Transfer Thesis

Noradila Nordin

Department of Electronic & Electrical Engineering

University College London

Torrington Place

London

WC1E 7JE

noradila.nordin.12@ucl.ac.uk

January 4, 2016

Abstract

This report presents the current state of research work that has been carried out in the context of the PhD. (describe why do it). The PhD work proposes a new decentralised multi-channel tree building protocol with a centralised controller for ad-hoc sensor networks. The protocol alleviates the effect of interference which results in improved network efficiency and stability, and link reliability. The proposed protocol takes into account all available channels to utilise the spectrum and aims to use the spectrum efficiently by transmitting on several channels. The protocol detects which channels suffer interference and changes away from those channels. The algorithm for channel selection is a two-hop colouring protocol that reduces the chances of nearby nodes to transmit on the same channel. All nodes are battery operated except for the low power border router (LPBR). This enables a centralised channel switching process at the LPBR. The protocol is built based on the routing protocol for low power and lossy networks (RPL). In its initial phase, the protocol uses RPL's standard topology formation to create an initial working topology and then seeks to improve this topology by switching channels. The report discusses the main engineering and research challenges raised by the protocol, and describes and explains the principles and mechanisms used to support the proposed protocol. It then presents an extensive evaluation of the protocol and other other approaches. The implementation and evaluation of the protocol is performed using the Contiki framework. The report then describes the future main research issues that will be investigated in the context of this PhD.

In this report,

The proposed approach The report discusses

Acknowledgements

Acknowledge all the things!

Contents

1 Introduction **8**

1.1 Context and Motivation 8

1.2 Problem Statement 9

1.3 Contribution 9

1.4 Current Work 10

1.5 Report Outline 10

2 Literature Review **12**

2.1 Wireless Sensor Networks 12

2.1.1 Overview (Application Scenarios for WSNs 12

2.2 Maximize Lifetime and Minimizing Energy 13

2.2.1 MAC Protocols 14

2.2.2 Routing Protocols 15

2.2.3 Transmit Power Control 16

2.2.4 Energy Harvesting 16

2.3 Multichannel Protocol (Data Link Layer) 16

2.3.1 Introduction (Solutions) 17

2.3.2 Synchronous Systems 18

2.3.3 Asynchronous Systems 18

2.3.4 Comparison and Discussion 21

2.4 Routing Protocols (Network Layer Protocols) 21

2.4.1 Classification of Routing Protocols 22

3 Multichannel Routing Protocol (Prot and Algo) 29

3.1 MCRP Design 30

3.2 Channel Selection Strategy 31

3.3 Channel Switching 33

3.4 Channel Quality Checking 34

3.5 Reconnection Strategy 35

4 Implementation 36

4.1 Protocol Stack (changes at MAC, RT, NBR TB, BR etc.) 36

4.2 MCRP Implementation 39

4.2.1 Low Power Border Router 39

4.2.2 Other Nodes (MAC Layer?) 43

4.2.3 Routing Layer - Neighbour Discovery 45

4.3 Memory Footprint/Setup Overhead? 45

5 Results and Discussions 46

5.1 Experimental Setup 46

5.2 Evaluation 48

5.2.1 Packet loss rates with single channel RPL versus multi-channel 49

5.2.2 Setup Overhead 50

6 Energy vs Loss Tradeoff 52

7 Future Work 53

7.1 Conclusions 53

7.2 Future Works 53

Bibliography 54

List of Figures

3.1 Channel switching processes 33

4.1 Low power border router 39
4.2 LPBR processes 43

5.1 Level of packet loss for mild, moderate and extreme interference levels using single channel 49
5.2 Level of packet loss for scenario 1 and scenario 2 using multi channel 49

List of Tables

4.1 Contiki Network Stack 37

Chapter 1

Introduction

1.1 Context and Motivation

Wireless Sensor Networks (WSN) are ad-hoc networks that consist of sensor nodes that typically use low power radios such as IEEE 802.15.4, a relatively short range transmission standard radio technology in the 2.4 GHz band. The standard allows transmission to occur on several different channels within this band [1]. Unfortunately, the channels used by this technology often suffer interference [2, 3], for example, from Wi-Fi [4, 5] and Bluetooth. Sensor networks have to contend with an increasing number of devices that cause this wireless interference. Organising the network topology around this interference becomes an enabler for increasing transmission efficiency at a smaller energy cost. WSNs need to be able to operate reliably in the presence of such interference. It is important to minimise energy costs in these networks since deployments can be for weeks, months or longer.

Multichannel communication in wireless networks can alleviate the effects of interference which, as a result, can improve the network efficiency and stability, link reliability and minimise latency [6]. It also enables communication between physically proximate nodes to occur simultaneously without the risk of collision when the communicating nodes use different channels. However, not all channels are free from interference; thus, there is a gain to hop to another channel when the quality of the channel deteriorates. Two commonly used types of channel hopping [6] are blind channel hopping and whitelisting. In blind channel hopping, nodes

choose from all available channels. Whitelisting, on the other hand, gives a set list of channels that avoids those that are known to commonly suffer interference. Many studies make use of channel whitelisting such as in Chrysso [7] and MiCMAC [8].

Note that potentially Chrysso and MiCMAC could use all available channels. However, they do not have a mechanism to check the channel condition before using it for packet transmission. MiCMAC sees its performance degraded when using more than 4 channels, thus the decision on specifying 4 channels to be included in their experiment. MiCMAC uses a different channel chosen at random each time it wakes up. It might require several wake up periods which is time consuming, before a clear channel is found from the 16 channels, to deliver the packet. Chrysso on the other hand, switches the affected nodes to a new set of channels upon detecting interference which entails frequent channel switching if all channels are to be considered.

1.2 Problem Statement

It is clear that it is impossible to find a single channel guaranteed free from interference and there is no consensus on the best channel to use. Our work takes into account all available channels to utilise the spectrum and checks the condition of the channels before hopping to avoid those channels with interference. Several previous studies have developed a multichannel MAC layer but, despite the potential benefits none are yet widely implemented in real world deployments.

1.3 Contribution

Important aspects of this work will be investigating lossy multichannel. Designing the protocol for multichannel raises several research challenges. The decision making process is (centralized and decentralized explain here). The main benefits of this approach are (). The work in this PhD will address the issues raised by (). More specifically it will investigate how the channel selection is determined from the nodes interactions. In addition, it will investigate the cross layer interaction.

1.4 Current Work

In the context of the research efforts carried out from the beginning of the PhD research work, the followings have been investigated. A new multi-channel protocol called Multichannel Cross-Layer Routing Protocol (MCRP) has been developed. The proposed approach is so that the nodes are able to communicate on many channels in order to avoid interference and channel congestion in a centralized and decentralized manner. This paper presents a Multichannel Cross-Layer Routing Protocol (MCRP) which consists of two main parts; a centralised intelligence at LPBR, and decentralised nodes. LPBR implements a two-hop colouring algorithm to avoid interference between physically proximate nodes trying to communicate on the same channel. The information on channel interference and network topology from the lower layer is made available to the application layer. This allows the centralised controller (LPBR) to have an overall view of the system to make decisions at the network and MAC layers about which channels nodes should listen on. The systemis fail safe in the sense that the WSN functions if the central system which assigns channels fails temporarily or permanently. We implement MCRP in Contiki [9], an open source operating system for WSNs and evaluate the protocol in Contiki network simulator, Cooja [10]. We demonstrate that MCRP avoids channels with interference which greatly reduces the effects of interference on the network. The performance of the approach has been evaluated using (). The evaluation has been performed with respect to the ().s

1.5 Report Outline

The remainder of the report is organised as follows. Chapter 2 introduces the state-of-the art in the area of multichannel protocols. It also presents the main current research efforts towards () Section **??** presents related work to multichannel protocols. Chapter 3 presents the main features and mechanisms used in MCRP. It describes (). It also presents (). Section **??** describes the key idea of our proposed protocol and the high-level design, and the implementation of the protocol in Contiki. We describe and evaluate the experimental results in Section **??**. Chapter 7

summarises the current work and presents the future research works that will be investigated in the context of this PhD.

Chapter 2

Literature Review

2.1 Wireless Sensor Networks

A WSN is a network of sensor nodes with the purpose of collecting sensor measurements from the target environment, and sending these measurements over the radio. One classic example is environmental monitoring, where sensor nodes are distributed over the area of interest measuring some properties there; such as temperature. Sensor nodes can be used for continuous sensing, event detection, location sensing and local control of actuators.

There are 5 types of WSNs [11]. Multimedia WSN [12].

The evolution of wireless sensor networks is largely driven by a set of emerging applications. The smart grid is intended to improve the electrical power grid and save considerable amounts of energy for society as a whole. Building and home automation improves the quality of indoor environments in terms of temperature, air quality, and lighting, while saving energy in the process. Industrial automation improves the quality of industrial processes. Smart cities allow new services inside increasingly populated cities such as automatic parking management and pollution monitoring. Wireless sensors are an integral part of all these applications [13]

2.1.1 Overview (Application Scenarios for WSNs

The application can be categorised into 5; military, environment, health, home and other commercial areas [14]

2.2 Maximize Lifetime and Minimizing Energy

-through routing protocol (clustering), adjust transmission range, MAC sleep awake -energy harvesting?

Network lifetime depends on many factors including network architecture and protocols, data collection initiation, lifetime definition, channel characteristics and energy consumption model. Two physical layer parameters are crucial to network lifetime: the channel state and the residual energy of sensors. It indicates that lifetime maximizing protocols should exploit both the channel state information and the residual energy information of individual sensors. Important network characteristics that affect network lifetime includes network architecture (routing topology), channel and energy consumption model - sensor sleeping, energy consumed in transmission, reception and possible channel acquisition; lifetime definition - network lifetime is the time span from the deployment to the instant when the network is considered nonfunctional. When a network should be considered nonfunctional is however, application-specific. It can be for example, the instant when the first sensor dies, a percentage of sensors die, the network partitions, or the loss of coverage occurs. Define the network lifetime as the time span until any sensor in the network dies (the first death) or no sensor has enough energy for transmission during a data collection (the first failure in data collection), whichever occurs first [51].

A precise definition of a WSN lifetime is application dependent: some application might tolerate a loss of considerable number of nodes, while for other applications a loss of even a node is critical. Network lifetime is defined as the length of time until the first battery expends its available energy. Although network lifetime is strongly related to minimal remaining energy of all the nodes, the best network lifetime could not be achieved by only maximized the minimal remaining energy of all the nodes. Usually place too heavy burden of forwarding data on a couple of key nodes so that these nodes drain out their batteries quickly which shortens lifetime of the WSN [15].

Wireless systems are often powered by batteries whereby the system's energy consumption determines its lifetime.

Prolonging network lifetime and saving energy are two critical issues for WSNs. Routing protocols are needed in which data packets are transmitted via multi-hop manner to reach the sink or destination. Therefore, it is critical for a WSN to run effectively to design an energy-efficient routing algorithm in which battery energy is expended efficiently while the WSN has a longer lifetime. The best network lifetime could not be achieved by only considering the minimal remaining energy of all the nodes in the WSN. Combining the minimal remaining energy and total energy consumption plays a key role in prolonging network lifetime of a WSN [15].

1. MAC - multichannel 2. Routing 3. Transmit power control

2.2.1 MAC Protocols

Main causes of energy consumption. Collisions - may happen when a node is within the transmission range of two or more nodes that are simultaneously transmitting so that it does not capture any frame. The energy drained in the transmission and reception of collided frames is just wasted. Overhearing - happens when a node drains energy receiving irrelevant packets or signals. Idle listening - when a node does not know when it will be the receiver of a frame; the node keeps its radio on while listening to the channel waiting for potential data frames. The amount of energy wasted whilst the radio is on is considerable even when it is neither receiving nor transmitting frames. Nodes waste considerable amounts of energy as they keep their radios on for large time intervals while listening to an idle channel. Energy-efficient MACs should make nodes sleep for long periods of time instead of enabling them to be permanently active. Put nodes to sleep as long as possible while avoiding deafness and reducing overhearing and overhead. In order to extend nodes lifetime, applications need to save more energy by lowering the duty cycle. Lowering the duty cycle implies putting nodes in sleep mode for larger periods, which means extending the check interval. [16]

A major problem is deploying WSNs in their dependence on limited battery power. A main design criterion is to extend the lifetime of the network without jeopardizing reliable and efficient communications from sensor nodes to the other

nodes as well as data sinks. One solution is the medium access control (MAC). The prime role of the MAC is to coordinate access to and transmission over a medium common to several nodes. The common medium is the wireless channel. Since the radio is controlled by the MAC, the MAC is central in optimizing the WSN's lifetime. The aim of a WSN design is to guarantee its longevity under the given energy and complexity constraints. The MAC plays a central part in this design since it controls the active and sleeping state of each node. The MAC protocols hence needs to trade longevity, reliability, fairness, scalability and latency; throughput is rarely a primary design factor. [16]

Many energy-efficient MAC protocols have been proposed to improve the lifetime of sensor networks by reducing the energy consumed by idle listening and overhearing. The idle listening problem refers to a node listening to the channel even though there are no radio transmissions to receive. The overhearing problem refers to a node receiving a packet it is not intended to receive [17].

2.2.2 Routing Protocols

A routing algorithm termed Energy-efficient Routing Algorithm to Prolong Lifetime (ERAPL) is proposed. A data gathering sequence (DGS) used to avoid and eliminate mutual transmission and loop transmission among nodes (node is only allowed to transmit to its neighboring node in forward direction only to avoid loop), is constructed and each node proportionally transmit traffic to the links confined in the DGS. The main task of the ERAPL is to determine the optimal outgoing traffic to maximize the network lifetime for a given WSN. In addition, a mathematical programming model, in which minimal remaining energy of nodes and total energy consumptions are included, is presented to optimized network lifetime. ERAPL is a centralised algorithm and runs at the sink; sink knows the topology of the WSN. Sink inform all the nodes in the WSN of a packet that contains the constructed DGS which guides all the nodes to transmit traffic to their respective neighbors so that mutual transmission among nodes and route loop is avoided and accordingly energy is saved. ERAPL can improve network lifetime while expending energy efficiently by constructing a DGS and finding the optimal outgoing traffic proportions for all

the nodes to distribute packets to their respective neighbouring nodes [15].

2.2.3 Transmit Power Control

Power control to improve energy efficiency of the wireless sensor network. Whilst it is a useful tool to control traffic flows, congestion and interference levels, the power savings due reduced transmission powers are negligible since the radio's power consumption is in the range of the transmit power levels typically employed for embedded WSN nodes. [16]

2.2.4 Energy Harvesting

Energy harvesting involves nodes replenishing its energy from an energy source. Potential energy sources include solar cells, vibration, fuel cells, acoustic noise and a mobile supplier. In terms of harvesting energy from the environment, solar cell is the current mature technique that harvest energy from light. There is also work in using mobile energy supplier such as a robot to replenish energy. The robots would be responsible in charging themselves with energy and then delivering energy to nodes [11].

Sparse sensor placement may result in long-range transmission and higher energy usage while dense sensor placement may result in short-range transmission and less energy consumption.

Note that different powering mechanisms are available, such as non-rechargeable battery; rechargeable battery with regular recharging (e.g. sunlight); rechargeable battery with irregular recharging (e.g. opportunistic energy scavenging); capacitive/inductive energy provision (e.g. RFPD); etc [16].

2.3 Multichannel Protocol (Data Link Layer)

In single-channel MAC protocols, all nodes are configured to use a single frequency all the time. Frequency-agile MAC protocols switch between multiple frequencies during run-time. Recent radio chips are able to switch between frequency channels fast (e.g. in less than 100μs). Multi-channel can be used to increase robustness against narrowband long-lasting and transient interference. [16]

2.3.1 Introduction (Solutions)

Multichannel communication has potential benefits for wireless networks that possibly include improved resilience against external interference, reduced latency, enhanced reception rate and increased throughput. There have been some proposals/solutions for multichannel. These approaches focus on (the mac layer) and depending on ().

The duty cycling technique saves energy by switching nodes between awake and sleeping states. The duty cycling is an important mechanism for reducing energy consumption in sensor networks. Existing duty cycling energy-efficient MAC protocols can be categorized into two types; synchronous and asynchronous.

Radio duty cycling mechanisms can be classified into two categories; synchronous and asynchronous systems. A synchronous system is a system that requires a tight time synchronization between nodes. It uses time-scheduled communication where the network clock needs to be periodically synchronized in order for the nodes not to drift in time. Asynchronous system on the other hand, do not require synchronization but instead is a sender or receiver initiated communication. In asynchronous systems the nodes are able to self-configure without time synchronization and this can have advantages. There are many studies done in multichannel for both categories.

The sender-initiated approach, a sender transmits preamble before a packet transmission to notify the receiver of the upcoming packet. With receiver-initiated approach, in contrast, sender preambles are replaced with receiver wakeup beacons. Receiver-initiated wakeup beacons are used to avoid long sender-initiated preambles. However, a larger sender duty cycle due to idle listening until the receiver wakes up. In sender-initiated protocols, a sender often shows much larger duty cycle than a receiver, transmitting the preamble until the receiver wakes up [17]. To overcome this, many sender-initiated protocols use predictive wakeup in sensor network MAC protocols to enable reducing the preamble length (WiseMAC - fixing the node wakeup interval. PW-MAC - wake up according to independently generated pseudo-random scheduless).

Two main approaches; reservation-based protocols (synchronous) where it requires the knowledge of the network topology to establish a schedule that allows each node to access the channel and communicate with other nodes. It needs dependency on network topology and time synchronization. Tight synchronization to ensure a common schedule among nodes. Both knowledge of topology and strict synchronization requires large overheads. Contention-based protocols - neither global synchronization nor topology knowledge is required. Nodes compete for the use of the wireless medium and only the winner of this competition is allowed to access to the channel and transmit. In CSMA for instantce, a node having a packet to transmit first senses the channel before actually transmitting. In the case that the node finds the channel busy, it postpones its transmission to avoid interfering with the ongoing transmission. Contention-based protocols suffer from degraded performance in terms of throughput when the traffic load increases. [16]

2.3.2 Synchronous Systems

-TSCH, MC-LMAC, YMAC

2.3.2.1 TSCH

2.3.2.2 MC-LMAC

2.3.2.3 YMAC

2.3.3 Asynchronous Systems

Recent asynchronous multi channel MAC layers are Chrysso and MiCMAC. MiCMAC is built based on ContikiMAC, the default radio duty cycling in Contiki 2.7 that works in a single channel. The details of these are explained below.

2.3.3.1 ContikiMAC

ContikiMAC radio duty cycling mechanism is the default radio duty cycling mechanism in Contiki 2.7. It uses a power efficient wake up mechanism with a set of timing constraints to allow device to keep their transceivers off. The wireless transceiver consumes as much power when passively listening for transmissions from other devices as it does when actively transmitting, so the transceiver must

be completely turned off to save power. ContikiMAC keep their radios turned off for roughly 99% of the time. ContikiMAC uses only asynchronous mechanisms, no signalling messages, and no additional packet headers. ContikiMAC packets are ordinary link layer messages. ContikiMAC uses a fast sleep optimization, to allow receivers to quickly detect false positive wake-ups (fast sleep optimization to allow receivers to quickly go to sleep when faced with spurious radio interference), and a transmission phase-lock optimization. The idea of periodic wake-ups has been used by many protocols, such as B-MAC, X-MAC and BoX-MAC. The phase-lock optimization has been previously suggested by WiseMAC and has since been used by other protocols as well.

ContikiMAC uses a fast sleep optimization, to allow receivers to quickly detect false-positive wake-up and a transmission phase-lock optimization, to allow run-time optimization of the energy-efficiency of transmissions.

ContikiMAC is a radio duty cycling protocol that uses periodical wake-ups to listen for packet transmissions from neighbors. If a packet transmission is detected during a wake-up, the receiver is kept on to be able to receive the packet. UNICAST - When the packet is successfully received, the receiver sends a link layer acknowledgement. To transmit a packet, a sender repeatedly sends its packet until it receives a link layer acknowledgement from the receiver. Acknowledgement transmission is done as part of the unicast packet reception. Packets that are sent as broadcasts do not result in link layer acknowledgements. Instead, the sender repeatedly sends the packet during the full wake-up interval to ensure that all neighbors have received it. Since a broadcast transmission does not expect any link layer acknowledgement, the transmitter can turn of its radio between each packet transmission to save power.

ContikiMAC wake-up frequency of 8Hz which results in a wake-up interval of 125 ms. Radio duty cycle increase with the wake-up frequency; more wake-ups, the total power consumption of the network increase (channel check rate higher than 8Hz).

ContikiMAC wake-ups use an inexpensive Clear Channel Assessment (CCA) mechanism that uses the Received Signal Strength Indicator (RSSI) of the radio

transceiver to give an indication of radio activity on the channel. If the RSSI is below a given threshold, the CCA returns positive, indicating that the channel is clear. If the RSSI is above the threshold, the CCA returns negative, indicating that the channel is in use.

Detection - ContikiMAC CCAs do not reliably detect packet transmission: they only detect that the radio signal strength is above a certain threshold. The detection of a radio signal may mean that a neighbor is transmitting a packet to the receiver, that a neighbor is transmitting to another receiver, or that some other device is radiating radio energy that is being detected by the CCA mechanism. ContikiMAC must be able to discern between these events and react properly.

Fast Sleep - The fast sleep optimization lets potential receivers go to sleep earlier if the CCA woke up due to spurious radio noise. Specific pattern of ContikiMAC transmissions: If CCA detects radio activity but the radio activity has a duration that is longer than the maximum packet length, the CCA has detected noise and can go back to sleep (if the activity period is not followed by a silence period). If the radio activity is followed by a silence period that is longer than the interval between two successive transmissions, the receiver can go back to sleep. If the activity period is followed by a silence period of the correct length, followed by activity but no start of packet could be detected, the receiver can go back to sleep.

Transmission Phase-Lock - A sender can learn of a receiver's wake-up phase by making note of the time at which it saw a link layer acknowledgement from the receiver. The sender can assume that the reception of a link layer acknowledgement means that the sender has successfully transmitted a packet within the receiver's wake-up window and thus the sender has found the receiver's wake-up phase. The sender can commence its successive transmissions to this receiver just before the receiver is expected to be awake. The transmission will be significantly shorter than a normal transmission, because it occurs just before the neighbor is expected to be awake. Reducing the length of the transmission thus reduces radio congestion. The phase-lock mechanism is implemented as a separate module from ContikiMAC. The phase-lock mechanism maintains a list of neighbors and their wake-up phases.

Fast sleep and phase-lock optimizations significantly reduce power consumption. This is because of a phase-locked transmission being shorter than non-phased-locked transmissions, leading both to less energy being spent on transmissions and to less radio congestion [18].

2.3.3.2 MiCMAC

2.3.3.3 Chrysso

2.3.4 Comparison and Discussion

2.4 Routing Protocols (Network Layer Protocols)

The network layer is responsible in routing the data across the network from the source to the destination. Routing protocols in WSNs differs from traditional routing protocols depending on the Operating System. Contiki provides IP communication in both IPv4 and IPv6. However, as sensors have a small amount of memory, uIP, which is a small RFC-compliant TCP/IP stack that makes it possible to communicate over the Internet [19, 20]. uIP () to reduce the resources it requires. uIP implementation is designed to have only the absolute minimal set of features needed for a full TCP/IP stack [19, 20]. In order to maximize the use of multichannel in improving packet delivery, routing topology plays a big role in providing an optimized routing tree to the network that is scalable and energy efficient. Routing protocol approaches can be classified into () types which are flat based and data centric, hierarchical, location based and network flow and quality of service (QoA) aware.

2.4.1 Classification of Routing Protocols

2.4.1.1 Flat based and Data Centric

2.4.1.2 Location Based

2.4.1.3 Network Flow and QoS-aware

2.4.1.4 Hierarchical

CTP

2 principles for wireless routing protocols; datapath validation - data traffic quickly discovers and fixes routing inconsistencies; adaptive beaconing - extending the Trickle algorithm to routing control traffic reduces route repair latency and sends fewer beacons. CTP Neo - an implementation of CTP. Datapath validation actively uses data packets to validate the routing topology and detect loops. Each data packet contains the link-layer transmitters estimate of its distance. A node detects a possible routing loop when it receives a packet to forward from a node with a smaller or equal distance to the destination. CTP is a routing protocol that computes anycast routes to a single or a small number of designated sinks in a wireless sensor network. CTP may appear very simple. They provide best-effort, unreliable, anycast packet delivery to one of the data sinks in the network. 4 goals; reliability - a protocol should deliver at least 90Rapid topology changes necessitate distance-vector rather than link-state algorithms. Simple distance-vector protocols however suffer from routing loops and other problems that harm reliability and efficiency. Link topology changes may result in transient loops which causes packet drops. A collection protocol builds and maintains minimum cost trees to nodes that advertise themselves as tree roots. Collection is address-free; when there are multiple base stations, it sends to the one with the minimum cost without knowing its address. Every node maintains an estimate of the cost of its route to a collection point. ETX as the cost metric (any similar gradient metric can work just well). ETX does not effectively capture throughput. A nodes cost is the cost of its next hop plus the cost of its link to the next hop. The cost of a route is the sum of the costs of its links. Collection points advertise a cost of zero. Each data packet contains the transmit-

ters local cost estimates. When a node receives a packet to forward, it compares the transmitters cost to its own. Cost must always decrease. When a timer interval expires, Trickle doubles it, up to a maximum value. When Trickle hears a newer version number, it shrinks the timer interval to a small value. Trickle enables quick discovery of new nodes and recovery from failures, while at the same time enabling long beacon intervals when the network is stable [21].

CTP provides best effort anycast datagram communication to one of the collection roots in a network. A collection protocol delivers data to one of possibly several data sinks, providing many-to-one network layer. CTP uses routing frames to update and build collection tree in the network. CTP uses data frames to deliver application payload to the sink and to probe topology inconsistencies. CTP is a tree-based collection protocol. Some nodes advertise as tree roots. Nodes form a set of routing trees to these roots. CTP is address free in that a node does not send a packet to a particular root, instead, it implicitly chooses a root by choosing a next hop. Nodes generate routes to roots using a routing gradient. CTP assumes that it has link quality estimates of some number of nearby neighbors (ETX). These provides an estimate of the number of transmissions it takes for the node to send a unicast packet whose acknowledgement is successfully received. CTP uses expected transmission (ETX) as its routing gradient. A root has an ETX of 0. ETX of a node is the ETX of its parent plus the ETX of its link to its parent. CTP should choose the one with the lowest ETX value. Problem is routing loops; occur when a node choose a new route that has a significantly higher ETX than its old one. CTP tries to resolve the inconsistency by broadcasting a beacon frame. Packet duplication - when a node receives a data frame successfully and transmit an ACK but ACK is not received. Thus CTP keeps a small cache of packet signature for the packets it has seen to detect packet duplicates.CTP data frames has additional time has lived (THL) field which the routing layer increments on each hop. Link-layer retransmission has the same THL. If nodes ETX value changes significantly, CTP should transmit a broadcast frame to notify other nodes which might change their routes. A parent can detect when a childs ETX is significantly below its own. When

a parent hears a child advertise an ETX below its own, it must schedule a routing frame for transmission in the near future [22].

** Contiki Collect protocol and CTP are state-of-the-art address-free data collection protocols that provide a way for nodes to send data packets towards a data sink. Nodes do not need to know the address of the sink. Use ETX finding paths that minimize the number of packet transmissions to reach the root. Neither CTP nor Contiki Collect are IPv6-based. Contiki Collect uses the Contiki Rime stack [13].

The data collection is an address-free protocol that sends messages towards a sink node somewhere in the network. The protocol is address-free in the sense that the originating nodes do not send their messages to a specific addressed node. Instead, the nodes send their messages towards the nearest sink in the network. The protocol does two things. It first builds a tree that originates at the sink nodes. The nodes build the tree by sending periodic announcements containing the number of hops away from the sink. After having built the tree, the nodes start sending messages towards the root of the tree. The protocol sends the messages using hop-by-hop reliable unicast [**?**].

RPL

The protocol makes use of IPv6 and supports not only traffic in the upwards direction, but also traffic flowing from a gateway node to all other network participants. RPL is a distance vector routing protocol that makes use of IPv6. A Destination Oriented Directed Acyclic Graph (DODAG) which is routed at a single destination is built. The graph is constructed by the use of an Objective Function (OF) which defines how the routing metric is computed. OF specifies how routing constraints and other functions are taken into account during topology construction. The protocol tries to avoid routing loops by computing a nodes position relative to other nodes with respect to the DODAG root, called Rank and increases if nodes move away from the root and decreases when nodes move in the other direction. RPL specification defines 4 types of control messages for topology maintenance and information exchange. DODAG Information Object (DIO) is the main source

of routing control information. It may store information like the current Rank of a node, the current RPL Instance, the IPv6 address of the root, etc. Destination Advertisement Object (DAO) enables the support of down traffic and is used to propagate destination information upwards along the DODAG. DODAG Information Solicitation (DIS) makes it possible for a node to require DIO messages from a reachable neighbor. DAO-ACK (optional) is sent by a DAO recipient in response to a DAO message. RPL specification defines all 4 types of control messages as ICMPv6 information messages with a requested type of 155. RPL adapts the sending rate of DIO message by extending the Trickle algorithm. Upward routing is a standard procedure which enables network devices to send data to a common data sink, also called sometimes a gateway or root node. (BORDER ROUTER??!!) The Mode of Operation (MOP) field is set by the DODAG root A DIO message may be extended by the use of options. DODAG Configuration option plays a crucial role for parameter exchange. MaxRankIncrease field defines an upper limit for the Rank. MinHopIncrease field stores the minimum increase of the Rank between a node and any of its parent nodes. 3 types of nodes in a RPL network. 1. Root nodes which are commonly referred in literature as gateway nodes that provide connectivity to another network. 2. Routers which may advertise topology information to their neighbors. 3. Leafs that do not send any DIO messages and only have the ability to join an existing DODAG. The construction of the topology starts at a root node that begins to send DIO messages. Each node that receives the message run an algorithm to choose an appropriate parent. The choice is based on the used metric and constraints defined by the OF. Afterwards, each of them computes its own Rank and in case a node is a router, it updates the Rank in the DIO message and sends it to all neighboring peers. *In most sensor node deployments several data collection points (root nodes) are needed. Whenever the sending timer expired, RPL doubles it up to the maximum value. Whenever RPL detects an event which indicates that the topology needs active maintenance, it resets the timer to minimum value. Router nodes forward DIO control messages for topology maintenance - such messages are sent in a multicast manner to the neighboring nodes. RPL node

does not process DIO messages from nodes deeper (higher Rank) than itself. RPL metric - status includes typical resources such as CPU usage, available memory and left energy. Node energy consumption - node should consider the energy level of its neighbors before picking them as possible parents. RPL metric specification defines 3 possible states for the first information field: powered, on batteries and scavenger **** This may be a rough estimation of how much load a node experiences for a given period of time. ETX - is an approximation of the expected number of transmissions until a data packet reaches the gateway node*. A node that is one hop away from the root with perfect signal strength and very little interference may have ETX of 1. ETX is bidirectional single-hop link quality computation between 2 neighbor nodes.* A metric called Packet Reception Rate (PRR) is calculated at the receiver node for each window of received packets. Downward routing - by supporting P2MP traffic it is possible for a network administrator to control nodes that are even not in range. RPL specification defines 2 modes of operation for supporting P2MP. 1. Non-storing mode which makes use of source routing. In this mode each node has to propagate its parent list up to the root. After receiving such topology information, the root computes the path to the destinations. Each node has to extend the DAO message. After collecting the needed information, the root pieces the downward route together. If it needs to send a data packet to a given destination the IPv6 Source Routing header is used. 2. Storing mode which is fully stateful. Each non-root and non-leaf network participant has to maintain a routing table for possible destinations. DAO messages are used by RPL nodes to propagate routing information in order to enable P2MP traffic. DAO is no longer propagated to the DODAG root. Instead, it is sent as unicast to all parent nodes which maintain additional downward routing tables [23].

Routing protocol called RPL. RPL does not define any specific routing metrics, path costs or forwarding policies. RPL leaves this open so that different networks can apply different mechanisms to meet different objectives such as minimizing latency or minimizing energy consumption. ContikiRPL implementation of the RPL protocol which allows replaceable routing objective functions. ContikiRPL

is the main IPv6 routing protocol in Contiki. RPL is a distance-vector protocol for IPv6 networks comprising low-power devices connected by lossy links. The protocol maintains Directed Acyclic Graph (DAG) topologies toward root nodes. The topologies are built proactively according to an objective function. It is flexible regarding the rules to form topologies and to select next-hops for individual packets. Routing decisions are taken by the objective function, which essentially specifies the constraints and metrics used in a network. One objective function uses a simple hop count and one uses expected transmissions (ETX) to do the forwarding decision. The simple hop-count objective function results in a shorter path length at the expense of higher power consumption [24].

DIS - may be used to solicit a DIO from a RPL node. A node may use DIS to probe its neighborhood for nearby DODAGs. DIO - carries information that allows a node to discover a RPL Instance, learn its configuration parameters, select a DODAG parent set and maintain the DODAG. DAO - used to propagate destination information Upward along the DODAG. In Storing mode, the DAO message is unicast by the child to the selected parent(s). In Non-Storing mode, the DAO message is unicast to the DODAG root. The DAO message may optionally be acknowledged by its destination with a Destination Advertisement Acknowledgement (DAO-ACK) message back to the sender of the DAO. DAO-ACK message is sent as a unicast message packet by a DAO recipient (a parent or DODAG root) in response to a unicast DAO message [25].

RPL is a routing protocol that provides any-to-any routing in low-power Ipv6 networks, standardized by the IETF in March 2012. Its design is largely based on CTP, the reference data collection protocol for sensor networks. The RPL topology is a DODAG (Destination Oriented Directed Acyclic Graph) built in direction of the root, typically an access point to the Internet. Any-to-any traffic is routed first upwards, i.e. towards the root until a common ancestor of destination and source is found, and then downwards, following the nodes routing table. RPL uses a simple rooted topology instead of a full mesh; it is devoted to the maintenance of reliable paths to a single destination. The purpose of this strategy is to scale to large net-

works while containing the routing overhead, at the price of increased hop count (routing via a common ancestor). RPL terminology, the distance from a node to the root according to the routing metric is called rank. RPL requires sharing routing tables among siblings. In RPL, nodes propagate their routing entries through unicast (so-called DAO messages) to their parents. RPLs rank hysteresis mechanism prevents nodes from switching parent for too little rank improvements. Contiki-MAC has wakeup consists of two clear channel assessments and has a phase-lock mechanism where senders record their neighbors wake-up phase and use it to make the next transmissions cheaper. *Experience an outage during which they cannot receive or send any data. This reflects for example scenarios where battery maintenance requires to disconnect a part of the network, or where external interference (e.g. WiFi or Bluetooth) affects communication. RPL experiences a sharp drop in the reliability during the first outage, consequence of failed MAC transmissions. Nodes react by switching parent, which heals the topology and slowly improves reliability. The next outages result in less churn (less agitate) [26].

////OBJECTIVE FUNCTION

////TRICKLE TIMER A protocol uses Trickle to periodically advertise the most recent data it has received, typically through a version number. Routing control traffic - a protocol uses Trickle to control when it sends beacons that contain routing state. Once the RPL network is established, it reduces the rate of control messages, exponential increase. To avoid control message explosion, nodes suppress transmissions if it hears too many messages from other - called the Trickle algorithm.

Dynamically adjusting transmission windows allows Trickle to spread new information on the scale of link-layer transmission times while sending only a few messages per hour when information does not change.

To save energy the DIOs are sent periodically controlled by the trickle timer whose duration is doubled each time it is fired. The value of trickle timer starts from the lowest possible value lmin and is doubled each time it is transmitted until it reaches its maximum possible value of lmax [27].

Chapter 3

Multichannel Routing Protocol (Prot and Algo)

In this chapter, we focus specifically on the reliability of radio communication links in sensor networks. The channels used by this technology often suffer interference from Wi-Fi and Bluetooth. Suffer in data reliability on account of frequent occurrences of external interference. WSNs need to be able to operate reliably in the presence of such interference. Sensor node transceivers offer communication on different non-overlapping frequency channels. This multichannel feature could be leveraged to ensure a seamless operation in the face of severe external interference. Reducing packet loss (hence retransmissions) and increasing the efficiency of spectrum usage. As a multichannel solution for mitigating external interference, MCRP is introduced as a routing protocol that communicates across layers. Multichannel communication in wireless networks can alleviate the effects of interference which as a result can improve the network efficiency and stability, link reliability and minimize latency and minimal number of failures. Multichannel Cross-Layer Routing Protocol concentrates on finding channels for the nodes that are free from or have low interference. It allows the allocation of these channels in a way likely to minimize the chances of nodes which are physically near to communicate on the same channel. Hence, it reduces cross interference between different pairs of nodes.

We present MCRP, a decentralized cross-layer protocol with a centralized controller. Our cross layer multi-channel protocol focuses on the network and appli-

cation allows. This allows channel assignment decisions to be made thoroughly without being limited by the low layer complexity. The system has two parts: a central algorithm which is typically run by the LPBR and selects which channel each node should listen on; and a protocol which allows the network to communicate the channel change decision, probe the new channel and either communicate the success of the change or fall back to the previous channel.

In the rest of this chapter, (the outline of the chapter is as follows)

3.1 MCRP Design

Before presenting the design on MCRP and the main components, we introduce the general design goals (several crucial observations). The design of the multichannel protocol is motivated/based on several crucial observations:

i. Channel assignment - Sensors have limited memory and battery capabilities. In order to maximize the sensors lifetime, a centralized LPBR that has larger memory and fully powered in used for decision making. LPBR has complete knowledge of the topology which enables it to make good channel assignment decisions based on a two-hop colouring algorithm centralized; thus nodes computation is transferred to LPBR.

ii. Interference - External interference cannot be predicted, thus channels cannot be allocated beforehand as it varies over time and locations. It is impossible to determine a single channel that is free from interference at any location. Our protocol checks the channel condition each time before deciding on a channel change to reduce interference and maximize throughput. //Transmission collisions may occur in wireless networks, especially with bursty traffic that may be present in sensor network. //Interference may lead to packet losses which need to be catered for with suitable retransmission mechanisms.

iii. Frequency diversity - Multichannel increases the robustness of the network towards interference. However, applying multichannel to the existing RPL may hinder detection of the new nodes and cause problems for maintaining the RPL

topology. We overcome thus problem by two mechanisms. (((((??)Existing nodes maintain a table of channels on which their neighbours listen and use unicast to contact those nodes. New nodes listen on a Contiki default channel (26) and when connecting search through all channels. As in RPL, periodically all nodes broadcast RPL control messages on the default channel in an attempt to contact new nodes.

Our work (make use) of existing standards and focus on improvement that can be used with the standards. MCRP is compatible with RPL with minor changes in order to be able to be used as a multichannel protocol as MCP concentrates on cross layers between the network and application layers. Minor changes on the MAC layer (ContikiMAC that is energy efficient DETAILS???) in order to be compatible with multichannels to be able to change to the correct channel when transmitting/retransmitting by accessing the channel information that are stored on the network layer.

3.2 Channel Selection Strategy

One main advantage of the system we propose is generality. Any algorithm can be used at the LPBR to assign channels. In this paper we use a two-hop colouring algorithm to select a channel to be assigned to a node. The two-hop colouring algorithm attempts to ensure that nearby nodes do not communicate on the same channel and risk interfering with each other. The protocol is inspired by the graph colouring problems [28]. The core idea is that no node should use the same listening channel as a neighbour or a neighbour of a neighbour (two hops). This allows fair load balancing on the channels and reduces channel interference that could occur when two nearby nodes transmit together on the same channel. The nodes used in this paper have a transmission range of approximately 20-30 metres indoors and 75-100 metres outdoors [29]. It could be the case that many nodes in a sensor network are in the transmission range of each other and potentially interfered with.

All nodes are initialised to channel 26 which is the common default channel for Contiki MAC layer since it often has fewer interference problems with Wi-Fi and

other sources. The studies in [7, 8, 6] use a set list of whitelisted channels in their experiments and have channel 26 in common. The usual RPL set up mechanism is used to exchange control messages that are required to form an optimised topology before channel assignments can take place. The nodes will only be on the same channel once during the initial setup. This enables the node to detect and find nearby neighbours that are in range before it can decides on the best route based on the list of neighbours it can be connected to.

In the two-hop colouring algorithm, the LPBR chooses a node to which it will assign a channel to listen on. The selection is random (from channels 11 to 26) based on the full range available [1].

The value is random - no need to pseudo-random number generators to avoid node persistently generating the same numbers as the channel that is bad for one node might gives good result for another node depending on the location of the node which might not be within where the channel is bad at.

The protocol checks neighbours and neighbours of neighbours to see if any of those are listening on this channel already. If any are, a new channel is picked from the remaining list of available channels. If the LPBR has knowledge of existing bad channels then those channels can be blacklisted. Knowledge of channel interference which is gained by probing can be used to decide that a channel should not be used. If a channel is found then the channel switching protocol is triggered. If no channel can be found meeting these conditions, the current channel is kept.

The node selection algorithm must only attempt one channel change at a time to ensure probing is done on the correct new channel and for the node to finalise the channel to be used before another node attempts a channel change. The protocol ascertains that the channel change attempt will always result in a message returned to the LPBR either confirming the new channel or announcing a reversion to the old channel. Until one or other of these happens, no new channel change will be made to enable the neighbours transmitting on the correct channel.

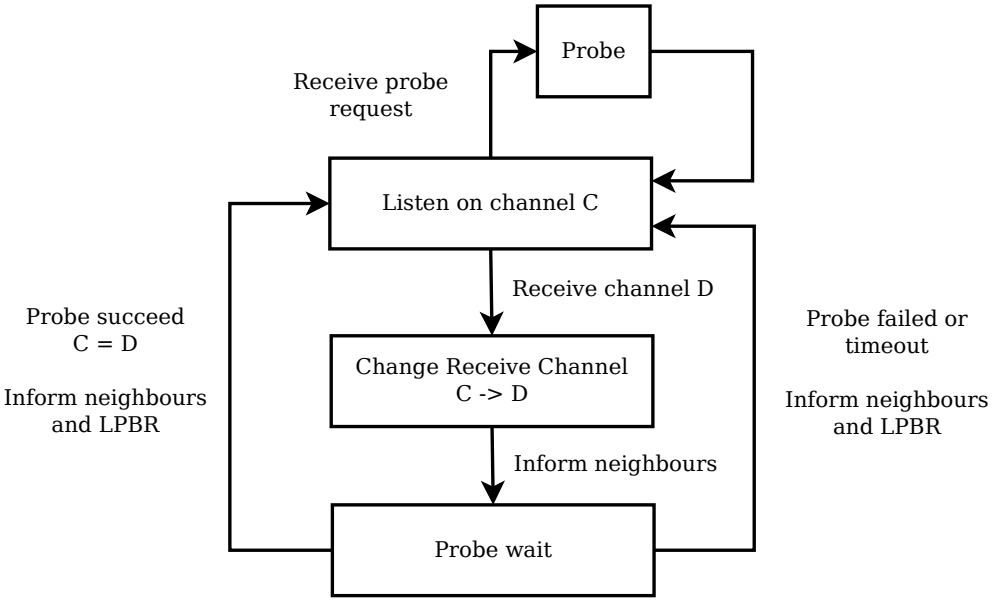

Figure 3.1: Channel switching processes

3.3 Channel Switching

Figure 3.1 shows the state machine for the channel switching protocol. As explained in the previous section, a choice of a new channel by the channel selection protocol causes a change channel message to be sent to the appropriate node. Upon receiving a channel change message, a node *N* stores its current channel *C* and communicates to all its neighbours the new channel *D* that it wishes to change to. Those neighbours will update their neighbour tables to ensure that they now send to node *N* on channel *D*. The node *N* begins the channel quality checking process with each neighbour in turn by sending them a probe request. If this process fails for any neighbour then the node reverts to channel *C*. If all channel quality checks succeed, the node *N* will listens on channel *D*. In both cases, node *N* informs its neighbours of the decision to channel *C* or *D* and informs the LPBR of the channel checking results. The channel checking process uses probe packets that might interfere with other transmissions temporarily. However, it is important to emphasise that the network remains fully functional and connected at all stages of this protocol.

3.4 Channel Quality Checking

The channel quality checking is invoked each time a node changes channel after receiving a message from the LPBR. A node *N* changing to channel *D* informs all neighbours in turn, of the new channel *D* it will be listening on as described in the previous section. It then enters the *Probe Wait* state and begins channel quality checking with each tree neighbour in turn. In describing the channel quality checking process, it is worth emphasising the distinction between neighbours and tree neighbours. Node neighbours are all nodes that a given node knows it could transmit to. Tree neighbours are the nodes that a node does transmit to through the topology formed by the RPL protocol.

In the *Probe Wait* state, node *N* sends a *Probe* message to each neighbour in turn. The neighbours respond to the message by sending eight packets to *N* on the new channel *D*. The buffer can accommodate eight packets at a time. As the packets might not be sent immediately due to wakes up and collisions, sending more packets would have the risk of being dropped. The condition of the channel is further investigated through the number of retransmissions and packet collisions of the probing packets for accuracy of the channel condition.

If the probing process times out (because of some communication failure) or the number of probe packets received is above a threshold (currently set to 16, including retransmissions and collisions) then node *N* immediately exits *Probe Wait* state and reverts to channel *C* its previous channel.

All neighbours are informed of the change back to channel *C* and the LPBR is informed of the quality check failure with a summary of all probes received. If, on the other hand, all channel quality checks succeed, the change to channel *D* becomes permanent for node *N* and it informs the LPBR of the results of the probing (numbers of packets received) and the channel change.

Probing is essential to make the channel change decision. It gives a quick overview of the channel condition based on the number of probing messages received. It is worth noting that probing is only done between the node and the tree neighbours. Neighbours that are not tree neighbours will not use the node as a route

during their transmission thus, there is no need for probing to take place with those neighbours. However, the neighbours still need to know the channel value given that RPL control messages are sent to neighbours directly without using the routes.

3.5 Reconnection Strategy

RPL topology stability (using routing metric) remains the same in multi channel [30, 25]. The nodes can still change the parents as usual as all neighbours know each other new channels. The neighbours that are not part of the route do not probe the parent when making the channel decision. However, the neighbours are informed of any channel changes. This enables the topology to be optimised when communication fails and further improved through MCRP as the nodes have knowledge of the listening channels of all other nodes within the range. If a new node tries to join the topology, it sends a RPL control message through all channels as the listening nodes are unlikely to be on the default channel. The listening nodes send a broadcast on a default channel to discover new nodes (in Contiki default, new nodes will start on channel 26) and send RPL messages through unicast when the neighbours are known to reduce unnecessary transmissions in broadcast. New nodes and nodes which fall off the network can now rejoin on many potential channels.

Chapter 4

Implementation

MCRP is implemented on the TelosB mote platform. It uses Contiki operating system as the software development (platform?) with full support of the standard IPv6. The implementation of MCRP are describe, including changes that were done (undertaken) in addition to the default parameters and settings for Contiki.

Contiki is a lightweight operating system with support for dynamic loading and replacement of individual programs and services. The purpose of the Contiki design is to reduce size and complexity, as well as to preserve flexibility. A running Contiki system consists of the kernel, libraries, the program loader and a set of processes (may be either an application program or a service - functionality used by more than one application process e.g. includes communication protocol stacks, sensor device drivers) [9].

4.1 Protocol Stack (changes at MAC, RT, NBR TB, BR etc.)

MCRP is a cross layer protocol implemented on Contiki version 2.7. (explain what is cross layer? why do cross layer? FIGURE OF STACKS)

Contiki is a four layers network stack; network, MAC, radio duty cycling (RDC) and radio layers. The network layer includes support for TCP and UDP, IPv6, IPv4, 6lowpan and RPL (routing). Traditional TCP/IP implementations have required far too much resources which is impossible to fit in a sensor that has limited RAM capabilities (RAM is the most scarce resource). uIP [31] is a small RFC-

Contiki	IoT/IP	Applications
Network	Application	HTTP
	Transport	TCP, UDP
	Network, Routing	IPv6, IPv4, RPL
	Adaptation	6LoWPAN
MAC	MAC	CSMA/CA
RDC	Duty Cycling	ContikiMAC
Radio	Radio	IEEE 802.15.4

Table 4.1: Contiki Network Stack

compliant TCP/IP stack that makes it possible for Contiki to communicate over the Internet. uIP implementation is designed to have only the absolute minimal set of features needed for a full TCP/IP stack such as IP, ICMP, UDP and TCP protocols. The uIP is mostly concerned with the TCP and IP protocols and upper layer protocols [19, 20].

////BUFFER MANAGEMENT - since it's related to uIP (Chameleon, Rime etc)

It uses (EXPLAIN CONTIKIMAC - refer to contikimac paper; why it's good, how it works). Also about retransmission and buffers. (maybe at next section???) The transmitting channel is set at the MAC layer as packets are not send immediately if there are packets being queued. The channel is reset to the transmitting channel before it tries to send and it is then reset to the listening channel to wait and listen to any packets that is being sent to the node.

The default ContikiMAC is a single channel protocol. It is modified to be able to work with multi channel nodes while (stick/hold/is) on the same principle of a low power ContikiMAC (minor changes to support multi channel without changing the main purpose of ContikiMAC).

RPL border router is used as LPBR in order to move most processing decisions on a PC as it has more RAM and better processing capabilities than a sensor. (Explain BR-SR how it works!) TelosB has limited RAM and ROM of 10K bytes and 48K bytes of flash memory [29]. By using a border router, this allows channel changing to be decided in real time without draining the memory and battery on a sensor. The border router also acts as the root of the tree. The border router will

setup the IPv6 prefix of the network and will initiate the creation of the RPL routing tree. It (tunslip6) sets up an interface on the Linux IP stack and connects this interface via a socket to the border router node. Border router is used to bridge the wireless IPv6 network to a PC via serial link which enables the IPv6 network traffic to reach outside network and the Internet. A node is used as a wireless interface (IEEE 802.15.4 to enable the serial socket server), a host machine as border router to bridge the wireless IPv6 network to outside network and the Internet.

Serial Line Internet Protocol (SLIP) //need reference!!!] is used for the communication between the sink and the device which it connected to such as an embedded PC. SLIP is commonly used to encapsulate IP packets for transmission across the serial line of micro-controller devices. SLIP has a low complexity and small overhead. For the communication between the devices (embedded PCs), any reliable network can be used (e.g. Ethernet). The sinks are connected to an embedded PC which contains an Ethernet interface. The communication between the sink (sensor node) and the embedded PC makes use of SLIP. Contiki already provides support for SLIP communication and includes a tunslip tool (need reference!!!) which make it possible to communicate with devices using SLIP. The tool constructs a SLIP tunnel between a physical serial interface and a virtual network adaptor. By using tunslip the communication between the sink and the embedded PC is facilitated [32].

Tunslip is a too used to bridge IP traffic between a host and another network element, typically a border router, over a serial line. Tunslip creates a virtual network interface (tun) on the host side and uses SLIP (serial line internet protocol) to encapsulate and pass IP traffic to and from the other side of the serial line. The network element sitting on the other side of the line does a similar job with it's network interface. The tun interface can be used like any real network interface: routing, traffic forwarding etc [33].

RPL is used as the routing protocol. (explain how RPL works briefly since it's explained in LITERATURE REVIEW).

We tailored RPL control messages to be able to accommodate MRCP proposal

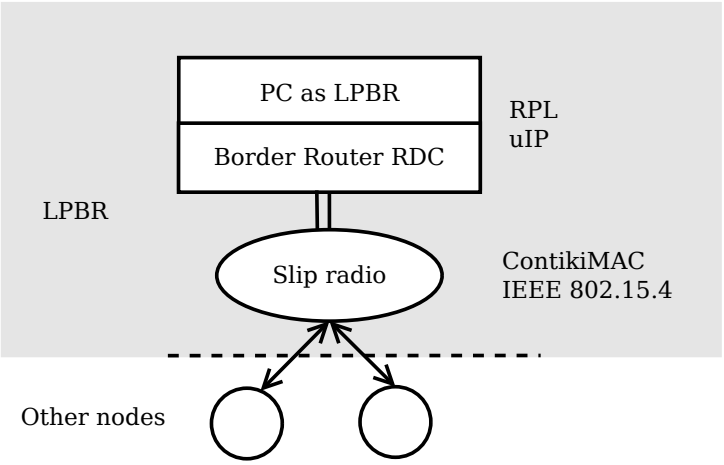

Figure 4.1: Low power border router

by enabling unicast to know neighbours and broadcast to detect new nodes to join the tree.

4.2 MCRP Implementation

MCRP is implemented as an extension to the existing implementation of RPL with ContikiMAC; to enable multi channel. The protocol is implemented by tailoring existing code of ContikiMAC, network layer and RPL.

//separate into 2; setCh and xSetCh as LPBR is separated with BR and SR //access Network layer - Neighbour table, set channel

4.2.1 Low Power Border Router

As sensors have limited memory, we decided to move most decision making (processing decisions) at LPBR to a PC as it more RAM and better processing capabilities. This enables us to have more thorough processes. As describes previously, the protocol stack is divided into two main parts as shown is figure 4.1 where the PC is responsible as the application, transport, network and routing layers while a sensor is set as the wireless interface to enable the PC to communicate with the other nodes. LPBR is a special case as channel changes at LPBR is not as direct as other sensor nodes due to this two parts. However, it works similar ways to the other nodes.

LPBR main responsibility is to decide on the new channel selection. LPBR has no knowledge of all the channels condition at this point, thus, a channel is selected at random. LPBR keeps the results from the channel changes processes and based on it when selecting a new channel for the next node to ensure the new channel is at least two-hop away from another node using the same channel. This is done to ensure that the nearby nodes do not communicate of the same channel and risk interfering with each other.

The pseudo-code of the two-hops colouring algorithm that we implemented in new channel selection is shown in Algorithm 1. When the new channel is selected, LPBR will send the value to the intended node.

Algorithm 1 Pseudo-code for two-hop colouring algorithm

Notations
R is a node that is a Route
N is a node Neighbour
RN is the Route's Neighbour node
currentCh is the node current listening channel
newCh is the new channel the node will change to
Pseudo-code
if *R currentCh* $\neq$ *newCh* **then**
    succeed one-hop
    check all *RN* channels
    if *RN* channel $\neq$ *newCh* **then**
        succeed two-hop
        confirm *newCh*
    end if
else
    generate a new *newCh*
    update the number of *newCh* generated for *R*
    use default channel 26 is all tries fail
end if

All layers of the netstack operate on the packet buffer. One buffer holds a single packet. ***Uses a single buffer for both incoming and outgoing packets [31, 34, 35, 9]. The packet buffer only holds the current packet.

Before passing to slip-radio, the header from uip_buf is compress to $packetbuf$. [34] introduces Chameleon, a communication architecture for sensor networks consists of Rime communication stack and a set of packet transformation

modules. Rime communication stack provides a set of lightweight communication primitives ranging from best-effort anonymous local area broadcast to reliable network flooding. Protocols or applications running on top of Rime stack can implement additional protocols that are not in the Rime stack such as TCP/IP.

Rime draws heavily from communication abstractions for distributed programming where layers of simple abstractions are combined to form powerful high-level abstraction. The purpose of Rime is to simplify implementation of sensor network protocols and facilitate code reuse. The thin layers in Rime enable code reuse within the stack. An underlying MAC or link layer may chose to implement parts of the Rime stack [35].

Chameleon is a header construction. The parsing is done separately from the communication stack which allows the use of uIP or Rime communication stack that both are supported in Contiki.

6LoWPAN working group specified header compression and fragmentation for IPv6 over IEEE 802.15.4 (///REFERENCE!!!) and the IETF RoLL working group designed the RPL protocol as a proposed standard for IPv6 routing in low-power and lossy networks (LLNs). Contiki implement the necessary parts of the IPv6 protocol, IPv6 header compression and fragmentation with the 6LoWPAN adaptation layer, routing over LLNs with the RPL protocol, as well as a set of protocols from the TCP/IP protocol suite [13].

Application can use either, both or none communication stack. uIP can run over Rime. Rime can run over uIP [36].

Buffer for uIP and Rime is the same. They use the same buffer.

Chameleon does not define any packet headers but instead uses $packetattributes$, an abstract representation of the information usually found in packet headers to allow applications to access low-level information without violating the layering principle. Packet headers are produced by separate header transformation modules that transform application data and packet attributes into packets with header and payload. The use of packet attributes makes it possible to adapt the output from the protocol stack to other communication protocols such as link and MAC layer

protocols and TCP/IP. Chameleon architecture allows for sensor network protocols that are implemented on top of the architecture to take advantage of the features of underlying MAC and link layer protocols.

In the buffer management of Chameleon architecture, all packets both outgoing and incoming are stored in a single buffer [35], called the Rime buffer. The Rime buffer contains both the application data and the packet attributes. All access to the Rime buffer is done at a single priority level so no locking mechanisms need to be used.

Protocols that need to queue packets, such as MAC protocols that wait for the radio medium to be free, can allocate so-called queue buffers to hold the queued packet. Queue buffers are dynamically allocated from a pool of queue buffers. The contents of the Rime buffer, including the packet attributes, are copied into the queue buffer when it is allocated.

The new channel is stored in the buffer before the data is sent over SLIP to the radio-chip (slip-radio). As the slip radio is unable to access the neighbour table where the next hop node channel is stored, the channel value is passed through the buffer. LPBR keeps the updated value of all it's neighbours channels in the neighbour channel. Slip radio that receives the packet buffer can access the channel value and kept the value is a simplified version of the neighbour table. This is done in order to ensure that the packet that is being queued or retransmitted are send on the correct channel. The packets destination, which in this case, the next hop node is first check before the packet is transmitted each time. The MAC layer sets the channel accordingly before sending. ContikiMAC can access the simplified neighbour table as it is on the slip radio. The simplified neighbour table only keeps the information of the node neighbours and neighbours channels which are the critical information in order to transmit packets correctly. The other information that is related to the neighbours conditions are monitored at the PC.

LPBR processes are shown in 4.2. The slip-radio resets to it's listening channel after the packet is transmitted. LPBR will wait and listen to any incoming packets. *In the channel probing phase, LPBR does not take part in probing. However,

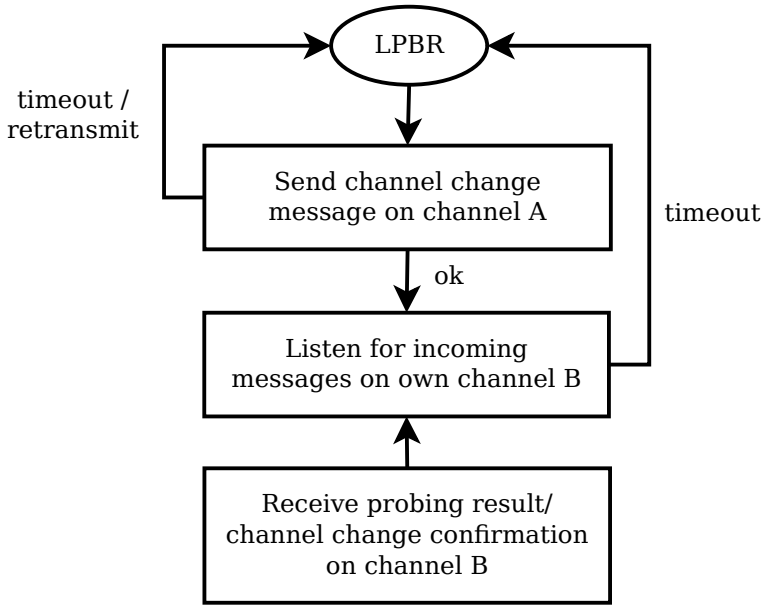

Figure 4.2: LPBR processes

LPBR is informed of the results of probing and kept a table of the probing results and channels to be able to use the information when deciding on a channel change based on the previous results of probing on the known channels.

-send to BR RDC, check nbr table for the currentCh. pass the value to s-r. save the chvalue in a simplified nbr table to allow retx/queue to send on correct ch. MAC set the channel accordingly before sending. Change to its listening channel after finish sending.

4.2.2 Other Nodes (MAC Layer?)

As MCRP is a cross-layer protocol, packets that have not been transmitted are kept in the buffer. In order for the transmission to be on the correct channel, the neighbour table in the network layer is accessed and the channel is set to the transmitting channel.

ContikiMAC has retransmitted, collisions valued. These values are used in probing to decide on the channel condition. These values are passed to the application layer to decide on channel change. The packet can be retransmitted for ( ) times before it is dropped. However, if the channel is busy, and the packet has not

been sent (collisions before sending), it can stay in the loop for a long time.

All nodes keep the neighbours channel in the neighbour table. Unlike LPBR, other nodes have all the layers within the node. This makes change changes less complicated, however, the nodes are being limited by the number of RAM is has which resulted in results of probing to be kept by the centralised LPBR. The node however, keep probing results temporarily before deciding on the channel. The node is cross-layer as it can access information at any layer.

When LPBR sends a channel change message to the destination node, the destination node will send a packet to acknowledge the channel change message. If LPBR does not receive the message, the channel change message is retransmitted. LPBR is then wait and listens for any incoming packets. At this point, channel changes processes will take place between the node and its neighbours.

The node sends the new channel to all the neighbours. At this point, the new channel is not check. However, all neighbours need to know the new channel as the node will be listening on the new channel. Otherwise, packets cannot be received by the node. The neighbours that receive the node channel will update their neighbour table at the application layer. As this is an important step in order to reduce the number of packet loss due to sending on the wrong channel, neighbours will send an acknowledgement of the new channel. Otherwise, it will be retransmitted.

The node will then send a message to the neighbour that is a route node to start sending probing messages on the new channel. Not all neighbours are used routes. The neighbours are chosen as route based on RPL (//describe RPL briefly how if choose route). The node will listens on the new channel and wait for the probe message. The route node starts sending probing messages every 3 seconds to allow retransmission or collision that could happen due to the channel being busy (////what is retransmission? what is collision??? how it happens? how long? collision has no time out!). As the we are sending a small number of probing, to increase the channel reliability of the probing, we also take into account the number of retransmission and collision that happen. As the retransmission and collision is a link layer process, the values are kept in a table and is sent in the next probe

message. This is because the value is only valid for that run. It gets reset each time a new packet is sent or received. The table is accessed at the application layer before the next probing message is sent. The probing message includes the number of tries the previous packet had taken before it is received. These values are used to decide if the channel is better than the current channel by giving a good probing result, meaning less retransmission.

The node keeps the value of probing messages it receives. It sends all the probing results to LPBR. LPBR could use the information to decide on a channel or blacklist bad channels. It then use the values to decide whether the new channel is better than the previous channel by setting a threshold. The node then send the channel that it confirms to be listening on to all neighbours. The channel can be the new channel or reverting to the previous channel. The neighbours will send an acknowledgement back to the node. This is also important to ensure that all neighbours could communicate with the node on the correct channel. The neighbours will update their neighbour table of the node channel.

4.2.3 Routing Layer - Neighbour Discovery

///net/rpl/rpl-icmp6.c and rpl-timers.c - the changes done!!

////DIO UNICAST RPL sends the control messages as broadcast. However, as we are now dealing with multi channels, using broadcast for all channels would waste the bandwidth and costly as it would take a longer time to go through all channels. It would also cause congestion and the node to be on the broadcast channel and not ready on it's listening channel to be able to receive any incoming packets as it has not finish with the control message broadcast. RPL DIO message is able to deal with either broadcast or unicast. By default, broadcast was used as RPL is usually used with a single channel MAC. We enable the unicast DIO.

////MULTI CHANNEL DIS If a new node tries to join the tree, it will send a DIS message on all channels until it finds the neighbours.

4.3 Memory Footprint/Setup Overhead?

-how many packets more than usual? -memory consumption?

Chapter 5

Results and Discussions

-include prelim results from testbed

The testbed provides the ability to validate performance in real wireless channel environments. However, relying solely on testbed results complicates the task of examining the network's behaviour and diagnosing the root causes of performance degradations. For this purpose, while simulators lack the realism of testbeds, we leverage their ability to provide repeatability and full control over the parameter space. Furthermore, unlike a testbed setup, which is by necessity confined both in time and in space, a simulation can be made arbitrary complex (introduce interference**. Although simulators cannot be expected to fully mimic the behaviour of real wireless channels, they are effective in discovering the root causes underlying the performance degradations caused from implementation-specific design choices [13].

5.1 Experimental Setup

We evaluate the protocol in the Cooja simulated environment with emulation of TMote sky nodes that feature the CC2420 transceiver, a 802.15.4 radio. The nodes run on IPv6, using UDP with standard RPL and 6LoWPAN protocols. The network consists of 31 nodes which are used to run the simulation where we have 1 border router node, 16 interference nodes, and 14 duty cycled nodes that act as UDP clients to send packets to LPBR spanning over 20-30 metres between each node. RPL border router is used as LPBR in order to move most processing decisions on a

PC as it has more RAM and better processing capabilities than a sensor. TelosB has limited RAM and ROM of 10K bytes and 48K bytes of flash memory [29]. By using a border router, this allows channel changing to be decided in real time without draining the memory and battery on a sensor. The border router also acts as the root of the tree.

We simulated a controlled interference node that generates semi-periodic bursty interference to resemble a simplified Wi-Fi or Bluetooth transmitter on several channels at random. We use the interference model proposed in [2]. The interference has two states, a clear state and an interference state. In the interference state, the interference node generates packets for a time that is uniformly distributed between $9/16$ seconds and $15/16$ seconds. In the clear state the interferer produces no packets and stays in this state for between $3/4 * clear\_time$ and $5/4 * clear\_time$ where *clear_time* refers to the rate of interference. We use multiple channels interference in our simulation to show our hypothesis that our multichannel protocol can help avoid interference. We consider the scenario where ContikiMAC with RPL system is subject to interference on its channel after set up has successfully completed so the RPL set up is allowed to complete before interference begins.

We use an end-to-end packet delivery performance metric to evaluate MCRP. The transmission success rate is calculated from the sender to the receiver over multiple hops. We also look at the loss over time to observe the protocol performance in the presence of interference. We considered two multiple channels interference scenarios; (1) extreme and no interference rate on 8 channels each and (2) extreme, moderate, mild and no interference rate on 4 channels each. The interference channels are randomly chosen from the available 16 channels and the same interference channels and rates are used throughout the experiments. However, channel 26 is kept clear from interference in order to ensure RPL set up is unaffected. In scenario 1, we fixed the interference rate to extreme and no interference to observe the effect it has on channel changing decisions. In scenario 2, we vary the interference rate to observe how MCRP copes in deciding a channel when there is more interference than scenario 1 but with less interference intensity.

We run the simulation for a duration of 45-60 minutes to send 210-560 packets. When the nodes are switched on for the first time, all nodes are initialised to channel 26, the default channel for Contiki MAC layer. RPL is allowed five minutes to set up (which is ample time). RPL topology is formed in a minute. We wait for another five minutes to allow trickle timer to double the interval length so that RPL control messages are not being sent frequently. We then let our multichannel protocol runs for 25 minutes. In our 15 nodes simulation, our protocol takes 20-25 minutes to run the channel change set up. We allow another 5 minutes wait time if retransmissions happen. In a single channel simulation, all the nodes are changed to channel 22 after 5 minutes of RPL set up time. This allows RPL to have enough time to discover all nodes to form an optimised topology. The topology formation does not form completely if the interference node interferes from the beginning. The interference node starts sending packets to interfere after 3 minutes the system is switched on so that the interference channel is involve in the channel changes decision. We proved that our protocol tries to avoid changing to the interference channel through time out and probing failures. After 30 minutes, the client nodes will send a normal packet periodically every 30-60 seconds to LPBR. This is done in order to avoid collision of the nodes sending at the same time.

The simulations are repeated ten times. In all plots, the mean value of the ten simulations is plotted with error bars corresponding to one standard deviation in either deviation to give a measure of repeatability. The plots are of the proportion of received packets (from 0% to 100%) against time where the loss is measured over the previous time period. The x-value is shifted slightly left and right to prevent error bars overlapping.

5.2 Evaluation

The performance of MCRP is compared against the standard ContikiMAC with RPL. We analyse MCRP using an end-to-end packet delivery performance metric.

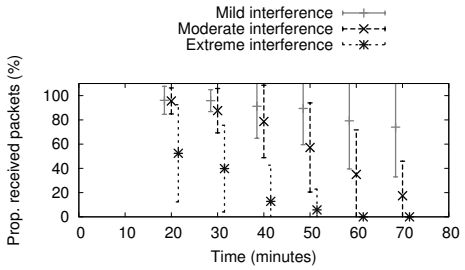

Figure 5.1: Level of packet loss for mild, moderate and extreme interference levels using single channel

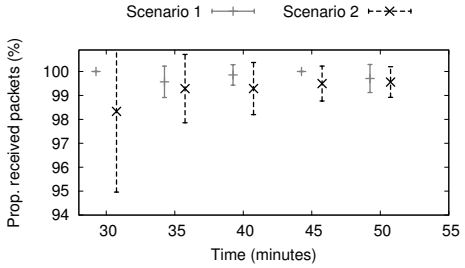

Figure 5.2: Level of packet loss for scenario 1 and scenario 2 using multi channel

5.2.1 Packet loss rates with single channel RPL versus multi-channel

The performance obtained in ContikiMAC with RPL (single channel) is compared with our protocol in terms of packet loss rate. As described previously, levels of interference used (referred to as *clear_time* in [2]) vary between 100% (no interference), 75% (mild), 50% (moderate) and 25% (extreme) where the percentage is the ratio of the time the channel is clear for transmission. All our tests have a common format: the RPL procedure is allowed to set up without interference in order not to bias subsequent tests. Then the interferers begin to operate with a constant level (none, mild, moderate or extreme).

Figure 5.1 shows the results for ContikiMAC with RPL protocol. It can be seen that the level of packet loss varies considerably between experiments (the error bars are always large). It can also be seen that even for mild interference there is considerable loss and this gets worse as time proceeds. In the extreme interference case the loss always goes up until no packets are received. For mild interference the system evolves until it is losing around 20% of packets but this can increase.

For our new multiple channel protocol we consider two interference scenarios. In scenario 1 half the channels (including the original channel) have no interference at all and half the channels have extreme interference. In scenario 2, four channels (including the original channel) have no interference, four have mild, four moderate and four extreme interference. Figure 5.2 shows multi channel results for these two scenarios. In scenario 1 the protocol performs extremely well, the packet loss is near zero and the protocol successfully detects channels with interference. Scenario 2 has similar results as in scenario 1. The protocol does well at reducing the effects of interference and could detect moderate and mild interference.

In MiCMAC [8], it is stated that MiCMAC has a transmission success rate of 99% when using four channels. However, when more than four channels are used (8 or 16 channels), MiCMAC performance degrades to approximately 88% (16 channels) due to interference channels. The interference model that MiCMAC uses is different than ours. They compare the result with Chrysso where Chrysso has a transmission success rate of approximately 88% for 4 and 8 channels and suffers greatly in the case of 16 channels with 60% success rate. Our protocol on the other hand, shows greatly reduced loss rate with any number of channels at approximately 99%.

5.2.2 Setup Overhead

Obviously the system of changing channels and probing to see if a channel is free of interference introduces a certain amount of overhead into the protocol. This takes the form of (a) extra messages passed and (b) extra time taken to set up. Default RPL on ContikiMAC for the topology considered in these experiments completed its set up using 276 packets. Our multi-channel protocol completed its set up in 716 packets, that is an overhead of 440 packets on top of RPL. This overhead comes from the channel changing messages to nodes and neighbours, probing messages, channel confirmation messages and acknowledgement packets which are required to ensure a thorough channel change decision. However, it is worth mentioning that this is a one-off cost. This represents (in this experimental set up) approximately one hour of extra packets in the situation of a deployment that is meant to work for

weeks or months. In terms of set up time, our protocol begins to change channels only when the RPL set up process is complete (or at least stabilises). The set up time is 1154 seconds beyond the RPL set up time of 286 seconds. However, it should be noted that, in fact, our system remains fully functional and capable of sending packets during the set up so this set up overhead does not matter to data transmission. Therefore we conclude that data sending costs (extra packets) of set up are negligible in the context of a deployment that will last more than a day. The extra set up time is also negligible within this context and furthermore does not degrade performance of the network during this set up phase.

Chapter 6

Energy vs Loss Tradeoff

-requires more energy (to do MCRP) but reduce retransmissions in the long term -how much energy than usual? -improvement in loss when using MCRP?

Chapter 7

Future Work

-include gantt chart

7.1 Conclusions

7.2 Future Works

Bibliography

[1] IEEE. IEEE standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (LR-WPANs). *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pages 1–314, Sept 2011.

[2] Carlo Alberto Boano, Thiemo Voigt, Nicolas Tsiftes, Luca Mottola, Kay Römer, and Marco Antonio Zúñiga. Making sensornet MAC protocols robust against interference. In *Proceedings of the 7th European Conference on Wireless Sensor Networks*, EWSN'10, pages 272–288, 2010.

[3] M. Petrova, Lili Wu, P. Mahonen, and J. Riihijarvi. Interference measurements on performance degradation between colocated IEEE 802.11g/n and IEEE 802.15.4 networks. In *Networking, 2007. ICN '07. Sixth International Conference on*, pages 93–93, April 2007.

[4] IEEE. IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, March 2012.

[5] Yafeng Wu, J.A. Stankovic, Tian He, and Shan Lin. Realistic and efficient multi-channel communications in wireless sensor networks. In *IEEE INFOCOM 2008. The 27th Conference on Computer Communications*, April 2008.

[6] Thomas Watteyne, Ankur Mehta, and Kris Pister. Reliability through frequency diversity: Why channel hopping makes sense. In *Proceedings of the*

6th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, pages 116–123, 2009.

[7] V. Iyer, M. Woehrle, and K. Langendoen. Chrysso - a multi-channel approach to mitigate external interference. In *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 449–457, June 2011.

[8] B. Al Nahas, S. Duquennoy, V. Iyer, and T. Voigt. Low-power listening goes multi-channel. In *2014 IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 2–9, May 2014.

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

[10] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with COOJA. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641–648, Nov 2006.

[11] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Comput. Netw.*, 52(12):2292–2330, August 2008.

[12] Ian F. Akyildiz, Tommaso Melodia, and Kaushik R. Chowdhury. A survey on wireless multimedia sensor networks. *Comput. Netw.*, 51(4):921–960, March 2007.

[13] JeongGil Ko, Joakim Eriksson, Nicolas Tsiftes, Stephen Dawson-Haggerty, Jean-Philippe Vasseur, Mathilde Durvy, Andreas Terzis, Adam Dunkels, and David Culler. Beyond Interoperability: Pushing the Performance of Sensor Network IP Stacks. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 1–11, New York, NY, USA, 2011. ACM.

[14] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Comput. Netw.*, 38(4):393–422, March 2002.

[15] Yi-hua Zhu, Wan-deng Wu, Jian Pan, and Yi-ping Tang. An energy-efficient data gathering algorithm to prolong lifetime of wireless sensor networks. *Comput. Commun.*, 33(5):639–647, March 2010.

[16] A. Bachir, M. Dohler, T. Watteyne, and K.K. Leung. MAC essentials for wireless sensor networks. *Communications Surveys Tutorials, IEEE*, 12(2):222–248, Second 2010.

[17] Lei Tang, Yanjun Sun, O. Gurewitz, and D.B. Johnson. PW-MAC: An energy-efficient predictive-wakeup mac protocol for wireless sensor networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 1305–1313, April 2011.

[18] Adam Dunkels. The ContikiMAC radio duty cycling protocol. Technical Report T2011:13. ISSN 1100-3154 `http://dunkels.com/adam/dunkels11contikimac.pdf`, 2011.

[19] Contiki. Contiki 2.6. http://contiki.sourceforge.net/docs/2.6/, Jul 2012.

[20] Contiki. Contiki 2.6 The uIP TCP/IP stack. http://contiki.sourceforge.net/docs/2.6/a01793.html, Jul 2012.

[21] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, 2009.

[22] R. Fonseca, O. Gnawali, K. Jamieson, P. Levis S. Kim, and A. Woo. TEP 123: The Collection Tree Protocol, Aug 2006.

[23] Tsvetko Tsvetkov. RPL: IPv6 routing protocol for low power and lossy networks. *Sensor Nodes–Operation, Network and Application (SN)*, 59:2, 2011.

[24] Nicolas Tsiftes, Joakim Eriksson, Niclas Finne, Fredrik Osterlind, Joel Hglund, and Adam Dunkels. A framework for low-power IPv6 routing simulation, experimentation, and evaluation. In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, pages 479–480, New York, NY, USA, 2010.

[25] T Winter, P Thubert, T Clausen, J Hui, R Kelsey, P Levis, K Pister, R Struik, and J Vasseur. RPL: IPv6 routing protocol for low power and lossy networks, RFC 6550. `https://tools.ietf.org/html/rfc6550`, 2012.

[26] Simon Duquennoy, Olaf Landsiedel, and Thiemo Voigt. Let the tree bloom: Scalable opportunistic routing with ORPL. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 2:1–2:14, 2013.

[27] Philip Levis, T Clausen, Jonathan Hui, Omprakash Gnawali, and J Ko. RFC6206: The trickle algorithm. https://tools.ietf.org/html/rfc6206, 2011.

[28] T.R. Jensen and B. Toft. *Graph Coloring Problems*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2011.

[29] Crossbow Technology. TelosB - TelosB mote platform. Document Part Number: 6020-0094-01 Rev B.

[30] J Vasseur, M Kim, K Pister, N Dejean, and D Barthel. Routing metrics used for path calculation in low power and lossy networks. https://tools.ietf.org/html/rfc6551, 2012.

[31] Adam Dunkels. Full tcp/ip for 8-bit architectures. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, MobiSys '03, pages 85–98, New York, NY, USA, 2003. ACM.

[32] David Carels, Niels Derdaele, EliDe Poorter, Wim Vandenberghe, Ingrid Moerman, and Piet Demeester. Support of multiple sinks via a virtual root for

the rpl routing protocol. *EURASIP Journal on Wireless Communications and Networking*, 2014(1), 2014.

[33] FIT IoT-LAB. Building Contiki's tunslip6. https://www.iot-lab.info/tutorials/build-tunslip6/.

[34] Adam Dunkels, Fredrik Österlind, and Zhitao He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, SenSys '07, pages 335–349, New York, NY, USA, 2007. ACM.

[35] Adam Dunkels. Rime - a lightweight layered communication stack for sensor networks. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session, Delft, The Netherlands*, January 2007.

[36] Adam Dunkels. Contiki Crash Course. KTH Royal Institute of Technology, October 2008.

[37] Thang Vu Chien, Hung Nguyen Chan, and Thanh Nguyen Huu. A comparative study on operating system for wireless sensor networks. In *2011 International Conference on Advanced Computer Science and Information System (ICACSIS)*, pages 73–78, December 2011.

[38] Lanny Sitanayah, Cormac J. Sreenan, and Szymon Fedor. A cooja-based tool for maintaining sensor network coverage requirements in a building. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 70:1–70:2, 2013.

[39] Omprakash Gnawali. The minimum rank with hysteresis objective function, RFC6719. `https://tools.ietf.org/html/rfc6719`, 2012.

[40] Pascal Thubert. Objective function zero for the routing protocol for low-power and lossy networks (RPL), RFC6552. `https://tools.ietf.org/html/rfc6552`, 2012.

[41] Luigi Alfredo Grieco Thomas Watteyne, Maria Rita Palattella. Using IEEE802.15.4e time-slotted channel hopping (TSCH) in an internet of things (IoT): Problem statement. `https://tools.ietf.org/html/rfc7554`, May 2015.

[42] Ozlem Durmaz Incel, Lodewijk van Hoesel, Pierre Jansen, and Paul Havinga. MC-LMAC: A multi-channel MAC protocol for wireless sensor networks. *Ad Hoc Netw.*, 9(1):73–94, January 2011.

[43] Youngmin Kim, Hyojeong Shin, and Hojung Cha. Y-MAC: An energy-efficient multi-channel MAC protocol for dense wireless sensor networks. In *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pages 53–63, April 2008.

[44] A. Sivanantha, B. Hamdaoui, M. Guizani, Xiuzhen Cheng, and T. Znati. EM-MAC: An energy-aware multi-channel MAC protocol for multi-hop wireless networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, pages 1159–1164, Aug 2012.

[45] Asaduzzaman and Hyung Yun Kong. Energy efficient cooperative LEACH protocol for wireless sensor networks. *Communications and Networks, Journal of*, 12(4):358–365, Aug 2010.

[46] S. Lindsey and C.S. Raghavendra. PEGASIS: Power-efficient gathering in sensor information systems. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 3, pages 3–1125–3–1130 vol.3, 2002.

[47] Roman Lim, Federico Ferrari, Marco Zimmerling, Christoph Walser, Philipp Sommer, and Jan Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, IPSN '13, pages 153–166, New York, NY, USA, 2013. ACM.

[48] Joris Borms, Kris Steenhaut, and Bart Lemmens. Low-overhead dynamic multi-channel MAC for wireless sensor networks. In *Proceedings of the 7th*

European Conference on Wireless Sensor Networks, EWSN'10, pages 81–96, 2010.

[49] A.A. Aziz, Y.A. Sekercioglu, P. Fitzpatrick, and M. Ivanovich. A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks. *Communications Surveys Tutorials, IEEE*, 15(1):121–144, First 2013.

[50] An-Feng Liu, Peng-Hui Zhang, and Zhi-Gang Chen. Theoretical analysis of the lifetime and energy hole in cluster based wireless sensor networks. *Journal of Parallel and Distributed Computing*, 71(10):1327 – 1355, 2011.

[51] Yunxia Chen and Qing Zhao. On the lifetime of wireless sensor networks. *Communications Letters, IEEE*, 9(11):976–978, Nov 2005.