Transfer Thesis

Noradila Nordin

Department of Electronic & Electrical Engineering

University College London

Torrington Place

London

WC1E 7JE

noradila.nordin.12@ucl.ac.uk

December 14, 2015

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

List of Figures

List of Tables

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].

2.1.1 Overview (Application Scenarios for WSNs

The application can be categorised into 5; military, environment, health, home and other commercial areas [13]

2.2 Maximize Lifetime and Minimizing Energy

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

2.2.1 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.

2.3 Multichannel Protocol (Data Link Layer)

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 ().

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.

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 [14].

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 [15, 16]. 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 [15, 16]. 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 transmitters 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 [35].

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 [40].

RPL

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 application 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.

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 [17]. 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 [18]. 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 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.

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

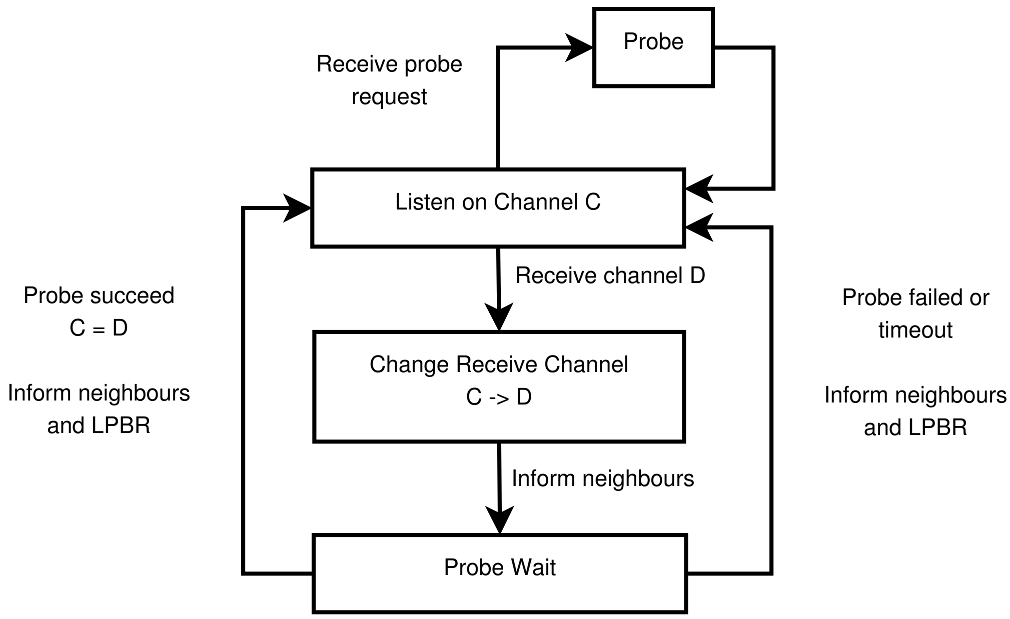

Figure 3.1: Channel switching processes

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 [19, 20]. 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.

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 is a small RFC-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 [15, 16].

It uses (EXPLAIN CONTIKIMAC - refer to contikimac paper; why it's good, how it works). Also about retransmission and buffers. (maybe at next section???)

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

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 [18]. 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 [21].

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 [22].

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 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.

4.2.1 Application Layer

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

4.2.2 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.

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

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 6

Results and Discussions

-include prelim results from testbed

6.1 Experimental Setup

6.2 Evaluation

Chapter 7

Future Work

-include gantt chart

7.1 Conclusions

7.2 Future Works

Appendix A

An Appendix About Stuff

(stuff)

Appendix B

Another Appendix About Things

(things)

Appendix C

Colophon

This is a description of the tools you used to make your thesis. It helps people make future documents, reminds you, and looks good.

(example) This document was set in the Times Roman typeface using LaTeX and BibTeX, composed with a text editor.

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

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

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

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

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

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

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

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

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

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

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

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