control in low duty cycle MAC protocols,” in *IEEE Wireless Communications and Networking Conference*, 2005, vol. 2, pp. 676–682.

[104] A. Bachir, M. Dohler, T. Watteyne, and K. K. Leung, “MAC essentials for wireless sensor networks,” *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 222–248, 2010.

[105] N. Salman, I. Rasool, and A. H. Kemp, “Overview of the IEEE 802.15.4 standards family for low rate wireless personal area networks,” in *IEEE 7th International Symposium on Wireless Communication Systems (ISWCS)*, 2010,

pp. 701–705.

[106] Y. Sun, O. Gurewitz, and D. B. Johnson, “RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks,” in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008, pp. 1–14.

[107] J. Kulik, W. Heinzelman, and H. Balakrishnan, “Negotiation-based protocols for disseminating information in wireless sensor networks,” *Wireless networks*, vol. 8, no. 2/3, pp. 169–185, 2002.

[108] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, “Directed Diffusion for Wireless Sensor Networking,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, 2003.

[109] D. Braginsky and D. Estrin, “Rumor routing algorithm for sensor networks,” in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, 2002, pp. 22–31.

[110] I. Dietrich and F. Dressler, “On the lifetime of wireless sensor networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 1, p. 5, 2009.

[111] D. B. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks,” in *Mobile computing*, Springer, 1996, pp. 153–181.

[112] S. Chiaravalloti, F. Idzikowski, and Ł. Budzisz, “Power consumption of WLAN network elements,” *TKN Technical Report Series*, no. TKN-11–002, 2011.

[113] S. Climent, A. Sanchez, J. V. Capella, N. Meratnia, and J. J. Serrano, “Underwater acoustic wireless sensor networks: advances and future trends in physical, MAC and routing layers,” *Sensors*, vol. 14, no. 1, pp. 795–833, 2014.

[114] L. M. Feeney and M. Nilsson, “Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment,” in *In IEEE Infocom*, 2001, pp. 1548–1557.

[115] N. H. Mak and W. K.-G. Seah, “How Long is the Lifetime of a Wireless Sensor Network?,” in *International Conference on Advanced Information Networking and Applications*, 2009, pp. 763–770.

[116] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, “A performance comparison of multi-hop wireless ad hoc network routing protocols,” in *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, 1998, pp. 85–97.

[117] P. Sthapit and J.-Y. Pyun, “Intelligent network synchronization for energy saving in low duty cycle MAC protocols,” in *IEEE International Symposium on a*

World of Wireless, Mobile and Multimedia Networks Workshops, 2009, pp. 1–6.

[118] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, “The Broadcast Storm Problem in a Mobile Ad Hoc Network,” *Wireless Networks*, vol. 8, no. 2–3, pp. 153–167, Mar. 2002.

[119] D. Koscielnik and J. Stepien, “The methods of broadcasting of information in ad-hoc wireless networks with mobile stations,” in *IEEE International Symposium on Industrial Electronics (ISIE)*, 2011, pp. 2043–2048.

[120] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih, “Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network,” *IEEE Transactions on Computers*, vol. 52, no. 5, pp. 545–557, May 2003.